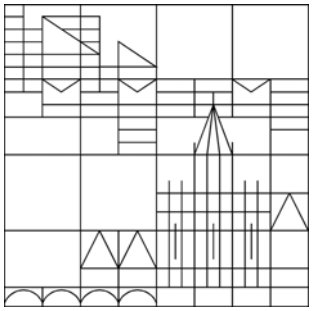

Universität Konstanz

Efficient POD reduced-order modeling for parametrized nonlinear PDE systems

Oliver Lass

Konstanzer Schriften in Mathematik

Nr. 310, November 2012

ISSN 1430-3558

Konstanzer Online-Publikations-System (KOPS)
URL: http://nbn-resolving.de/urn:nbn:de:bsz:352-208711

EFFICIENT POD REDUCED-ORDER MODELING FOR PARAMETRIZED NONLINEAR PDE SYSTEMS

O. LASS

Abstract. In this paper a model order reduction method for a nonlinear elliptic-parabolic system is developed. Systems of this type arise from mathematical models for lithium ion batteries. A non-intrusive reduced order approach based on proper orthogonal decomposition (POD) is presented. In addition to this the interpolation method introduced by Barrault et al. [3] is applied in order to achieve efficient evaluations of the nonlinear coupling terms. Numerical examples illustrate the efficiency of the proposed reduced order modeling technique.

1. Introduction

We consider an elliptic-parabolic partial differential equation (PDE) system consisting of two elliptic and one parabolic equation. Coupled multi component systems of this type can be viewed as generalizations of mathematical models for lithium ion batteries; see, e.g., [8, 18, 20]. The elliptic equations in the nonlinear system of PDEs model the potentials in liquid and solid phase and the parabolic equation the concentration of lithium ions. The three equations are coupled by a strong nonlinear term involving the hyperbolic sine, the square root and the logarithmic function.

The discretization of the nonlinear system of PDEs using, e.g., finite element techniques, lead to very large systems that are expensive to solve. The goal is to develop a reduced order model (ROM) for the nonlinear system of PDEs that is cheap to evaluate. This is motivated by applications like parameter estimations, optimal control and design, where repeated evaluations of the nonlinear systems are required. Therefore the spatial approximation is realized by the Galerkin scheme using proper orthogonal decomposition (POD); see, e.g., [13, 14, 19]. POD is used to generate a basis of a subspace that expresses the characteristics of the expected solution. This is in contrast to more general approximation methods, such as the finite element method, that do not correlate to the dynamics of the underlying system. In this paper we focus on the development of a non-intrusive model order reduction technique. This allows the construction of ROMs independently from the original discretizations. Hence for the construction of the ROM black box solvers

Date: November 15, 2012.

2000 *Mathematics Subject Classification.* 49K20, 65K10, 65K30.

Key words and phrases. Elliptic-parabolic systems, semilinear equations, Newton method, proper orthogonal decomposition, empirical interpolation.

The author gratefully acknowledges support by the German Science Fund *Numerical and Analytical Methods for Elliptic-Parabolic Systems Appearing in the Modeling of Lithium-Ion Batteries* (Excellence Initiative) and the support by the Zukunftsfonds Steiermark project *Prognosesicheres Batteriemodell für Elektro- und Hybridfahrzeuge.*

can be used. This is different to the projection approach used in [5, 16], where the ROMs are obtained by projecting the original algebraic systems onto the POD subspaces. To obtain an efficient reduced order model an empirical interpolation method (EIM) is applied [3]. This method is often used in the combination with the reduced basis approach; see, e.g., [10, 17]. The EIM methods are very effective in reducing the complexity of the evaluation of the nonlinear terms.

Further the POD basis generation is investigated. Under the assumption that several simulations of the nonlinear systems have already been performed two strategies to compute POD basis are investigated. The standard POD approach is compared with an algorithm based on the greedy procedure. The goal is to extract the best possible POD basis from the given data. The strategy involving the greedy procedure is utilized to generate the POD basis functions iteratively. Similar approaches are also used in reduced basis methods [11, 17].

