Reduced-Order Modeling of Reacting Flows Using Data-Driven Approaches

K. Zdybał, M. R. Malik, A. Coussement, J. C. Sutherland, and A. Parente

Abstract Data-driven modeling of complex dynamical systems is becoming increasingly popular across various domains of science and engineering. This is thanks to advances in numerical computing, which provides high fidelity data, and to algorithm development in data science and machine learning. Simulations of multicomponent reacting flows can particularly profit from data-based reduced-order modeling (ROM). The original system of coupled partial differential equations that describes a reacting flow is often large due to high number of chemical species involved. While the datasets from reacting flow simulation have high state-space dimensionality, they also exhibit attracting low-dimensional manifolds (LDMs). Data-driven approaches can be used to obtain and parameterize these LDMs. Evolving the reacting system using a smaller number of parameters can yield substantial model reduction and savings in computational cost. In this chapter, we review recent advances in ROM of turbulent reacting flows. We demonstrate the entire ROM workflow with a particular focus on obtaining the training datasets and data science and machine learning techniques such as dimensionality reduction and nonlinear regression. We present recent results from ROM-based simulations of experimentally measured Sandia flames D and F. We also delineate a few remaining challenges and possible future directions to address them. This chapter is accompanied by illustrative examples using the recently developed Python software, **`PCAfold`**. The software can be used to obtain, analyze and improve low-dimensional data representations. The examples provided herein can be helpful to students and researchers learning to apply dimensional-

K. Zdybał · M. R. Malik · A. Coussement · A. Parente (✉)
Aero-Thermo-Mechanics Laboratory, École polytechnique de Bruxelles, Université Libre de Bruxelles, Brussels, Belgium
e-mail: alessandro.parente@ulb.be

Brussels Institute for Thermal-fluid Systems, Brussels (BRITE), Université Libre de Bruxelles and Vrije Universiteit Brussel, Brussels, Belgium

J. C. Sutherland
Department of Chemical Engineering, University of Utah, Salt Lake City, UT, USA
e-mail: james.sutherland@utah.edu

N. Swaminathan and A. Parente (eds.), *Machine Learning and Its Application to Reacting Flows*, Lecture Notes in Energy 44, https://doi.org/10.1007/978-3-031-16248-0_9

ity reduction, manifold approaches and nonlinear regression to their problems. The Jupyter notebook with the examples shown in this chapter can be found on GitHub at `https://github.com/kamilazdybal/ROM-of-reacting-flows-Springer`.

1 Introduction

There is growing interest and numerous recent developments in reduced-order modeling (ROM) of complex dynamical systems (Kutz et al. 2016; Taira et al. 2017; Lusch et al. 2018; Mendez et al. 2019; Raissi et al. 2019; Dalakoti et al. 2020; Ramezanian et al. 2021; Han et al. 2022; Zhou et al. 2022). While these systems can be characterized by a large number of degrees of freedom, they often exhibit low-rank structures (Maas and Pope 1992; Holmes et al. 1997; Pope 2013; Yang et al. 2013; Mendez et al. 2018). Describing the evolution of those structures provides a powerful modeling approach with substantial reduction to the number of partial differential equations (PDEs) solved in computational simulations (Sutherland and Parente 2009; Biglari and Sutherland 2015; Echekki and Mirgolbabaei 2015; Owoyele and Echekki 2017; Malik et al. 2018, 2020).

Reacting flow simulations can profit from model reduction due to initially high state-space dimensionality stemming from large chemical mechanisms. Reacting systems can often be effectively re-parameterized with much fewer variables. Numerous physics-based parameterization techniques can be found in the combustion literature (Maas and Pope 1992; Van Oijen and De Goey 2002; Jha and Groth 2012; Gicquel et al. 2000). An alternative to the physics-motivated parameterization is a data-driven approach, where low-dimensional manifolds (LDMs) are constructed directly from the training data (Sutherland and Parente 2009; Yang et al. 2013). In particular, dimensionality reduction techniques can be used to define LDMs in the original thermo-chemical state-space. Among many available linear and nonlinear techniques, principal component analysis (PCA) (Jolliffe 2002) is commonly used in combustion to obtain a linear mapping between the original variables and the LDM (Sutherland and Parente 2009; Mirgolbabaei and Echekki 2013; Echekki and Mirgolbabaei 2015; Isaac et al. 2015; Biglari and Sutherland 2015). In PCA, the new parameterizing variables, called principal components (PCs), can be obtained by projecting the training data onto a newly identified basis. A small number of the first few PCs defines the LDM. ROMs can then be built based on this new parameterization. As one example of ROM, PDEs describing the first few PCs can be evolved in combustion simulations (Sutherland and Parente 2009) which result in a substantial reduction of computational costs as compared to transporting the original state variables.

Often, ROM workflows incorporate nonlinear regression to bypass the reconstruction errors associated with an inverse basis transformation. Regression can thus provide an effective route back from the reduced space to the original state-space where the thermo-chemical quantities of interest such as temperature, pressure and

composition, can be retrieved. Regression models can also provide closure for any non-conserved manifold parameters. Nonlinear regression techniques such as artificial neural network (ANN) (Mirgolbabaei and Echekki 2014; Dalakoti et al. 2020), multivariate adaptive regression splines (MARS) (Biglari and Sutherland 2015) or Gaussian process regression (GPR) (Isaac et al. 2015; Malik et al. 2018, 2020) were used in the past in the context of ROM.

In this chapter, we present the complete ROM workflow for application in reacting flow simulations. We begin with a concise mathematical description of a general multicomponent reacting flow. Understanding the governing equations of the analyzed system is a crucial starting point for applying data science tools on the resulting thermo-chemical state vector. After a discussion of training datasets, we present the derivation of the ROM in the context of reacting flows. We review the combination of dimensionality reduction techniques with nonlinear regression. We discuss three popular choices for nonlinear regression: ANNs, GPR and kernel regression. Finally, we review recent results from *a priori* and *a posteriori* ROM of challenging combustion simulations.

Throughout this chapter, we delineate a few outstanding challenges that remain in ROM of combustion processes. For instance, projecting the data onto a lower-dimensional basis, as is done in many ROMs, can introduce undesired behaviors on LDMs. Observations that are distant in the original space can be collapsed into a single, overlapping region. In the overlapping region, those observations are indistinguishable and the projection can become multi-valued. When the identified manifold is used as regressor, these topological behaviors on LDMs can make the regression process more difficult. Ideally, we would like to search for such parameters defining the LDM, that the resulting regression function uniquely represents all dependent variables. Recent work by Zhang et al. (2020) has demonstrated that regressing variables that have significant spatial gradients can be challenging using ANN. Steep gradients can be particularly associated with minor species whose non-zero mass fractions can be located on small portions of the manifold. Problems with ANN reconstruction of minor species on a PCA-derived manifold have recently been reported by Dalakoti et al. (2020). Nevertheless, the attempts to link the poor regression performance with the manifold topology are still scarce in the existing literature, with only a few studies emerging recently (Malik et al. 2022a; Perry et al. 2022; Zdybał et al. 2022c). We show examples of quantitative measures to assess the quality of LDMs that can help bridge this gap. We argue that the future research efforts should focus on advancing strategies that improve regression on manifolds. This should allow to better leverage the capability of techniques such as ANNs or GPR to approximate even highly nonlinear relationships between variables (Hornik et al. 1989).

`PCAfold` examples

The present chapter includes illustrative examples using **`PCAfold`** (Zdybał et al. 2020), a Python software package for generating, analyzing and improving LDMs. It incorporates the entire ROM workflow from data preprocessing, through dimensionality reduction to novel tools for assessing the quality of LDMs. **`PCAfold`** is composed of three main modules: `preprocess`, `reduction` and `analysis`. In brief, the `preprocess` module allows for data preprocessing such as centering and scaling, sampling, clustering and outlier removal. The `reduction` module introduces dimensionality reduction using PCA. The available variants are global and local PCA, subset PCA and PCA on sampled datasets. Finally, the `analysis` module combines functionalities for assessing LDM quality and nonlinear regression results. Each module is accompanied by plotting functions that allow for efficient viewing of results. For instructions on installing the software and for further illustrative tutorials, the reader is referred to the documentation: `https://pcafold.readthedocs.io/`. In the **`PCAfold`** examples that follow, we present a complete workflow that can be adopted for a combustion dataset, using all three modules in series: `preprocess` → `reduction` → `analysis`. We begin by importing the three modules:

```
from PCAfold import preprocess
from PCAfold import reduction
from PCAfold import analysis
```

2 Governing Equations for Multicomponent Mixtures

In this section, we begin with the description of the governing equations for low-Mach multicomponent mixtures, whose solution is the starting point for obtaining training datasets for ROMs in reacting flow applications. In the discussion that follows, $\nabla \cdot \boldsymbol{\phi}$ denotes the divergence of a vector quantity $\boldsymbol{\phi}$, $\nabla \boldsymbol{\phi}$ (or $\nabla \phi$) denotes the gradient of a vector quantity $\boldsymbol{\phi}$ (or a scalar quantity ϕ) and the: symbol denotes tensor contraction. The material derivative is defined as $D/Dt := \partial/\partial t + \mathbf{v} \cdot \nabla$. We let $\mathbf{v}$ be the mass-averaged convective velocity of the mixture, defined as

$$\mathbf{v} := \sum_{i=1}^{n} Y_i \mathbf{u}_i \,, \tag{1}$$

where Y_i is the mass fraction of species i, $\mathbf{u}_i$ is the velocity of species i and n is the number of species in the mixture. At a given point in space and time, transport of physical quantities in a multicomponent mixture can be described by the following set of governing equations written in the conservative (strong) form:

- Continuity equation:

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot \rho \mathbf{v}\,, \tag{2}$$

where ρ is the mixture density.

- Species mass conservation equation:

$$\frac{\partial \rho Y_i}{\partial t} = -\nabla \cdot \rho Y_i \mathbf{v} - \nabla \cdot \mathbf{j}_i + \omega_i \qquad \text{for } i = 1, 2, \ldots, n-1\,, \tag{3}$$

where $\mathbf{j}_i$ is the mass diffusive flux of species i relative to the mass-averaged velocity and ω_i is the net mass production rate of species i due to chemical reactions. Note, that summation of Eqs. (3) over all n species yields the continuity equation (Eq. (2)) since $\sum_{i=1}^{n} Y_i = 1$, $\sum_{i=1}^{n} \mathbf{j}_i = 0$ and $\sum_{i=1}^{n} \omega_i = 0$. For this reason, only $n-1$ independent species mass conservation equations are solved. Mass fraction of the nth species can be computed from the constraint $\sum_{i=1}^{n} Y_i = 1$.

- Momentum equation:

$$\frac{\partial \rho \mathbf{v}}{\partial t} = -\nabla \cdot \rho \mathbf{v}\mathbf{v} - \nabla \cdot \boldsymbol{\tau} - \nabla \cdot p\mathbf{I} + \rho \sum_{i=1}^{n} Y_i \mathbf{f}_i\,, \tag{4}$$

where $\boldsymbol{\tau}$ is the viscous momentum flux tensor, p is pressure, $\mathbf{I}$ is the identity tensor and $\mathbf{f}_i$ is the net acceleration from body forces applied on species i.

with one of the following forms of the energy equation:

- Total internal energy equation:

$$\frac{\partial \rho e_0}{\partial t} = -\nabla \cdot \rho e_0 \mathbf{v} - \nabla \cdot \mathbf{q} - \nabla \cdot \boldsymbol{\tau} \cdot \mathbf{v} - \nabla \cdot p\mathbf{v} + \sum_{i=1}^{n} \mathbf{f}_i \cdot \mathbf{n}_i\,, \tag{5}$$

where e_0 is the mixture specific total internal energy, $\mathbf{q}$ is the heat flux and $\mathbf{n}_i := \rho Y_i \mathbf{u}_i$ is the total mass flux of species i.

- Internal energy equation:

$$\frac{\partial \rho e}{\partial t} = -\nabla \cdot \rho e \mathbf{v} - \nabla \cdot \mathbf{q} - \boldsymbol{\tau} : \nabla \mathbf{v} - p \nabla \cdot \mathbf{v} + \sum_{i=1}^{n} \mathbf{f}_i \cdot \mathbf{j}_i\,, \tag{6}$$

where e is the mixture specific internal energy.

- Enthalpy equation:

$$\frac{\partial \rho h}{\partial t} = -\nabla \cdot \rho h \mathbf{v} - \nabla \cdot \mathbf{q} - \boldsymbol{\tau} : \nabla \mathbf{v} + \frac{Dp}{Dt} + \sum_{i=1}^{n} \mathbf{f}_i \cdot \mathbf{j}_i\,, \tag{7}$$

where h is the mixture specific enthalpy.

- Temperature equation:

$$\frac{\partial \rho T}{\partial t} = -\nabla \cdot \rho T \mathbf{v} - \frac{1}{c_p} \nabla \cdot \mathbf{q} + \frac{\alpha T}{c_p} \frac{Dp}{Dt} - \frac{1}{c_p} \boldsymbol{\tau} : \nabla \mathbf{v} + \frac{1}{c_p} \sum_{i=1}^{n} \left(h_i (\nabla \cdot \mathbf{j}_i - \omega_i) + \mathbf{f}_i \cdot \mathbf{j}_i \right), \quad (8)$$

  where T is the temperature, α is the coefficient of thermal expansion of the mixture ($\alpha = 1/T$ for an ideal gas), c_p is the mixture isobaric specific heat capacity and h_i is the enthalpy of species i.

The governing equations can also be re-formulated using a reference velocity different from the mass-averaged velocity used here. A different mixture velocity would not only affect the terms involving $\mathbf{v}$ explicitly, but also an appropriate diffusive flux will have to be formulated.

The set of governing equations is closed by a few additional relations. The first one is an equation of state. For an ideal gas, we have

$$p = \frac{\rho R_u T}{M}, \quad (9)$$

where R_u is the universal gas constant and $M = \left(\sum_{i=1}^{n} Y_i / M_i \right)^{-1}$ is the molar mass of the mixture where M_i is the molar mass of species i. For a chemically reacting flow, we also require a chemical mechanism that relates temperature, T, pressure, p, and composition, $[Y_1, Y_2, \ldots, Y_n]$, to the chemical source terms, ω_i. The heat flux, $\mathbf{q}$, requires modeling as it in general can include all possible means of heat transfer. One encountered model for $\mathbf{q}$ can be written using the standard Fourier term and the term representing heat transfer through molecular diffusion of species:

$$\mathbf{q} = -\lambda \nabla T + \sum_{i=1}^{n} h_i \mathbf{j}_i, \quad (10)$$

where λ is the mixture thermal conductivity. We also require a model for the diffusive fluxes, $\mathbf{j}_i$. Assuming Fick's law as a model for diffusion, we can express the mass diffusive flux as

$$\mathbf{j}_i = -\rho \boldsymbol{\mathcal{D}} \nabla Y_i, \quad (11)$$

where $\boldsymbol{\mathcal{D}}$ is a matrix of Fickian diffusion coefficients that are functions of the binary diffusion coefficients and composition. Finally, we require a model for the viscous momentum flux tensor, $\boldsymbol{\tau}$. Assuming Newtonian fluids, $\boldsymbol{\tau}$ can be expressed as:

$$\boldsymbol{\tau} = -\mu \left(\nabla \mathbf{v} + (\nabla \mathbf{v})^{\top} \right) + \left(\frac{2}{3} \mu - \kappa \right) (\nabla \cdot \mathbf{v}) \mathbf{I}, \quad (12)$$

where μ is the mixture viscosity. κ is the mixture dilatational viscosity and $\top$ denotes matrix transpose. The reader is referred to numerous great resources for a deeper

discussion of multicomponent mass transfer or derivation of the equations above (Taylor and Krishna 1993; Giovangigli 1999; Bird et al. 2006; Kee et al. 2005).

The governing equations given by Eqs. (2)–(8) can be written in a general matrix form:

$$\frac{\partial \mathbf{X}^{\top}}{\partial t} = -\nabla \cdot \mathbf{C}^{\top} - \nabla \cdot \mathbf{D}^{\top} + \mathbf{S}^{\top}, \tag{13}$$

where $\mathbf{X} \in \mathbb{R}^{N \times Q}$ is the thermo-chemical state vector, $\mathbf{C} \in \mathbb{R}^{d \times N \times Q}$ is the convective flux vector, $\mathbf{D} \in \mathbb{R}^{d \times N \times Q}$ is the diffusive flux vector and $\mathbf{S} \in \mathbb{R}^{N \times Q}$ is the source terms vector. Here, Q is the number of transported properties, d is the number of spatial dimensions of the problem and N is the number of observations. The observations can for instance be linked to measurements on a spatio-temporal grid of a discretized domain. Typically, $N \gg Q$, but the magnitude of Q strongly depends on the number of species in the mixture. In combustion problems, Q can easily reach the order of hundreds when large chemical mechanisms are used (Lu and Law 2009). The appropriate formulation of $\mathbf{X}$, $\mathbf{C}$, $\mathbf{D}$ and $\mathbf{S}$ will depend on a given problem and the assumed simplifications to the governing equations. In the most general case, when all transport equations are solved and no further simplifications are made to the governing equations as given by Eqs. (2)–(8), we form the columns of $\mathbf{X}$, $\mathbf{C}$, $\mathbf{D}$ and $\mathbf{S}$ as per Table 1. Note, that the order of columns in $\mathbf{X}$ does not matter, as long as the corresponding column in $\mathbf{C}$, $\mathbf{D}$ and $\mathbf{S}$ carries an appropriate term. Since the thermo-chemical state of a single-phase multicomponent system is defined by $Q = n + 1$

Table 1 Formulation of the thermo-chemical state vector, $\mathbf{X}$, the convective flux vector, $\mathbf{C}$, the diffusive flux vector, $\mathbf{D}$, and the source terms vector, $\mathbf{S}$, in the most general case, where no further assumptions are imposed to the strong form of the governing equations given by Eqs. (2)–(8)

Equation	State vector (Columns of $\mathbf{X}$)	Convective flux vector (Columns of $\mathbf{C}$)	Diffusive flux vector (Columns of $\mathbf{D}$)	Source terms vector (Columns of $\mathbf{S}$)
Continuity	ρ	$\rho\mathbf{v}$	0	0
Species mass	ρY_i	$\rho Y_i \mathbf{v}$	$\mathbf{j}_i$	ω_i
Momentum	$\rho\mathbf{v}$	$\rho\mathbf{v}\mathbf{v}$	$\boldsymbol{\tau} + p\mathbf{I}$	$\rho \sum_{i=1}^{n} Y_i \mathbf{f}_i$
Total internal energy	ρe_0	$\rho e_0 \mathbf{v}$	$\mathbf{q} + \boldsymbol{\tau} \cdot \mathbf{v} + p\mathbf{v}$	$\sum_{i=1}^{n} \mathbf{f}_i \cdot \mathbf{n}_i$
Internal energy	ρe	$\rho e \mathbf{v}$	$\mathbf{q}$	$-\boldsymbol{\tau} : \nabla\mathbf{v} - p\nabla \cdot \mathbf{v} + \sum_{i=1}^{n} \mathbf{f}_i \cdot \mathbf{j}_i$
Enthalpy	ρh	$\rho h \mathbf{v}$	$\mathbf{q}$	$-\boldsymbol{\tau} : \nabla\mathbf{v} + \frac{Dp}{Dt} + \sum_{i=1}^{n} \mathbf{f}_i \cdot \mathbf{j}_i$
Temperature	ρT	$\rho T \mathbf{v}$	0	$-\frac{1}{c_p}\nabla \cdot \mathbf{q} + \frac{\alpha T}{c_p}\frac{Dp}{Dt} - \frac{1}{c_p}\boldsymbol{\tau} : \nabla\mathbf{v} + + \frac{1}{c_p}\sum_{i=1}^{n}\left(h_i(\nabla \cdot \mathbf{j}_i - \omega_i) + \mathbf{f}_i \cdot \mathbf{j}_i\right)$

variables, an example state vector that follows from the conservative form of the governing equations can be: $\mathbf{X} = [\rho, \rho e, \rho Y_1, \rho Y_2, \ldots, \rho Y_{n-1}]$ (the conserved state vector). For the reasons explained earlier, we only include $n - 1$ independent species mass fractions. Mass fraction of the most abundant species is most often removed (Niemeyer et al. 2017). Historically, specific momentum quantity ($\rho\mathbf{v}$) has not been included in the state vector in ROM of reacting flows (Sutherland and Parente 2009). Various other definitions of the state vector, $\mathbf{X}$, can be adopted with the caveat that the system given by Eq. (13) should not be over-specified (Giovangigli 1999; Hansen and Sutherland 2018). In the next section, we review several strategies to obtain data matrices $\mathbf{X}$, $\mathbf{C}$, $\mathbf{D}$ and $\mathbf{S}$.

3 Obtaining Data Matrices for Data-Driven Approaches

High-dimensional datasets, that are typical to reacting flow applications, can come from numerical simulations or experiments. A few types of numerical datasets of varying complexity often used in the context of ROM are presented in Fig. 1. In particular, solving the governing equations presented in Sect. 2 for simple reacting systems is one computational strategy to obtain training data for ROM. Those simple systems can include zero-dimensional reactors, strained laminar flamelets (Peters 1988), one-dimensional flames or one-dimensional turbulence (ODT) (Kerstein 1999; Sutherland et al. 2010; Echekki et al. 2011). With sufficient amount of assumptions made to the governing equations, we can obtain those datasets at a relatively cheap computational cost. Relaxing some of those assumptions, on the other hand, can move us along the axis of an increasing complexity of the training data, incorporating more information about the turbulence-chemistry interaction. At the end of the complexity spectrum, we have a full direct numerical simulation (DNS), which results in high-fidelity data with all spatial and temporal scales directly resolved. Resorting to more expensive numerical simulations, such as large eddy simulation (LES) or DNS, might not be necessary for ROM purposes. For instance, ODT datasets have been shown to reproduce the DNS conditional statistics well (Punati et al. 2011; Abboud et al. 2015; Lignell et al. 2015; Punati et al. Oct 2016) and have therefore been frequently used in the context of ROM (Mirgolbabaei and Echekki 2014; Mirgolbabaei et al. 2014; Mirgolbabaei and Echekki 2015; Biglari and Sutherland 2015) since they are computationally cheaper to obtain. For an additional overview of datasets presented in Fig. 1 the reader is referred to (Zdybał et al. 2022a).