The paper is organized in the following manner: In Section 2 the nonlinear elliptic-parabolic system is formulated. Section 3 is devoted to the POD method. The POD Galerkin approximation is then introduced in Section 4. The numerical realization of the POD Galerkin scheme is presented in Section 5. Here the non-intrusive POD approach is illustrated in details. Different approaches for the computation of the POD basis are presented. Moreover, the EIM is introduced to obtain efficient reduced order models. In Section 6 numerical results are presented to underline the efficiency of the obtained models. Finally, a conclusion is drawn in the last section.

2. The nonlinear elliptic-parabolic system

In this section we formulate the nonlinear elliptic-parabolic system. Suppose that $\Omega = (a, b) \subset \mathbb{R}$, $a < b$, is the spatial domain with boundary $\Gamma = \{a, b\}$. We set $H = L^2(\Omega)$, $V = H^1(\Omega)$ and

$$V_a = \{\varphi \in H^1(\Omega) \,|\, \varphi(a) = 0\}.$$

For the definition of Sobolev spaces we refer, e.g., to [1, 9]. For the terminal time $T > 0$ let $Q = (0, T) \times \Omega$ and $\Sigma = (0, T) \times \Gamma$. The space $L^2(0, T; V)$ stands for the space of (equivalence classes) of measurable abstract functions $\varphi : [0, T] \to V$, which are square integrable, i.e.,

$$\int_0^T \|\varphi(t)\|_V^2 \,\mathrm{d}t < \infty.$$

When t is fixed, the expression $\varphi(t)$ stands for the function $\varphi(t, \cdot)$ considered as a function in Ω only. Recall that

$$W(0, T; V) = \{\varphi \in L^2(0, T; V) \,|\, \varphi_t \in L^2(0, T; V)\}$$

is a Hilbert space supplied with its common inner product; see [2, 6].

For a given parameter $\mu = (\mu_1, \mu_2, \mu_3) \in \mathcal{P}_{ad} \subset \mathbb{R}^3$ the triple $(y, p, q) : Q \to \mathbb{R}$ satisfies the nonlinear elliptic-parabolic systems

$$y_t(t,x) - \big(c_1(x) y_x(t,x)\big)_x + \mathcal{N}(x, y(t,x), p(t,x), q(t,x); \mu) = 0, \tag{2.1a}$$

$$-\big(c_2(y(t,x);\mu) p_x(t,x)\big)_x + \mathcal{N}(x, y(t,x), p(t,x), q(t,x); \mu) = 0, \tag{2.1b}$$

$$-\big(c_3(x) q_x(t,x)\big)_x - \mathcal{N}(x, y(t,x), p(t,x), q(t,x); \mu) = 0 \tag{2.1c}$$

for almost all (f.a.a.) (t,x) in Q together with the homogeneous Neumann boundary conditions for y and p

$$y_x(t,a) = y_x(t,b) = p_x(t,a) = p_x(t,b) \tag{2.1d}$$

and Dirichlet-Neumann for the variable q

$$q(t,a) = 0 \quad \text{and} \quad q_x(t,b) = \mathcal{I}(t) \tag{2.1e}$$

f.a.a. $t \in (0,T)$ and the initial condition

$$y(0,x) = y_\circ(x) \tag{2.1f}$$

f.a.a. $x \in \Omega$. The diffusion coefficients c_1 and c_3 are supposed to be piecewise constant and positive. Further the diffusion coefficient c_2 is positive and nonlinearely dependent on the variable y and a parameter μ. Given the parameter set

$$\mathcal{P}_{ad} = \big\{(\mu_1,\mu_2,\mu_3) \in \mathbb{R}^3 \,\big|\, \mu_1 > 0,\ \mu_2 < 0 \text{ and } \mu_3 \geq 0\big\},$$

we consider a polynomial diffusion coefficient $c_2 : \mathbb{R} \times \mathcal{P}_{ad} \to \mathbb{R}$ of the form

$$c_2(y;\mu) = (1+\mu_3 y)^3. \tag{2.2}$$

The nonlinear coupling term $\mathcal{N} : \Omega \times \mathcal{Z}_{ad} \times \mathcal{P}_{ad} \to \mathbb{R}$ is given by