As an illustrative example, the governing equations for an adiabatic, incompressible, zero-dimensional reactor simplify to:

$$\frac{\partial T}{\partial t} = -\frac{1}{\rho c_p} \sum_{i=1}^{n} h_i \omega_i, \qquad \frac{\partial Y_i}{\partial t} = \frac{\omega_i}{\rho} \qquad \text{for } i = 1, 2, \ldots, n-1.$$

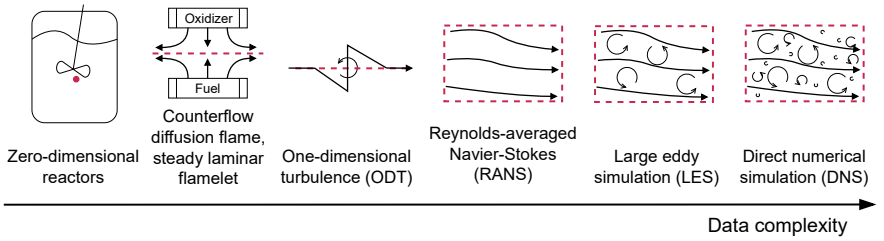

Fig. 1 Schematic overview of training datasets for ROM. As we move along the axis of an increasing complexity, more physical detail is incorporated into the reacting flow simulation

Since a zero-dimensional reactor represents combustion happening in a single point in space, all spatial derivatives present in Eqs. (2)–(8) vanish. Collecting all observations of T and Y_i into a matrix $\mathbf{X}$, and collecting all observations of $-1/\rho c_p \sum_{i=1}^{n} h_i \omega_i$ and ω_i/ρ into a matrix $\mathbf{S}$, we get

$$\mathbf{X} = \begin{bmatrix} \vdots & \vdots & \vdots & & \vdots \\ T & Y_1 & Y_2 & \dots & Y_{n-1} \\ \vdots & \vdots & \vdots & & \vdots \end{bmatrix} \quad \text{and} \quad \mathbf{S} = \begin{bmatrix} \vdots & & \vdots & \vdots & & \vdots \\ -\frac{1}{\rho c_p} \sum_{i=1}^{n} h_i \omega_i & & \frac{\omega_1}{\rho} & \frac{\omega_2}{\rho} & \dots & \frac{\omega_{n-1}}{\rho} \\ \vdots & & \vdots & \vdots & & \vdots \end{bmatrix}.$$

Note, that even though we have removed the transport equation for the nth species, the temperature equation still couples all species through the $-\sum_{i=1}^{n} h_i \omega_i$ term, which represents the heat release rate.

4 Reduced-Order Modeling

At this point, we have learned how to construct training datasets which are the starting point for applying data-driven approaches. It has been a frequent trend in recent years to apply dimensionality reduction techniques to combustion datasets, both for ROM and for data analysis. In the context of combustion, techniques such as PCA (Sutherland and Parente 2009), local PCA (Parente et al. 2009, 2011), kernel PCA (Mirgolbabaei and Echekki 2014), t-distributed stochastic neighbor embedding (t-SNE) (Fooladgar and Duwig 2018), independent component analysis (ICA) (Gitushi et al. 2022), non-negative matrix factorization (NMF) (Zdybał et al. 2022a) or autencoders (Zhang et al. 2021) have been used. In this chapter, we focus on using dimensionality reduction techniques solely to model reduction. We use the premise that the original dataset, $\mathbf{X}$, of high rank can be efficiently approximated by a matrix of a much lower rank. The data can then be re-parameterized with the new mani-

fold parameters (Sutherland et al. 2007). Dimensionality reduction is often coupled with nonlinear regression to provide a more robust mapping between the manifold parameters and the quantities of interest. In this section, we review ROM strategies for reacting flows that include dimensionality reduction and nonlinear regression.

4.1 Data Preprocessing

The first step towards applying dimensionality reduction is data preprocessing. The most straightforward way is data normalization (centering and scaling), which allows to equalize the importance of physical variables of different numerical ranges. Any variable ϕ in a dataset can be centered and scaled using the general formula $\widetilde{\phi} = (\phi - c)/d$, where c is the center computed as the mean value of ϕ and d is the scaling factor. Other data preprocessing means can include data sampling to tackle imbalance in sample densities, data subsetting (feature selection), or outlier removal. The effect of data preprocessing, including scaling and outlier removal, on the resulting LDMs was studied in (Parente and Sutherland 2013). In the discussion that follows, we assume that the training datasets have been appropriately preprocessed.

4.2 Reducing the Number of Governing Equations

Data-driven model reduction has emerged in recent years with applications to complex dynamical systems. Model reduction of complex systems typically starts with changing the basis to represent the original high-dimensional system. Let $\mathbf{A} \in \mathbb{R}^{Q \times Q}$ be the matrix of modes defining the new basis. The matrix $\mathbf{A}$ can be found directly from the training data using a dimensionality reduction technique, such as PCA. As long as $\mathbf{A}$ is constant in space and time, the governing equations of the form presented in Eq. (13) can be written as:

$$\frac{\partial \mathbf{A} \cdot \mathbf{X}^{\top}}{\partial t} = -\nabla \cdot \mathbf{A} \cdot \mathbf{C}^{\top} - \nabla \cdot \mathbf{A} \cdot \mathbf{D}^{\top} + \mathbf{A} \cdot \mathbf{S}^{\top}, \tag{14}$$

where $\mathbf{X}$ can in general contain all state variables as presented in Sect. 2, or a subset of those. Equation (14) represents transformation of the original governing equations to the new basis defined by $\mathbf{A}$.

4.2.1 Principal Component Transport

PCA is one dimensionality reduction technique that can be used to obtain the basis matrix $\mathbf{A}$ by performing eigendecomposition of the data covariance matrix. PCA can provide optimal reaction variables, PCs, that are linear combinations of the original

thermo-chemical state variables (Sutherland 2004; Sutherland and Parente 2009; Parente et al. 2009). We can define the matrix of PCs, $\mathbf{Z} \in \mathbb{R}^{N \times Q}$, as $\mathbf{Z} = \mathbf{XA}$, which represents the transformation of $\mathbf{X}$ to the new PCA-basis. The governing equations written in the form of Eq. (13) can be linearly transformed to this new PCA-basis as per Eq. (14). This yields a new set of transport equations for the PCs:

$$\frac{\partial \mathbf{Z}^\top}{\partial t} = -\nabla \cdot \mathbf{C_Z}^\top - \nabla \cdot \mathbf{D_Z}^\top + \mathbf{S_Z}^\top , \tag{15}$$

where $\mathbf{C_Z} = \mathbf{CA}$ are the projected convective fluxes, $\mathbf{D_Z} = \mathbf{DA}$ are the projected diffusive fluxes and $\mathbf{S_Z} = \mathbf{SA}$ are the PC source terms – the source terms of the original state-space variables transformed to the new PCA-basis. We will further refer to the jth PC (the jth column of $\mathbf{Z}$) as Z_j and to the jth PC source term (the jth column of $\mathbf{S_Z}$) as $S_{Z,j}$. By solving the transport equations for the first q PCs only, we can significantly reduce the number of PDEs in Eq. (15) as compared to Eq. (13). PCA further guarantees that the q first PCs are the most important ones in terms of the variance retained in the data. From the Eckart-Young theorem (Eckart and Young 1936), we know that approximating the dataset $\mathbf{X}$ with only q first PCs gives the closest rank-q approximation to $\mathbf{X}$. This approximation can be obtained through an inverse basis transformation: $\mathbf{X} \approx \mathbf{Z_q}{\mathbf{A_q}}^{-1}$, where the subscript $\mathbf{q}$ denotes truncation to q components. With the PCA modeling approach, the first q PCs become the reaction variables that re-parameterize the original thermo-chemical state-space. They also define the q-dimensional manifold, embedded in the originally Q-dimensional state space.

Formulation of PC-transport was first proposed by Sutherland and Parente (2009). Since then, numerous *a priori* (Biglari and Sutherland 2012; Mirgolbabaei and Echekki 2013; Mirgolbabaei et al. 2014; Malik et al. 2018; Ranade and Echekki 2019; Dalakoti et al. 2020; D'Alessio G et al. 2022; Zdybał et al. 2022c) and *a posteriori* (Isaac et al. 2014; Biglari and Sutherland 2015; Echekki and Mirgolbabaei 2015; Coussement et al. 2016; Owoyele and Echekki 2017; Ranade and Echekki 2019; Malik et al. 2020, 2022a, b) studies have been conducted. The advantage of PCA-based modeling is that models can be trained on datasets coming from simpler systems that are cheap to compute (such as zero-dimensional reactors or laminar flamelets, see Sect. 3). This has been shown to be a feasible modeling strategy (Malik et al. 2018, 2020), as long as the training data covers the possible states of the reacting system that might be accessed during simulation of real systems.

There are a few additional ingredients of the PC-transport modeling approach. First, since Eq. (15) is solved for the PCs which do not have any physical relevance, we require a mapping back to the original thermo-chemical state-space, where physical quantities of interest can be retrieved. Second, we need to parameterize the source terms, $\mathbf{S_Z}$, of any non-conserved manifold parameters (Sutherland 2004; Sutherland and Parente 2009). While in the original state space we have known relations between the transported variables and their source terms, we lack such explicit relations in the space of PCs. Both these points can be handled by coupling nonlinear regression with the PC-transport model—this will be further discussed in Sect. 4.4. Finally, in the

presence of diffusion, diffusive fluxes need to be represented in the new PCA-basis as well. Treatment of PC diffusive fluxes was proposed by Mirgolbabaei and Echekki (2014) and by Biglari and Sutherland (2015). A study by Echekki and Mirgolbabaei (2015) further looked into mitigating the multicomponent effects associated with diffusion of PCs. Another study by Coussement et al. (2016) looked at the influence of differential diffusion on PCA-based models. The work done in (Coussement et al. 2016) looked at how rotation of the PCs can diagonalize the PCs diffusion coefficients matrix and thus make the treatment of diffusion of PCs easier.

Computing the PCs and the PC source terms

In this example, we demonstrate how one can obtain the PCs and the PC source terms from the state vector, **X**, and the source terms vector, **S**, respectively. We use a syngas/air steady laminar flamelet dataset and generate its two-dimensional (2D) projection onto the PCA-basis. The dataset was generated using **`Spitfire`** Python library (Hansen et al. 2022) and the chemical mechanism by Hawkes et al. (2007).

Load the dataset, removing the nth species, N_2:

```
import numpy as np
X = np.genfromtxt('syngas-air-SLF-state-space.csv', delimiter=',')
     [:,0:-1]
S = np.genfromtxt('syngas-air-SLF-state-space-sources.csv', delimiter
     =',')[:,0:-1]
f = np.genfromtxt('syngas-air-SLF-mixture-fraction.csv', delimiter
     =',')
chi = np.genfromtxt('syngas-air-SLF-dissipation-rates.csv', delimiter
     =',')
(n_observations, n_variables) = X.shape
```

Perform PCA on the dataset:

```
pca = reduction.PCA(X, scaling='auto', n_components=2)
```

Transform the state vector, **X**, to the new PCA basis:

```
Z = pca.transform(X)
```

Transform the source terms vector, **S**, to the new PCA basis (note the `nocenter=True` flag):

```
S_Z = pca.transform(S, nocenter=True)
```

Visualize the 2D projection of the dataset, colored by the two PC source terms, $S_{Z,1}$ and $S_{Z,2}$ (Fig. 2):

```
plt = reduction.plot_2d_manifold(Z[:,0], Z[:,1],
                                 color=S_Z[:,0],
                                 s=15,
                                 x_label='$Z_{1}$ [$-$]', y_label='
     $Z_{2}$ [$-$]',
                                 colorbar_label='$S_{Z, 1}$\n[$-$]',
                                 color_map='inferno',
                                 grid_on=True,
                                 figure_size=(6,4))
```

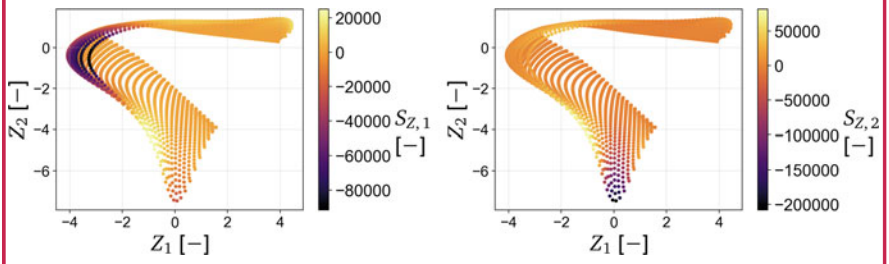

Fig. 2 Outputs of `analysis.plot_2d_manifold`

It is visible from the plot above that this 2D projection introduces significant non-uniqueness that particularly affects the dependent variable $S_{Z,1}$. At the same time, this visible overlap in the (Z_1, Z_2) space does not coincide with the region of the largest variation in the second PC source term, $S_{Z,2}$, values. We can expect that $S_{Z,1}$ will be much more strongly affected by the manifold non-uniqueness than $S_{Z,2}$.

4.3 Low-Dimensional Manifold Topology

Apart from PCA, numerous manifold learning methods can help identify LDMs in high-dimensional combustion datasets. Although the approach presented in Sect. 4.2.1 allows for substantial model reduction, several manifold challenges need to be addressed. In particular, during projection of data to a lower-dimensional basis, non-uniqueness can be introduced in the manifold topology which can hinder successful model definition. A good model should provide unique definition of all relevant dependent variables as functions of the manifold parameters (Sutherland 2004; Pope 2013). With this premise, the future research directions can be twofold. First, we require techniques to characterize the quality of LDMs. Second, we should seek strategies that provide an improved manifold topology. Both points should feed one another and can be tackled simultaneously.

Measures such as the coefficient of determination (Biglari and Sutherland 2012) or manifold nonlinearity (Isaac et al. 2014) have been used in the past to assess manifold parameterizations *a priori*. A recently proposed normalized variance derivative metric (Armstrong and Sutherland 2021) is much more informative in comparison. It can characterize manifold quality with respect to two important aspects: feature sizes and multiple scales of variation in the dependent variable space. Multiple scales of variation can often indicate non-uniqueness in manifold parameterization. A more compact metric based on the normalized variance derivative has also been proposed recently (Zdybał et al. 2022b). It reduces the manifold topology to a single number and can be used as a cost function in manifold optimization tasks.

Some topological challenges can be mitigated through appropriate data preprocessing prior to projecting to a lower-dimensional space. The most straightforward strategy is data scaling, with Pareto (Noda 2008) or VAST (Hector et al. 2003) scalings most commonly used (Biglari and Sutherland 2015; Isaac et al. 2015; Malik et al. 2018, 2020). Other authors have tackled manifold challenges by training combustion models on only a subset of the original thermo-chemical state-space variables (Chatzopoulos and Rigopoulos 2013; Mirgolbabaei and Echekki 2013, 2014; Echekki and Mirgolbabaei 2015; Isaac et al. 2015; Owoyele and Echekki 2017; Malik et al. 2020; Nguyen et al. 2021; Gitushi et al. 2022). Recent work developed a strategy for a manifold-informed state vector subset selection (Zdybał et al. 2022b). A study done by Coussement et al. (2012) suggests that tackling initial imbalance in data density can yield a more accurate low-dimensional representation of the flame region.

Another important decision that needs to be made at the modeling stage is what manifold dimensionality, q, should we select? Additional number of parameters may be required for more complex manifold topologies. While techniques such as PCA provide orthogonal manifold parameters (PCs), each bringing information about variance in another orthogonal data dimension, it is not clear how many PCs is sufficient to provide a good quality, regressible manifold topology. From the computational cost point of view, keeping low manifold dimensionality is desired. However, keeping q small should not be at the expense of the parameterization quality. Admittedly, more work is required to provide answers to those questions.

Low-dimensional manifold assessment

Below, we demonstrate how we can assess the quality of LDMs obtained from PCA using the novel normalized variance derivative metric (Armstrong and Sutherland 2021). We will assess the generated 2D projections and we take the two PC source terms as the two dependent variables.

Define the bandwidth values, σ:

```
bandwidth_values = np.logspace(-5, 1, 100)
```

Specify the names of the dependent variables:

```
variable_names=['$S_{Z,1}$', '$S_{Z,2}$']
```

Compute the normalized variance derivative, $\hat{\mathcal{D}}(\sigma)$:

```
variance_data = analysis.compute_normalized_variance(Z, S_Z,
    variable_names,
                                                     bandwidth_values
    =bandwidth_values)
```

Plot the $\hat{\mathcal{D}}(\sigma)$ curves for the two PC source terms (Fig. 3):

```
analysis.plot_normalized_variance_derivative(variance_data,
                                             color_map='Greys',
                                                   figure_size=(10,2.5)
    )
```

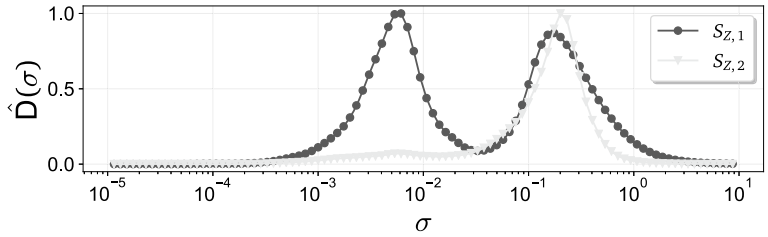

Fig. 3 Output of `analysis.plot_normalized_variance_derivative`

The normalized variance derivative, $\hat{\mathcal{D}}(\sigma)$, quantifies the information content on a manifold at various length scales specified by the bandwidth, σ. The peaks in the $\hat{\mathcal{D}}(\sigma)$ profile happening at very small length scales can often be linked to non-uniqueness in manifold topologies. In the plot above, we can observe two distinct peaks corresponding to the $\hat{\mathcal{D}}(\sigma)$ curve for the first PC source term, $S_{Z,1}$. The peak happening for smaller σ can be understood from our visualization of the manifold topology in Fig. 2. In our visualization we have seen clear overlap, where the observations corresponding to highly negative values of $S_{Z,1}$ were projected directly above observations corresponding to $S_{Z,1} \approx 0$. The information provided by $\hat{\mathcal{D}}(\sigma)$ is valuable at the modeling stage, as it allows to quantitatively assess the quality of low-dimensional data projections.

4.4 Nonlinear Regression

Nonlinear regression is often used to provide an effective mapping between the manifold parameters and the dependent variables of interest (Biglari and Sutherland 2015; Mirgolbabaei and Echekki 2015; Malik et al. 2018; Dalakoti et al. 2020). The set of dependent variables, $\boldsymbol{\phi}$, typically include the PC source terms, $\mathbf{S_Z}$, and the thermochemical state-space variables, such as temperature, density and composition. Unlike the inverse basis transformation discussed in Sect. 4.2.1, regression has the potential to yield much more accurate dependent variable reconstructions (Mirgolbabaei and Echekki 2015). Nonlinear regression techniques allow us to encode nonlinear relationships between the manifold parameters and the dependent variables. This characteristic is especially desired for modeling source terms, which are highly nonlinear functions of the independent variables. In the past research, reconstruction of the PC source terms has been shown to be much more challenging than reconstruction of the state variables (Biglari and Sutherland 2012, 2015). This is due to the fact that the state-space variables evolve nonlinearly according to the Arrhenius relations.

In this section, we are concerned with a set of n_ϕ dependent variables defined as $\boldsymbol{\phi} = [\mathbf{S_Z}, T, \rho, \mathbf{Y}_i]$, where $\mathbf{Y}_i$ is a vector of $n-1$ species mass fractions, $\mathbf{Y}_i = [Y_1, Y_2, \ldots, Y_{n-1}]$. In mathematical terms, the goal of nonlinear regression is to find a function $\mathscr{F}$, such that:

$$\phi \approx \mathscr{F}(\mathbf{Z_q}), \tag{16}$$

where ϕ is a dependent variable and $\mathbf{Z_q}$ are the q first PCs. It is worth noting that some regression techniques allow to obtain all dependent variables at once; other require regressing dependent variables one-by-one. Three popular nonlinear regression techniques are reviewed in this section. Our main focus is in presenting how the function $\mathscr{F}$ is defined in each technique.

Nonlinear regression

In the examples that follow, we will perform and assess ANN, GPR and kernel regression of the two PC source terms defined earlier. The nonlinear regression models will be trained on 80% and tested on the remaining 20% of the data. Below, we use the sampling functionalities to randomly sample train and test data:

```
sample_random = preprocess.DataSampler(np.zeros((n_observations,)).
    astype(int),
                                       random_seed=100,
                                       verbose=True)
(idx_train, idx_test) = sample_random.random(80)
Z_train = Z[idx_train,:]; Z_test = Z[idx_test,:]
S_Z_train = S_Z[idx_train,:]; S_Z_test = S_Z[idx_test,:]
```

4.4.1 Artificial Neural Network

Artificial neural networks (ANNs) are a network of connected layers that compute the output(s) based on some convolution of the layer's input(s) (Russell and Norvig 2002). The layer's inputs and outputs are called neurons. ANNs form a parametric technique that can be used both for regression and classification and are broadly used in the context of ROM. This applies to both reacting (Mirgolbabaei and Echekki 2013, 2014, 2015; Echekki and Mirgolbabaei 2015; Ranade and Echekki 2019; Dalakoti et al. 2020; Zhang et al. 2020) and non-reacting (pure fluid) applications (Farooq et al. 2021).

For an architecture with a single neural layer (input $\rightarrow$ output), the regression function $\mathscr{F}$ at some query point P can be written as:

$$\mathscr{F}\big|_P = g_1(\mathbf{Z_q}\big|_P \mathbf{W}_1 + \mathbf{b}_1), \tag{17}$$

where $\mathbf{W}_1 \in \mathbb{R}^{q \times n_\phi}$ is the matrix of weights and $\mathbf{b}_1 \in \mathbb{R}^{1 \times n_\phi}$ is the vector of biases, and g_1 is the activation function. Both $\mathbf{W}_1$ and $\mathbf{b}_1$ are learned from the training data by

solving an optimization problem. For a deep neural network (DNN) which allow for multi-layer architecture, the regression function becomes a composition of functions of the form shown in Eq. (17). Assuming m neural layers, we can write that

$$\mathscr{F}\big|_P = g_m(g_{m-1}(\cdots g_2(g_1(\mathbf{Z_q}\big|_P\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2)\cdots\mathbf{W}_{m-1} + \mathbf{b}_{m-1})\mathbf{W}_m + \mathbf{b}_m)\,, \tag{18}$$

where all matrices $\mathbf{W}_l$ as well as all vectors $\mathbf{b}_l$ for layers $l = 1, 2, \ldots, m$, do not need to be of the same size, since the number of neurons can vary in different layers. Also the activation functions g_l can vary for different layers. The Eq. (18) essentially states in matrix notation that the output of one layer becomes an input of the following layer.

The advantage of using ANN regression is that predictions are relatively cheap to compute once the ANN model has been trained. As can be seen from Eqs. (17)–(18), predicting a single observation of $\boldsymbol{\phi}$ given a set of query inputs, $\mathbf{Z_q}\big|_P$, requires vector-matrix multiplication(s), where $\mathbf{W}_l$ is typically a small matrix. This makes ANNs very appealing from the computational cost point of view. However, the optimization used to determine weights and biases is prone to reaching local minimum. The best one can hope for is that the local minimum will result in reasonable predictions. The overall performance of the trained network is dependent on many factors that the user can tune, such as the architecture or the choice of the activation function(s). The ANN predictions are also dependent on the random initial guess for the weights and biases which can greatly affect gradient descent -based algorithms. To improve the network performance, Bayesian optimization can be used to determine the ANN hyper-parameters (Mockus 2012; Bergstra et al. 2013; Barzegari and Geris 2021).