$$\mathcal{N}(x,z;\mu) = \chi(x)\mu_2\sqrt{y}\sinh\big(\mu_1(q-p) - \ln y\big) \tag{2.3}$$

for $z = (y,p,q) \in \mathcal{Z}_{ad}$ and $\mu = (\mu_1,\mu_2,\mu_3) \in \mathcal{P}_{ad}$, where χ is an indicator function,

$$\mathcal{Z}_{ad} = \big\{(y,p,q) \in \mathbb{R}^3 \,\big|\, y \geq y_{\min}\big\}$$

and $y_{\min} > 0$ holds. Note that the positivity of the variable y is needed to evaluate the term $\ln y$ in the nonlinearity (2.3). Moreover, the given initial condition $y_\circ : \Omega \to \mathbb{R}$ is bounded. With these choices (2.1) can be seen as a generalization of a mathematical model for lithium ion batteries; see, e.g., [8, 18, 20]. Next we introduce the function space

$$Z = \big(W(0,T;V) \times L^2(0,T;V_a) \times L^2(0,T;V_b)\big) \cap L^\infty(Q)^3$$

and call the triple (y,p,q) a *weak solution* to (2.1) if $z = (y,p,q) \in Z$, $y(0) = y_\circ$ in H and

$$\int_\Omega y_t(t)\psi^y + c_1 y_x(t)(\psi^y)' + \mathcal{N}(x,y(t),p(t),q(t);\mu)\psi^y \,\mathrm{d}x = 0, \tag{2.4a}$$

$$\int_\Omega c_2(y(t);\mu)p_x(t)(\psi^p)' + \mathcal{N}(x,y(t),p(t),q(t);\mu)\psi^p \,\mathrm{d}x = 0, \tag{2.4b}$$

$$\int_\Omega c_3 q_x(t)(\psi^q)' - \mathcal{N}(x,y(t),p(t),q(t);\mu)\psi^q \,\mathrm{d}x - \int_\Gamma \mathcal{I}(t)\psi^q \,\mathrm{d}S = 0 \tag{2.4c}$$

for all $\psi^y, \psi^p \in V$ and $\psi^q \in V_a$. It follows from [20] that (2.4) admits a unique weak solution. Therefore, the nonlinear solution operator $\mathcal{S} : \mathcal{P}_{ad} \to Z$ is well-defined, where $z = \mathcal{S}(\mu)$ is the weak solution to (2.1) for the parameter value μ. Note that ψ^f, $f = \{y,p,q\}$, denotes the test functions and can be chosen differently for each equation.

3. The POD method

The goal is to construct a reduced order model to solve (2.4). For this the ansatz functions for the discretization of the nonlinear system should be chosen in an optimal way. Let us next introduce the POD method for the infinite dimensional system (2.4). Suppose that $z = (y, p, q) = \mathcal{S}(\mu)$ is the weak solution to (2.1) for a chosen parameter $\mu \in \mathcal{P}_{ad}$. We explain the computation of the POD basis for the variable y. Suppose that $\mathcal{H}$ denotes either the space H or the space V. Notice that for q we choose H or V_a. The goal is to construct a low dimensional basis by solving the optimization problem

$$\begin{cases} \min\limits_{\psi_1^y,\dots,\psi_{\ell_y}^y \in \mathcal{H}} \displaystyle\int_0^T \Big\| y(t) - \sum_{i=1}^{\ell_y} \langle y(t), \psi_i^y \rangle_{\mathcal{H}}\, \psi_i^y \Big\|_{\mathcal{H}}^2 \,\mathrm{d}t \\ \text{subject to} \quad \langle \psi_i^y, \psi_j^y \rangle_{\mathcal{H}} = \delta_{ij} \quad \text{for} \quad 1 \le i, j \le \ell_y, \end{cases} \tag{3.5}$$

where δ_{ij} stands for the Kronecker symbol, i.e., $\delta_{ij} = 0$ for $i \neq j$ and $\delta_{ii} = 1$. To solve (3.5) let us define the integral operator $\mathcal{R}$. For $y \in L^2(0,T;\mathcal{H})$ let $\mathcal{R} : \mathcal{H} \to \mathcal{H}$ be given by

$$\mathcal{R}\psi^y = \int_0^T \langle y(t), \psi^y \rangle_{\mathcal{H}}\, y(t)\,\mathrm{d}t \quad \text{for } \psi^y \in \mathcal{H}.$$

Clearly, $\mathcal{R}$ is a linear bounded, nonnegative, self-adjoint operator which can be expressed as $\mathcal{R} = \mathcal{Y}\mathcal{Y}^*$, where $\mathcal{Y} : L^2(0,T) \to \mathcal{H}$ is defined by

$$\mathcal{Y}v = \int_0^T v(t)\, y(t)\,\mathrm{d}t \quad \text{for } v \in L^2(0,T).$$

Furthermore, the adjoint $\mathcal{Y}^* : \mathcal{H} \to L^2(0,T)$ is given by

$$(\mathcal{Y}^*\psi^y)(t) = \langle y(t), \psi^y \rangle_{\mathcal{H}} \quad \text{for } \psi^y \in \mathcal{H}.$$

We shall also utilize the operator $\mathcal{K} : L^2(0,T) \to L^2(0,T)$ defined by

$$\mathcal{K} = \mathcal{Y}^*\mathcal{Y}$$

or explicitly as

$$(\mathcal{K}v)(t) = \int_0^T \langle y(t), y(s) \rangle_{\mathcal{H}}\, v(s)\,\mathrm{d}s \quad \text{for } v \in L^2(0,T).$$

For the proof of the following proposition we refer to [13] or [15, Proposition 2.1].

Proposition 3.1. *Except for possibly* 0, $\mathcal{K}$ *and* $\mathcal{R}$ *possess the same eigenvalues which are positive with identical multiplicities. Moreover,* ψ^y *is an eigenfunction associated with a nonzero eigenvalue of* $\mathcal{R}$ *if and only if* $\mathcal{Y}^*\psi^y = \langle y(\cdot), \psi^y \rangle_{\mathcal{H}} \in L^2(0,T)$ *is an eigenfunction of* $\mathcal{K}$.

The solution to (3.5) is given by the eigenfunctions corresponding to the ℓ_y largest eigenvalues λ_i^y of the eigenvalue problem

$$\mathcal{R}\psi_i^y = \int_0^T \langle y(t), \psi_i^y \rangle_{\mathcal{H}}\, y(t)\,\mathrm{d}t = \lambda_i^y \psi_i^y \qquad \text{for } i = 1, \dots, \ell_y, \tag{3.6a}$$

$$\langle \psi_i^y, \psi_j^y \rangle_{\mathcal{H}} = \delta_{ij} \qquad \text{for } i, j = 1, \dots, \ell_y. \tag{3.6b}$$

We shall utilize the POD basis $\{\psi_i^y\}_{i=1}^{\ell}$ with respect to $\mathcal{H} = H$ or $\mathcal{H} = V$ satisfying $\lambda_1^y \geq \ldots \geq \lambda_{\ell_y}^y > 0$,

$$(\mathcal{K}v_i)(t) = \int_0^T \langle y(t), y(s)\rangle_{\mathcal{H}}\, v_i(s)\,\mathrm{d}s = \lambda_i^y v_i(t) \quad \text{for } i = 1, \ldots, \ell^y \tag{3.7}$$

and

$$\psi_i^y = \frac{1}{\sqrt{\lambda_i^y}} \int_0^T v_i(t) y(t)\,\mathrm{d}t \in \mathcal{H} \quad \text{for } i = 1, \ldots, \ell^y.$$

The obtained POD-subspace for the variable y is then denoted by

$$V^{\ell_y} = \operatorname{span}\{\psi_1^y, \ldots, \psi_{\ell_y}^y\}.$$

Note that $\psi_i^y \in V$ holds also for $\mathcal{H} = H$. This follows from (3.6a) by using $y \in L^2(0,T;V)$. In addition, we have the approximation error

$$\int_0^T \Big\| y(t) - \sum_{i=1}^{\ell_y} \langle y(t), \psi_i^y\rangle_{\mathcal{H}}\, \psi_i^y \Big\|_{\mathcal{H}}^2 \,\mathrm{d}t = \sum_{i=\ell_y+1}^{\infty} \lambda_i^y. \tag{3.8}$$