ANN regression

In this example, we create an ANN model to obtain the parameterizing function, $\mathscr{F}$. We will use a popular Python library for ANN, **`Keras`** (Chollet et al 2015), which is a backend of the **`TensorFlow`** software (Abadi et al. 2015). Below, we import the necessary libraries:

```
from keras.models import Sequential
from keras.layers import Dense
from keras import optimizers
from keras import losses
```

We use a relatively simple architecture with two hidden layers with five neurons each:

```
model = Sequential([
Dense(5, input_dim=2, activation='sigmoid'),
Dense(5, activation='sigmoid'),
Dense(2, activation='linear')])
```

Normalize the ANN outputs to the $\langle -1; 1\rangle$ range:

```
(normalized_S_Z, centers, scales) = preprocess.center_scale(S_Z, '-1
    to1')
```

Sample the normalized train data outputs:

```
normalized_S_Z_train = normalized_S_Z[idx_train,:]
```

Compile the ANN model with the given architecture :

```
model.compile(optimizers.Adam(lr=0.001),
              loss=losses.mean_squared_error,
              metrics=['mse'])
```

Fit the compiled ANN model with the training data, specifying the hyper-parameters:

```
history = model.fit(Z_train,
                    normalized_S_Z_train,
                    batch_size=100, epochs=500,
                    validation_split=0.2, verbose=0)
```

Finally, we predict the two PC source terms, remembering to invert the $\langle -1; 1 \rangle$ normalization applied initially:

```
S_Z_ANN_predicted = model.predict(Z)
S_Z_ANN_predicted = preprocess.invert_center_scale(S_Z_ANN_predicted,
                                                   centers, scales)
```

We can visualize the regression result in 3D (Fig. 4):

```
analysis.plot_3d_regression(Z[:,0],
                            Z[:,1],
                            S_Z[:,0],
                            S_Z_ANN_predicted[:,0],
                            elev=30,
                            azim=200,
                            x_label='$Z_1$ [$-$]',
                            y_label='$Z_2$ [$-$]',
                            z_label='$S_{Z, 1}$ [$-$]',
                            figure_size=(12,6))
```

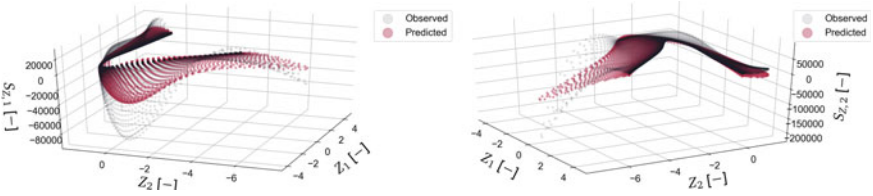

Fig. 4 Outputs of `analysis.plot_3d_regression`

The figure above demonstrates qualitatively how regression can struggle to regress dependent variables on an ill-behaved manifold. We can observe regions with large mismatch between the observed and the predicted values of the two PC source terms. In particular, highly negative values of $S_{Z,1}$ are poorly predicted. This behavior can be linked to our manifold topology assessments in the earlier examples, where we have seen non-uniqueness affecting highly negative values of $S_{Z,1}$.

4.4.2 Gaussian Process Regression

Gaussian process regression (GPR) is a kernel-based, semi-parametric regression technique (Williams and Rasmussen 2006). A powerful characteristic of GPR is that prior knowledge about the functional relationship between the independent and dependent variables can be injected at the modeling stage. For instance, if the system dynamics is known to have an oscillatory behavior, the kernel can be built using a periodic function. Another important feature of GPR is that it provides uncertainty bounds on the predicted variables, while techniques such as ANN or kernel regression only provide predictions.

In GPR, the regression function $\mathscr{F}$ is learned from the data:

$$\mathscr{F}(\mathbf{Z_q}) = \mathscr{GP}(m(\mathbf{Z_q}), \mathbf{K}(\mathbf{Z_q}, \mathbf{Z_q}')) \,, \tag{19}$$

where $\mathscr{GP}$ denotes a Gaussian process, m is the mean function and $\mathbf{K}$ is the covariance matrix. The covariance matrix, $\mathbf{K} \in \mathbb{R}^{n_x \times n_y}$, can be populated using any kernel of choice as long as the elements in $\mathbf{K}$ satisfy $k_{i,j} = k_{j,i}, \forall_{i \neq j}$. Typically, kernels are functions of the distance between data observations, x_i and x_j. Squared exponential kernel is commonly used to populate $\mathbf{K}$:

$$k_{i,j} = h^2 \exp\left(\frac{(x_i - x_j)^2}{\lambda^2}\right), \tag{20}$$

where h is the scaling factor and λ is the bandwidth of the kernel. Figure 5a visualizes the effect of increasing the kernel bandwidth, λ, on the resulting covariance matrix structure. With a larger λ, we are allowing observations that are further apart

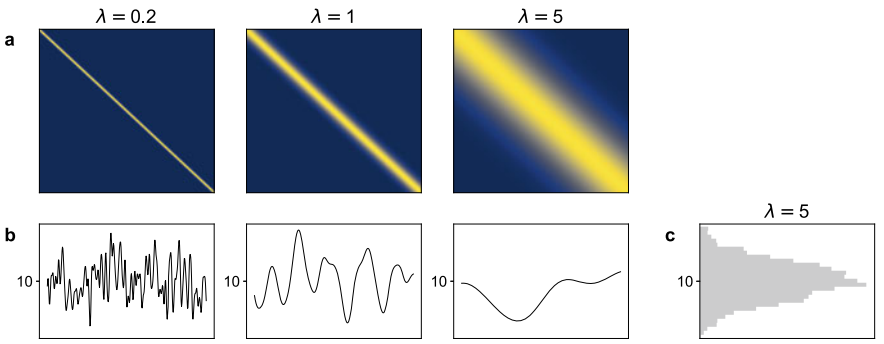

Fig. 5 The effect of kernel bandwidth on smoothing the Gaussian process regression predictions. In this example, the scaling factor $h = 0.1$. **a** Heatmaps of three covariance matrices, $\mathbf{K}$, generated using the squared exponential kernel with an increasing kernel bandwidth, λ. **b** Example regression function realizations resulting from each covariance matrix. **c** Histogram of one hundred function realizations corresponding to the $\lambda = 5$ case with the mean equal to 10. The mean dictates the most probable function value

to correlate. The structure of **K** is then reflected in possible regression function realizations (Fig. 5b). With a very narrow kernel (here $\lambda = 0.2$), the resulting realization looks very noisy—even nearby observations can have very different function values. The larger the kernel bandwidth, the smoother the realization function (Duvenaud 2014). With $\lambda = 5$ we can expect stronger correlation in function values even for observations that are further away. Figure 5c additionally shows a histogram of one hundred regression function realizations resulting from $\lambda = 5$. Since in this example we have chosen the mean equal to 10, the histogram has a Gaussian distribution centered around 10.

GPR regression

In this example, we create a GPR model to obtain the parameterizing function, $\mathscr{F}$. We will use a Python package **`george`** (Ambikasaran et al. 2016) to perform GPR:

```
import george
```

Create the squared exponential kernel:

```
kernel = george.kernels.ExpSquaredKernel(20, ndim=2)
```

Fit the GPR model with the training data:

```
gp = george.GP(kernel)
gp.compute(Z_train, yerr=1.25e-12,)
```

Predict the two PC source terms:

```
S_Z1_GPR_predicted, S_Z1_GPR_var = gp.predict(S_Z_train[:,0], Z,
    return_var=True)
S_Z2_GPR_predicted, S_Z2_GPR_var = gp.predict(S_Z_train[:,1], Z,
    return_var=True)
```

We visualize the predicted PC source terms (Fig. 6):

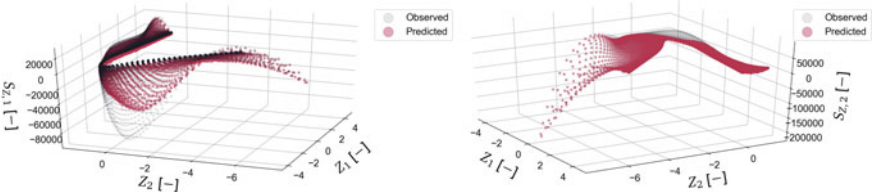

Fig. 6 Outputs of `analysis.plot_3d_regression`

In the plot above, we observe similar misprediction of the first PC source term, $S_{Z,1}$, as we have seen with ANN regression.

4.4.3 Kernel Regression

Kernel regression is a nonparametric technique that does not include the "training" step. Function $\mathscr{F}$ is inferred for each query point, P, directly from the training data samples in some vicinity of P. The regression function $\mathscr{F}$ is built from the Nadaraya-Watson estimator (Härdle 1990) as:

$$\mathscr{F}\big|_P = \frac{\sum_{i=1}^{N} K_{i,P}(\mathbf{Z_q}, \sigma)\phi_i}{\sum_{i=1}^{N} K_{i,P}(\mathbf{Z_q}, \sigma)}, \tag{21}$$

where K is the kernel function and σ is the kernel bandwidth. The Eq. (21) essentially represents a linear combination of the weighted observations of ϕ. Similarly as in GPR, various kernels can be used in place of K. The most popular Gaussian kernel yields:

$$K_{i,P}(\mathbf{Z_q}, \sigma) = \exp\left(\frac{-||\mathbf{Z_q}\big|_i - \mathbf{Z_q}\big|_P||_2^2}{\sigma^2}\right), \tag{22}$$

The larger the kernel bandwidth, σ, the larger the resulting coefficients K_i multiplying each data observation, ϕ_i. In other words, an increasing σ yields a stronger influence of data observations distant from P on the predicted function value at P. An implication of a larger σ on regression means that $\mathscr{F}$ becomes a smoother function – note the similarity of this concept with the covariance matrix discussion in Sect. 4.4.2.

Kernel regression

In this example, we create a kernel regression model to obtain the parameterizing function, $\mathscr{F}$. We specify the kernel bandwidth, σ, for the Nadaraya-Watson estimator:

```
bandwidth = 0.5
```

Fit the kernel regression model with the training data:

```
model = analysis.KReg(Z_train, S_Z_train)
```

Predict the two PC source terms:

```
S_Z_KReg_predicted = model.predict(Z, bandwidth=bandwidth)
```

Similarly as before, we visualize the predicted PC source terms (Fig. 7):

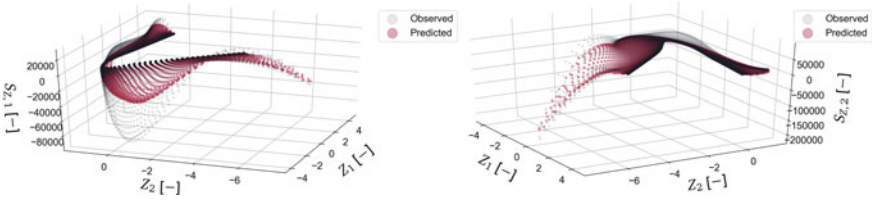

Fig. 7 Outputs of `analysis.plot_3d_regression`

Since kernel regression makes predictions by "smoothing out" function values over some neighborhood of a query point, the non-uniqueness in $S_{Z,1}$ values affected regression performance, similarly to what we have observed with ANN and GPR regression.

Nonlinear regression assessment

Here, we continue the kernel regression example and use various metrics to assess the regression performance. Two common metrics that are available are the coefficient of determination, R^2, and the normalized root mean squared error (NRMSE). For vector quantities, such as the PC source terms vector, another useful metric might be the good direction estimate (GDE) which is a measure derived from cosine similarity.

Compute the regression metrics for the two PC source terms:

```
metrics = analysis.RegressionAssessment(S_Z, S_Z_KReg_predicted,
                                        variable_names=variable_names
    ,
                                        norm='std',
                                        tolerance=0.05)
```

Display the regression metrics in a table format (Fig. 8):

```
metrics.print_metrics(table_format=['pandas'], metrics=['R2', 'NRMSE
    ', 'GDE'])
```

	R2	NRMSE	GDE
$S_{Z,1}$	0.4800	0.7211	45.2449
$S_{Z,2}$	0.8100	0.4358	45.2449

Fig. 8 Output of `analysis.RegressionAssessment print_metrics`.