For the solution components p and q we follow the same approach as for y. Hence we obtain POD bases $\{\psi_i^p\}_{i=1}^{\ell_p}$ and $\{\psi_i^q\}_{i=1}^{\ell_q}$, respectively. An analogous result to (3.8) holds for the POD bases $\{\psi_i^p\}_{i=1}^{\ell_p}$ and $\{\psi_i^q\}_{i=1}^{\ell_q}$. The introduced super-indices y, p and q for ψ emphasize that the bases for the three solution components are computed independently.

4. The Galerkin approximation and Reduced Order Modeling

In this section we introduce a Galerkin scheme for (2.1). Additionally, by utilizing the POD method the reduced order model (ROM) is developed.

4.1. **Discretization of the nonlinear system.**

To discretize the nonlinear system (2.1) in space we use a Galerkin scheme. For this we introduce the discrete spaces

$$V^{\ell_g} = \operatorname{span}\{\psi_1^g, \ldots, \psi_{\ell_g}^g\} \subset V$$

for $g = \{y, p, q\}$, where the ψ_i^g's denote the ℓ_g basis functions. At this point we do not make a restriction of what type of basis functions are being used. In the case of piecewise linear finite element discretization then the ψ_i^g's would denote the typical hat functions. For our later approach we here distinguish between the discrete spaces V^{ℓ_y}, V^{ℓ_p} and V^{ℓ_q}. When using a standard finite element discretization this distinction is in general not necessary. We proceed by introducing the standard Galerkin ansatz of the form

$$y^{\ell}(t) = \sum_{i=1}^{\ell_y} \mathrm{y}_i^{\ell}(t)\psi_i^y, \quad p^{\ell}(t) = \sum_{i=1}^{\ell_p} \mathrm{p}_i^{\ell}(t)\psi_i^p, \quad q^{\ell}(t) = \sum_{i=1}^{\ell_q} \mathrm{q}_i^{\ell}(t)\psi_i^q.$$

Here y_i^ℓ, p_i^ℓ and q_i^ℓ denote the time-dependent Galerkin coefficients corresponding to the variables y, p and q. The weak form (2.4) can now be written in the semi-discrete form as

$$\int_\Omega y_t^\ell(t)\psi_i^y + c_1 y_x^\ell(t)(\psi_i^y)' + \mathcal{N}(x, y^\ell(t), p^\ell(t), q^\ell(t); \mu)\psi_i^y \,\mathrm{d}x = 0, \quad 1 \le i \le \ell^y,$$

$$\int_\Omega c_2(y^\ell; \mu) p_x^\ell(t)(\psi_i^p)' + \mathcal{N}(x, y^\ell(t), p^\ell(t), q^\ell(t); \mu)\psi_i^p \,\mathrm{d}x = 0, \quad 1 \le i \le \ell^p,$$

$$\int_\Omega c_3 q_x^\ell(t)(\psi_i^q)' - \mathcal{N}(x, y^\ell(t), p^\ell(t), q^\ell(t); \mu)\psi_i^q \,\mathrm{d}x - \int_\Gamma \mathcal{I}(t)\psi_i^q \,\mathrm{d}S = 0, \quad 1 \le i \le \ell^q.$$