The `RegressionAssessment` class also allows to compare two regression results. It can color-code the displayed table and mark the metrics that got worse red and those that got better green.

In addition to a single value of each metric for the entire dataset, we can also compute stratified metrics values, in bins (clusters) of a dependent variable. This allows us to observe how regression performed in specific regions of the manifold. Below, we compute the stratified metrics in four bins of the first PC source term, $S_{Z,1}$. We then look at the kernel regression of the first PC source term in each bin.

We first use the function from the `preprocess` module that allows to manually partition the dataset into bins of a selected variable. Compute the bins:

```
(idx, _) = preprocess.predefined_variable_bins(S_Z[:,0],
                                               split_values=[-10000,
    0, 10000],
                                               verbose=False)
```

Those data bins (clusters) are visualized below on the syngas/air flamelet dataset in the space of mixture fraction and temperature (Fig. 9):

```
preprocess.plot_2d_clustering(f, X[:,0], idx,
                              x_label='$f$ [-]', y_label='$T$ [$K$]',
                              first_cluster_index_zero=False,
                              color_map='coolwarm',
                              figure_size=(8,4))
```

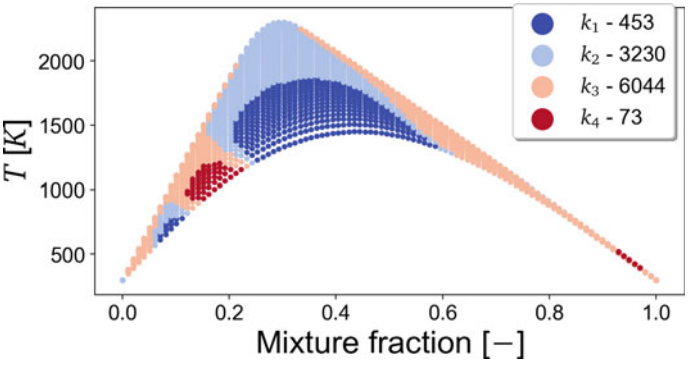

Fig. 9 Output of `preprocess.plot_2d_clustering`

Compute the stratified regression metrics:

```
metrics = analysis.RegressionAssessment(S_Z[:,0], S_Z_KReg_predicted
    [:,0],
                                        idx=idx,
                                        use_global_mean=True,
                                        norm='std',
                                        use_global_norm=True)
```

Display the stratified regression metrics in a table format (Fig. 10):

```
metrics.print_stratified_metrics(table_format=['pandas'], metrics=['
    NRMSE'])
```

	Observations	Min	Max	NRMSE
k1	453	-91,760.3782	-10,016.6359	3.0987
k2	3230	-9,989.6371	-0.0000	0.4287
k3	6044	0.0000	9,965.7716	0.1419
k4	73	10,002.6112	24,987.7263	0.6479

Fig. 10 Output of `analysis.RegressionAssessmentprint_stratified_metrics`.

The stratified metrics let us see that kernel regression performed relatively well for $S_{Z,1} > -10,000$ with NRMSE values less than 1.0 in bins k_2, k_3 and k_4. However, we see that for observations in bin k_1, corresponding to the smallest values of $S_{Z,1}$, the NRMSE is significantly higher. The results of the stratified NRMSE values are consistent with what we have seen in Fig. 7 that visualized the regression result. We have seen a significant departure from the observed and predicted data surface for highly negative values of $S_{Z,1}$. Finally, we note that the stratified regression metrics can be computed in bins obtained using any data clustering technique of choice. A good overview of data clustering algorithms can be found in (Thrun and Stier 2021). Some of those techniques are also implemented in the **`scikit-learn`** Python library (Pedregosa 2011).

5 Applications of the Principal Component Transport in Combustion Simulations

Using large detailed chemical mechanisms inside a numerical simulation can become a tedious task, especially when other complex phenomena are involved, such as turbulence or pollutant formation. Therefore, parameterization of the thermo-chemical state of a reacting system using a reduced set of optimally chosen variables is very appealing. In this context, the use of PCA is well-suited. PCA allows to automatically reduce dimensionality and retain most of the variance of the system. As we have seen in Sect. 4.2.1, substantial reduction in the number of governing equations of the system can be made by transporting only a subset of the PCs in a numerical simulation. In this section, we present recent applications of the PC-transport approach as reported in (Malik et al. 2018, 2020).

5.1 A Priori Validations in a Zero-Dimensional Reactor

We first show the application of the PC-transport approach in the context of zero-dimensional perfectly stirred reactor (PSR) calculations (Malik et al. 2018). The model validation was done *a priori*, meaning that the model training and validation were made using the same PSR configuration. Two different fuels were investigated: methane (CH_4) and propane (C_3H_8). For each fuel, the dataset for PCA was generated with unsteady PSR simulations, varying the residence time in the reactor from extinction to equilibrium. For each residence time inside the reactor, the entire temporal solution from initialization to steady-state was saved. The dataset for PCA generated in this way contained approximately 100,000 observations for each state variable for the methane case, and 420,000 observations for each state variable for

the propane case. In methane simulations, the GRI-3.0 chemical mechanism (Smith et al. 2022) was used, with the nth species, N_2, removed, resulting in 34 species. For the propane case, the `Polimi_1412` chemical mechanism (Humer et al. 2007) was used, containing 162 species. PCA-basis was computed using the species mass fractions alone ($\mathbf{X} = [Y_1, Y_2, \ldots, Y_{n-1}]$). The solution of the PC-transport model (as per Eq. (15)) without coupling with nonlinear regression was first obtained, where the predicted quantities were computed using an inverse PCA-basis transformation. Then, the PC-transport approach was coupled with GPR regression (PCA-GPR) in order to increase the dimensionality reduction potential of PCA. Both PC-transport approaches were compared with the full solution obtained by transporting the original species mass fractions (as per Eq. (3)).

5.1.1 Simulation Results for Methane/Air Combustion

Figure 11 shows the PSR solution for the temperature and the H_2O and OH mass fractions for the methane case. The results are obtained with the PC-transport model without nonlinear regression using $q = 24, q = 25$ and $q = 34$ PCs (Fig. 11a) and the PC-transport coupled with GPR regression using $q = 1$ and $q = 2$ PCs (Fig. 11b). For comparison, full solution solving governing equations for the original state variables is shown with the solid line. Using the PC-transport approach without nonlinear regression, at least $q = 25$ components out of 34 were required to obtain an accurate solution, which correspond to a model reduction of 26%. On the other hand, when the PC-transport model was coupled with GPR regression, the results show remarkable accuracy using only $q = 2$ PCs for the prediction of temperature, and both major and minor species. It can also be seen that the PCA-GPR model with $q = 1$ does not provide sufficient accuracy in the ignition region, under-estimating the ignition delay.

5.1.2 Simulation Results for the Propane/Air Combustion

Figure 12 shows the PSR solution for the temperature, and the CO_2 and O_2 mass fractions for the propane case. With the PC-transport model without regression (Fig. 12a), at least $q = 142$ components out of 162 are required in order to get an accurate description, representing a model reduction of 12%. By combining the PC-transport model with the potential offered by nonlinear regression (PCA-GPR), the number required components can be reduced down to $q = 2$. Although the reduced model performs well overall, some deviation from the full solution was observed in the ignition/extinction region. The PCA-GPR model was then further improved, by dividing the PCA manifold into two clusters and performing GPR regression locally in each cluster (PCA-L-GPR). By doing so, the level of accuracy of the model is significantly improved, leading to an almost perfect match with only $q = 2$ components instead of 162 (reduction of 98%). This improvement can be observed in Fig. 12b.

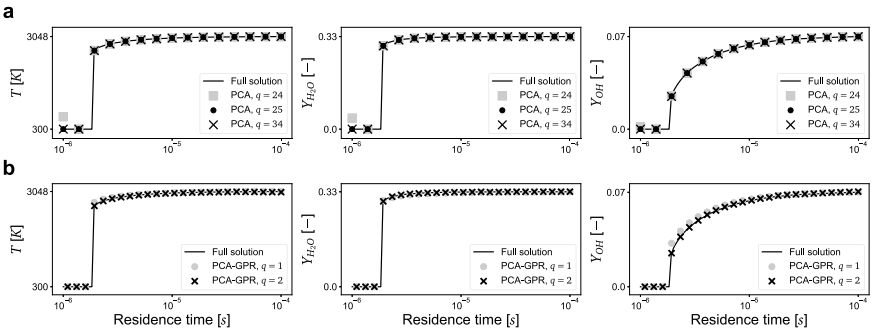

Fig. 11 Results of *a priori* PC-transport simulation of methane/air combustion in a zero-dimensional PSR reactor. Predictions of the temperature, H_2O and OH mass fractions as a function of the residence time in the reactor with the solid line representing the full solution. The results are shown for **a** the PC-transport model without regression using $q = 24$, $q = 25$ and $q = 34$ PCs and **b** the PC-transport model coupled with GPR regression using $q = 1$ and $q = 2$ PCs. Reprinted from (Malik et al. 2018) with permission from Elsevier

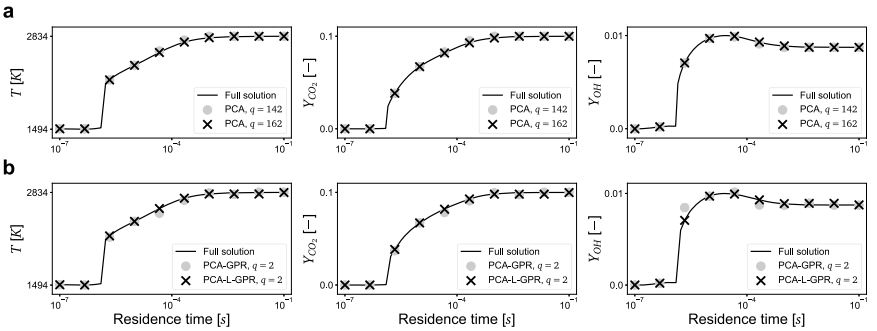

Fig. 12 Results of *a priori* PC-transport simulation of propane/air combustion in a zero-dimensional PSR reactor. Predictions of the temperature, CO_2 and OH mass fractions as a function of the residence time in the reactor with the solid line representing the full solution. The results are shown for **a** the PC-transport model without regression using $q = 142$ and $q = 162$ PCs and **b** the PC-transport model coupled with GPR regression performed globally (PCA-GPR) and locally (PCA-L-GPR) using $q = 2$ PCs. Reprinted from (Malik et al. 2018) with permission from Elsevier

5.2 A Posteriori Validations on Sandia Flame D and F

After validating the PCA-GPR approach in zero-dimensional calculations shown in the previous section, the current section shows the application of the PCA-GPR model in the framework of a non-premixed turbulent flame in a fully three-dimensional LES. The validation was done using the experimental measurements of the Sandia flames D and F (Barlow and Frank 1998). The Sandia flames D and F are piloted methane/air diffusion flames. The fuel is a mixture of CH_4 and air (25/75% by volume) at 294K. The fuel velocity is 49.6m/s for flame D and 99.2m/s for flame F,

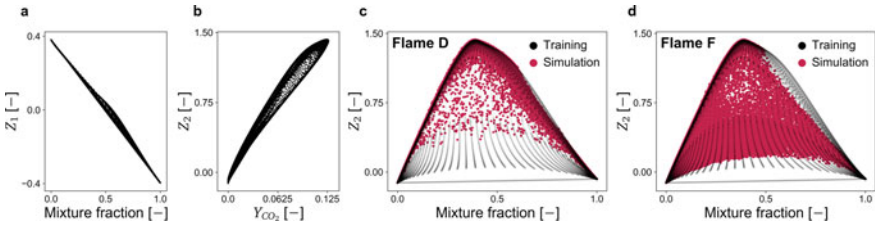

Fig. 13 The two-dimensional manifold obtained during PCA model training versus the manifold accessed during simulation of the Sandia flame D and F. With the training data preprocessing used here, **a** the first PC, Z_1, is highly correlated with mixture fraction and can be linked to the mixture stoichiometry, and **b** the second PC, Z_2, is highly correlated with the CO_2 mass fraction, Y_{CO2}. Z_2 can thus be interpreted as a variable describing reaction progress. **c**–**d** Scatter plots of the PCA manifold obtained from the training dataset (black points) and the manifold accessed during simulation (pink points) of **c** the Sandia flame D, and **d** the Sandia flame F. Points on the simulation-accessed manifolds were down-sampled to 100,000 observations on each plot for clarity. Reprinted from (Malik et al. 2020) with permission from Elsevier

the latter representing the most challenging test case, being close to global extinction. The pilot jet surrounding the fuel consists of burnt gases at 1880K and a low-velocity coflow of air at 291K surrounds the flame.

The dataset for PCA model training is based on unsteady one-dimensional counter-flow diffusion methane flames. The inlet conditions for the fuel and air were set as in the experimental setup. Different counter-flow flames were generated by varying the strain rate, from equilibrium to complete extinction. The dataset generated in this way contained approximately 80,000 observations for each of the state-space variables. The GRI-3.0 chemical mechanism (Smith et al. 2022) (without N_2 species) was used. With the data preprocessing used here (including Pareto scaling and removal of temperature from the state variables), the first PC (Z_1) was highly correlated to the mixture fraction, whereas the second PC (Z_2) can be linked to a progress variable with positive weights for the products and negatives weights for the reactants. These correlations between the PCs and physical variables is shown in Fig. 13a–b. It is interesting to point out that PCA identified these controlling variables without any prior assumptions or knowledge of the system of interest. All the state-space variables, such as temperature, density, species mass fraction as well as the PCs source terms, were regressed as function of Z_1 and Z_2 using GPR (PCA-GPR). A lookup table was then generated for the simulation.

The analysis of the manifold accessed during simulation is also interesting. In Fig. 13c–d, we show the training PCA manifold (black points) overlayed with manifold accessed during simulation of flame D and F respectively (pink points). In both figures, points on the simulation-accessed manifold were down-sampled to 100,000 observations for clarity. It can be observed that both flame D and flame F simulations polled from points that stayed close to the training manifold. The highest density of points for flame D (Fig. 13c) is located near the equilibrium solution. This confirms the experimental findings that flame D does not experience significant extinction and

re-ignition. On the other hand, it can be observed in Fig. 13d that flame F experiences a higher level of extinction and re-ignition phenomena, which was expected from the experimental data. For flame F, the point density is distributed more uniformly between the equilibrium solution and the extinction regions of the training manifold than for flame D. Thus, the manifold accessed during simulation of flame F covers larger region of the training manifold than for flame D.

5.2.1 Simulation Results for Methane/Air Combustion

The simulations were performed in OpenFOAM using tabulated chemistry approach. The PCs were transported, and the dependent variables $\boldsymbol{\phi} = [\mathbf{S}_{\mathbf{Z}_\mathbf{q}}, T, \rho, \mathbf{Y}_i]$ were recovered from nonlinear regression. Details about the numerical setup can be found in (Malik et al. 2020). Figure 14 shows the temperature and the OH mass fraction profiles on the centerline (Fig. 14a), close to the burner exit (Fig. 14b) and further downstream (Fig. 14c) for flame D. It can be observed that the PCA-GPR model was able to reconstruct all variables with great accuracy. Moreover, a comparison is made between the PCA-basis calculated from the full set of 35 species and the PCA-basis computed from the reduced set of five major species only. The results are comparable for both bases, suggesting that using only the major species in order to build the PCA-basis results in no major loss of information. Figure 15 shows a comparison between the experimental and numerical profiles of temperature and selected species mass fraction on the centerline for flame F. The PCA-GPR model accurately predicts the peak and the decay in temperature and the species mass fraction profiles.

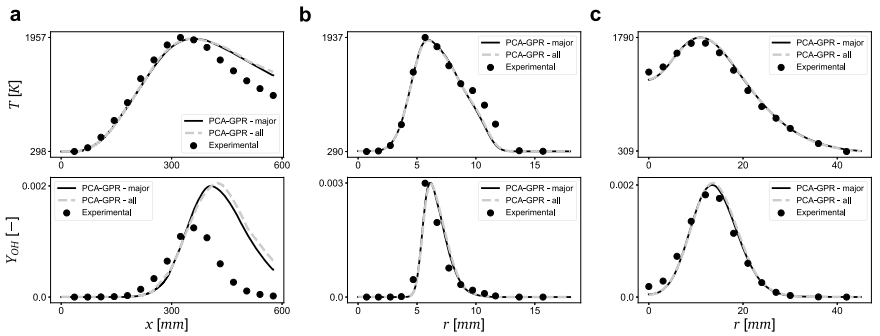

Fig. 14 Results of *a posteriori* PC-transport simulation of the Sandia flame D. Predictions of the temperature and the mass fraction of OH species **a** at the axial and **b**-**c** at the radial profiles. Results show a comparison between the PCA-basis calculated using the major species (PCA-GPR—major), the basis obtained using the full set of species (PCA-GPR—all) and the experimental data. Reprinted from (Malik et al. 2020) with permission from Elsevier

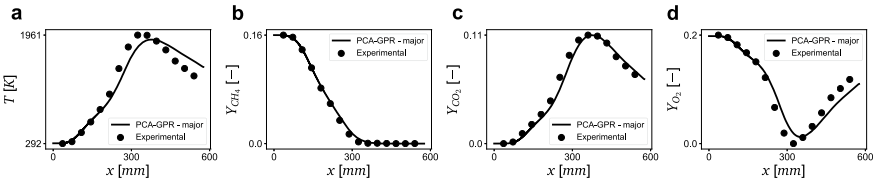

Fig. 15 Results of *a posteriori* PC-transport simulation of the Sandia flame F. Predictions of **a** the temperature and the major species mass fractions, **b** CH_4 **c** CO_2 and **d** O_2 against the experimental data at the flame centerline. The results are shown for the PC-transport model coupled with GPR regression where the PCA-basis was calculated using the major species (PCA-GPR—major). Reprinted from (Malik et al. 2020) with permission from Elsevier

6 Conclusions

In this chapter, we review the complete workflow for data-driven reduced-order modeling of reacting flows. We present strategies for model reduction using dimensionality reduction techniques and nonlinear regression. The originally high-dimensional datasets can be re-parameterized with the new manifold parameters identified directly from training data. The main focus is in the PC-transport approach, where the original system of PDEs is projected to a lower-dimensional PCA-basis. This approach allows for transporting a much smaller number of optimal manifold parameters and yields substantial model reduction. While in this chapter we review recent results from *a priori* and *a posteriori* combustion simulations using PC-transport, several important challenges still remain to be addressed in data-driven modeling of complex systems. For example, topological behaviors on manifolds, such as non-uniqueness or large spatial gradients of dependent variables, can hinder integration of model reduction with nonlinear regression. Possible future research directions that we delineate in this chapter are (1) developing tools for assessing quality of manifolds, (2) developing strategies to mitigate undesired topological behaviors on manifolds and (3) improving our understanding and performance of nonlinear regression models.

Acknowledgements The research of the first author is supported by the F.R.S.-FNRS Aspirant Research Fellow grant. Aspects of this material are based upon work supported by the National Science Foundation under Grant No. 1953350. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program under grant agreement no. 714605.

References

Abadi M et al (2015) TensorFlow: large-scale machine learning on heterogeneous systems. Software available from tensorflow.org

Abboud AW, Schroeder BB, Saad T, Smith ST, Harris DD, Lignell DO (2015) A numerical comparison of precipitating turbulent flows between large-eddy simulation and one-dimensional turbulence. AIChE J 61(10):3185–3197

Ambikasaran S, Foreman-Mackey D, Greengard L, Hogg DW, O'Neil M (2016) Fast direct methods for gaussian processes. IEEE Trans Patt Anal Mach Intell 38(2):252–265

Armstrong E, Sutherland JC (2021) A technique for characterising feature size and quality of manifolds. Combust. Theory Model. 1–23

Barlow RS, Frank JH (1998) Effects of turbulence on species mass fractions in methane/air jet flames. In: Symposium on Combustion, vol 27, pp 1087–1095. Elsevier

Barzegari M, Geris L (2021) An open source crash course on parameter estimation of computational models using a Bayesian optimization approach. J Open Source Educ 4(40):89

Bergstra J, Yamins D, Cox D (2013) Making a science of model search: hyperparameter optimization in hundreds of dimensions for vision architectures. In: Dasgupta S, McAllester D (eds) Proceedings of the 30th international conference on machine learning, vol 28, Proceedings of Machine Learning Research, Atlanta, Georgia, USA, 17–19 June 2013, pp 115–123. PMLR

Biglari A, Sutherland JC (2012) A filter-independent model identification technique for turbulent combustion modeling. Combust Flame 159(5):1960–1970

Biglari A, Sutherland JC (2015) An a-posteriori evaluation of principal component analysis-based models for turbulent combustion simulations. Combust Flame 162(10):4025–4035

Bird RB, Stewart WE, Lightfoot EN (2006) Transport phenomena. Wiley

Chatzopoulos AK, Rigopoulos S (2013) A chemistry tabulation approach via rate-controlled constrained equilibrium (RCCE) and artificial neural networks (ANNs), with application to turbulent non-premixed CH4/H2/N2 flames. Proc Combust Inst 34(1):1465–1473

Chollet F et al (2015) Keras. https://github.com/fchollet/keras

Coussement A, Gicquel O, Parente A (2012) Kernel density weighted principal component analysis of combustion processes. Combust Flame 159(9):2844–2855

Coussement A, Isaac BJ, Gicquel O, Parente A (2016) Assessment of different chemistry reduction methods based on principal component analysis: comparison of the MG-PCA and score-PCA approaches. Combust Flame 168:83–97

Dalakoti DK, Wehrfritz A, Savard B, Day MS, Bell JB, Hawkes ER (2020) An a priori evaluation of a principal component and artificial neural network based combustion model in diesel engine conditions. Proc Combust Inst

D'Alessio G, Sundaresan S, Mueller ME (2022) Automated and efficient local adaptive regression for principal component-based reduced-order modeling of turbulent reacting flows. Proc Combust Inst. https://doi.org/10.1016/j.proci.2022.07.235. https://www.sciencedirect.com/science/article/pii/S1540748922002607

Duvenaud D (2014) Automatic model construction with Gaussian processes. PhD thesis, University of Cambridge

Echekki T, Mirgolbabaei H (2015) Principal component transport in turbulent combustion: a posteriori analysis. Combust Flame 162(5):1919–1933

Echekki T, Kerstein AR, Sutherland JC (2011) The one-dimensional-turbulence model. In: Echekki T, Mastorakos E (eds) Turbulent combustion modeling, Chap. 11. Springer, pp 249–276

Eckart C, Young G (1936) The approximation of one matrix by another of lower rank. Psychometrika 1(3):211–218

Farooq H, Saeed A, Akhtar I, Bangash Z (2021) Neural network-based model reduction of hydrodynamics forces on an airfoil. Fluids 6(9):332

Fooladgar E, Duwig C (2018) A new post-processing technique for analyzing high-dimensional combustion data. Combust Flame 191:226–238

Gicquel O, Darabiha N, Thévenin D (2000) Liminar premixed hydrogen/air counterflow flame simulations using flame prolongation of ildm with differential diffusion. Proc Combust Inst 28(2):1901–1908
Giovangigli V (1999) Multicomponent flow modeling. Birkhäuser, Boston
Gitushi KM, Ranade R, Echekki T (2022) Investigation of deep learning methods for efficient high-fidelity simulations in turbulent combustion. Combust Flame 236:111814
Han X, Jia M, Chang Y, Li Y (2022) An improved approach towards more robust deep learning models for chemical kinetics. Combust Flame 238:111934
Hansen MA, Armstrong E, Sutherland JC, McConnell J, Hewson JC, Knaus, R (2022) Spitfire. https://github.com/sandialabs/Spitfire
Hansen MA, Sutherland JC (2018) On the consistency of state vectors and Jacobian matrices. Combust Flame 193:257–271
Härdle W (1990) Applied nonparametric regression. Cambridge University Press
Hawkes ER, Sankaran R, Sutherland JC, Chen JH (2007) Scalar mixing in direct numerical simulations of temporally evolving plane jet flames with skeletal CO/H2 kinetics. Proc Combust Inst 31(1):1633–1640
Holmes PJ, Lumley JL, Berkooz G, Mattingly JC, Wittenberg RW (1997) Low-dimensional models of coherent structures in turbulence. Phys Rep 287(4):337–384
Hornik K, Stinchcombe M, White H (1989) Multilayer feedforward networks are universal approximators. Neural Netw 2(5):359–366
Humer S, Frassoldati A, Granata S, Faravelli T, Ranzi E, Seiser R, Seshadri K (2007) Experimental and kinetic modeling study of combustion of JP-8, its surrogates and reference components in laminar nonpremixed flows. Proc Combust Inst 31(1):393–400
Isaac BJ, Coussement A, Gicquel O, Smith PJ, Parente A (2014) Reduced-order PCA models for chemical reacting flows. Combust Flame 161(11):2785–2800
Isaac BJ, Thornock JN, Sutherland JC, Smith PJ, Parente A (2015) Advanced regression methods for combustion modelling using principal components. Combust Flame 162(6):2592–2601
Jha PK, Groth CPT (2012) Tabulated chemistry approaches for laminar flames: evaluation of flame-prolongation of ildm and flamelet methods. Combust Theory Model 16(1):31–57
Jolliffe I (2002) Principal component analysis. Springer, New York
Kee RJ, Coltrin ME, Glarborg P (2005) Chemically reacting flow: theory and practice. Wiley
Kerstein AR (1999) One-dimensional turbulence: model formulation and application to homogeneous turbulence, shear flows, and buoyant stratified flows. J Fluid Mech 392:277–334
Keun HC, Ebbels TM, Antti H, Bollard ME, Beckonert O, Holmes E, Lindon JC, Nicholson JK (2003) Improved analysis of multivariate data by variable stability scaling: application to NMR-based metabolic profiling. Anal Chim Acta 490(1–2):265–276
Kutz JN, Brunton SL, Brunton BW, Proctor JL (2016) Dynamic mode decomposition: data-driven modeling of complex systems. SIAM
Lignell DO, Fredline GC, Lewis AD (2015) Comparison of one-dimensional turbulence and direct numerical simulations of soot formation and transport in a nonpremixed ethylene jet flame. Proc Combust Inst 35(2):1199–1206
Lu T, Law CK (2009) Toward accommodating realistic fuel chemistry in large-scale computations. Prog Energy Combust Sci 35(2):192–215
Lusch B, Kutz JN, Brunton SL (2018) Deep learning for universal linear embeddings of nonlinear dynamics. Nat Commun 9(1):1–10
Maas U, Pope SB (1992) Simplifying chemical kinetics: intrinsic low-dimensional manifolds in composition space. Combust Flame 88(3):239–264
Malik MR, Isaac BJ, Coussement A, Smith PJ, Parente A (2018) Principal component analysis coupled with nonlinear regression for chemistry reduction. Combust Flame 187:30–41
Malik MR, Vega PO, Coussement A, Parente A (2020) Combustion modeling using principal component analysis: a posteriori validation on Sandia flames D. E and F, Proc Combust Inst
Malik MR, Coussement A, Echekki T, Parente A (2022a) Principal component analysis based combustion model in the context of a lifted methane/air flame: Sensitivity to the manifold parameters

and subgrid closure. Combust Flame 244:112134. https://doi.org/10.1016/j.combustflame.2022.112134. https://www.sciencedirect.com/science/article/pii/S0010218022001535
Malik MR, Khamedov R, Hernández Pérez FE, Coussement A, Parente A (2022b) Dimensionality reduction and unsupervised classification for high-fidelity reacting flow simulations. Proc Combust Inst. https://doi.org/10.1016/j.proci.2022.06.017. https://www.sciencedirect.com/science/article/pii/S1540748922000207
Mendez MA, Scelzo MT, Buchlin J-M (2018) Multiscale modal analysis of an oscillating impinging gas jet. Exp Therm Fluid Sci 91:256–276
Mendez MA, Balabane M, Buchlin J-M (2019) Multi-scale proper orthogonal decomposition of complex fluid flows. J Fluid Mech 870:988–1036
Mirgolbabaei H, Echekki T (2013) A novel principal component analysis-based acceleration scheme for LES-ODT: an a priori study. Combust Flame 160(5):898–908
Mirgolbabaei H, Echekki T (2014) Nonlinear reduction of combustion composition space with kernel principal component analysis. Combust Flame 161:118–126
Mirgolbabaei H, Echekki T (2015) The reconstruction of thermo-chemical scalars in combustion from a reduced set of their principal components. Combust Flame 162(5):1650–1652
Mirgolbabaei H, Echekki T, Smaoui N (2014) A nonlinear principal component analysis approach for turbulent combustion composition space. Int J Hydrog Energy 39(9):4622–4633
Mockus J (2012) Bayesian approach to global optimization: theory and applications, vol 37. Springer Science & Business Media
Nguyen H-T, Domingo P, Vervisch L, Nguyen P-D (2021) Machine learning for integrating combustion chemistry in numerical simulations. Energy AI 5:100082
Niemeyer KE, Curtis NJ, Sung C-J (2017) pyJac: analytical Jacobian generator for chemical kinetics. Comput Phys Commun 215:188–203
Noda I (2008) Scaling techniques to enhance two-dimensional correlation spectra. J Mol Struct 883–884:216–227
Owoyele O, Echekki T (2017) Toward computationally efficient combustion DNS with complex fuels via principal component transport. Combust Theory Model 21(4):770–798
Parente A, Sutherland JC (2013) Principal component analysis of turbulent combustion data: data pre-processing and manifold sensitivity. Combust Flame 160(2):340–350
Parente A, Sutherland JC, Tognotti L, Smith PJ (2009) Identification of low-dimensional manifolds in turbulent flames. Proc Combust Inst 32(1):1579–1586
Parente A, Sutherland JC, Dally BB, Tognotti L, Smith PJ (2011) Investigation of the MILD combustion regime via principal component analysis. Proc Combust Inst 33(2):3333–3341
Pedregosa F et al (2011) Scikit-learn: machine learning in Python. J Mach Learn Res 12(85):2825–2830
Perry BA, Henry de Frahan MT, Yellapantula S (2022) Co-optimized machine-learned manifold models for large eddy simulation of turbulent combustion. Combust Flame 244:112286. https://doi.org/10.1016/j.combustflame.2022.112286. https://www.sciencedirect.com/science/article/pii/S0010218022003017
Peters N (1988) Laminar flamelet concepts in turbulent combustion. Int Symp Combust 21(1):1231–1250. Twenty-first international symposium on combustion
Pope SB (2013) Small scales, many species and the manifold challenges of turbulent combustion. Proc Combust Inst 34(1):1–31
Punati N, Sutherland JC, Kerstein AR, Hawkes ER, Chen JH (2011) An evaluation of the one-dimensional turbulence model: comparison with direct numerical simulations of CO/H2 jets with extinction and reignition. Proc Combust Inst 33(1):1515–1522
Punati N, Wang H, Hawkes ER, Sutherland JC (2016) One-dimensional modeling of turbulent premixed jet flames—comparison to DNS. Flow Turbul Combust 97(3):913–930 Oct
Raissi M, Perdikaris P, Karniadakis GE (2019) Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J Comput Phys 378:686–707

Ramezanian D, Nouri AG, Babaee H (2021) On-the-fly reduced order modeling of passive and reactive species via time-dependent manifolds. Comput Methods Appl Mech Eng 382:113882
Ranade R, Echekki T (2019) A framework for data-based turbulent combustion closure: a priori validation. Combust Flame 206:490–505
Ranade R, Echekki T (2019) A framework for data-based turbulent combustion closure: a posteriori validation. Combust Flame 210:279–291
Russell S, Norvig P (2022) Artificial intelligence: a modern approach. Prentice Hall
Smith GP, Golden DM, Frenklach M, Moriarty NW, Eiteneer B, Goldenberg M, Bowman CT, Hanson R, Song S, Gardiner Jr WC, Lissianski V, Qin Z (2022) GRI Mech 3.0. Available at: http://www.me.berkeley.edu/gri_mech/
Sutherland JC (2004) Evaluation of mixing and reaction models for large-eddy simulation of non-premixed combustion using direct numerical simulation. PhD thesis, Department of Chemical and Fuels Engineering, The University of Utah
Sutherland JC, Punati N, Kerstein AR (2010) A unified approach to the various formulations of the one-dimensional-turbulence model. Inst Clean Secur Energy
Sutherland JC, Parente A (2009) Combustion modeling using principal component analysis. Proc Combust Inst 32(1):1563–1570
Sutherland JC, Smith PJ, Chen JH (2007) A quantitative method for a priori evaluation of combustion reaction models. Combust Theory Model 11(2):287–303
Taira K, Brunton SL, Dawson STM, Rowley CW, Colonius T, McKeon BJ, Schmidt OT, Gordeyev S, Theofilis V, Ukeiley LS (2017) Modal analysis of fluid flows: an overview. AIAA J 55(12):4013–4041
Taylor R, Krishna R (1993) Multicomponent mass transfer. Wiley
Thrun MC, Stier Q (2021) Fundamental clustering algorithms suite. SoftwareX 13:100642
Van Oijen JA, De Goey LPH (2002) Modelling of premixed counterflow flames using the flamelet-generated manifold method. Combust Theory Model 6(3):463–478
Williams CK, Rasmussen CE (2006) Gaussian processes for machine learning. The MIT Press
Yang Y, Pope SB, Chen JH (2013) Empirical low-dimensional manifolds in composition space. Combust Flame 160(10):1967–1980
Zdybał K, Armstrong E, Parente A, Sutherland JC (2020) PCAfold: python software to generate, analyze and improve PCA-derived low-dimensional manifolds. SoftwareX 12:100630
Zdybał K, D'Alessio G, Aversano G, Malik MR, Coussement A, Sutherland JC, Parente A (2022a) Advancing reactive flow simulations with data-driven models. In: Mendez MA, Ianiro A, Noack BR, Brunton SL (eds) Data-driven fluid mechanics: combining first principles and machine learning, Chap. 15. Cambridge University Press
Zdybał K, Sutherland JC, Parente A (2022b) Manifold-informed state vector subset for reduced-order modeling. Manuscript submitted to Proc Combust Inst 39
Zdybał K, Armstrong E, Sutherland JC, Parente A (2022c) Cost function for low-dimensional manifold topology assessment. Sci Rep 12(1):1–19
Zhang Y, Xu S, Zhong S, Bai X-S, Wang H, Yao M (2020) Large eddy simulation of spray combustion using flamelet generated manifolds combined with artificial neural networks. Energy AI 2:100021
Zhang P, Liu S, Lu D, Sankaran R, Zhang G (2021) An out-of-distribution-aware autoencoder model for reduced chemical kinetics. Discrete Contin Dyn Syst - S
Zhou L, Song Y, Ji W, Wei H (2022) Machine learning for combustion. Energy AI 7:100128