Inserting the Galerkin ansatz for y^ℓ, p^ℓ and q^ℓ the obtained system can be written in matrix-vector form. For this we introduce

$$\left(\left(M_f^{\ell_g,\ell_h}\right)\right)_{ij} = \int_\Omega f(x)\psi_j^g\psi_i^h \,\mathrm{d}x,$$

$$\left(\left(S_f^{\ell_g,\ell_h}\right)\right)_{ij} = \int_\Omega f(x)(\psi_j^g)'(\psi_i^h)' \,\mathrm{d}x,$$

$$\left(\mathcal{N}^{\ell_g}(\mathrm{y}^\ell(t), \mathrm{p}^\ell(t), \mathrm{q}^\ell(t); \mu)\right)_i = \int_\Omega \mathcal{N}(x, y^\ell(t), p^\ell(t), q^\ell(t); \mu)\psi_i^g \,\mathrm{d}x,$$

$$\left(\mathcal{I}^{\ell_g}(t)\right)_i = \int_\Gamma \mathcal{I}(t)\psi_i^g \,\mathrm{d}S$$

for $1 \le i \le \ell_g$ and $1 \le j \le \ell_h$ and

$$\mathrm{y}^\ell(t) = \left(\mathrm{y}_i^\ell(t)\right)_{1\le i\le \ell_y}, \quad \mathrm{p}^\ell(t) = \left(\mathrm{p}_i^\ell(t)\right)_{1\le i\le \ell_p}, \quad \mathrm{q}^\ell(t) = \left(\mathrm{q}_i^\ell(t)\right)_{1\le i\le \ell_q}.$$

For ease of notation we define $M_f^{\ell_g} := M_f^{\ell_g,\ell_g}$ and $S_f^{\ell_g} := S_f^{\ell_g,\ell_g}$. Hence we obtain the system

$$M_1^{\ell_y}\mathrm{y}_t^\ell(t) + S_{c_1}^{\ell_y}\mathrm{y}^\ell(t) + \mathcal{N}^{\ell_y}(\mathrm{y}^\ell(t), \mathrm{p}^\ell(t), \mathrm{q}^\ell(t); \mu) = 0, \tag{4.1a}$$

$$S_{c_2(\mathrm{y}^\ell(t);\mu)}^{\ell_p}\mathrm{p}^\ell(t) + \mathcal{N}^{\ell_p}(\mathrm{y}^\ell(t), \mathrm{p}^\ell(t), \mathrm{q}^\ell(t); \mu) = 0, \tag{4.1b}$$

$$S_{c_3}^{\ell_q}\mathrm{q}^\ell(t) - \mathcal{N}^{\ell_q}(\mathrm{y}^\ell(t), \mathrm{p}^\ell(t), \mathrm{q}^\ell(t); \mu) - \mathcal{I}^{\ell_q}(t) = 0. \tag{4.1c}$$

Note that (4.1) is a semi-discrete system since only the space domain has been discretized. The time domain is still infinite dimensional. By introducing a discretization for the time variable this system can now be solved numerically. For simplicity we here use an equidistant discretization of the time interval and a fully implicit approach. For this we set $\mathrm{y}^{\ell,k} \approx \mathrm{y}^\ell(\mathrm{t}_k)$ and apply the implicit Euler method to (4.1a) and obtain

$$M_1^{\ell_y}\mathrm{y}^{\ell,k} + \delta\mathrm{t}S_{c_1}^{\ell_y}\mathrm{y}^{\ell,k} + \delta\mathrm{t}\mathcal{N}^{\ell_y}(\mathrm{y}^{\ell,k}, \mathrm{p}^{\ell,k}, \mathrm{q}^{\ell,k}; \mu) = M_1^{\ell_y}\mathrm{y}^{\ell,k-1}, \tag{4.2}$$

where $\delta\mathrm{t}$ denotes the step size, $k = 1, ..., N_t$ with N_t the number of time steps and

$$\mathrm{y}_j^{\ell,0} = \int_\Omega y_\circ\psi_j^y \,\mathrm{d}x \quad \text{for} \quad j = 1, \ldots, \ell_y$$

the initial condition. Using (4.2), (4.1b) and (4.1c) we obtain a fully discretized system. The resulting algebraic system is coupled and nonlinear. To solve it numerically in every time step the Newton method is applied. For this we introduce

the function $F : \mathbb{R}^{\ell_y+\ell_p+\ell_q} \to \mathbb{R}^{\ell_y+\ell_p+\ell_q}$ as

$$F(\mathrm{y}^{\ell,k}, \mathrm{p}^{\ell,k}, \mathrm{q}^{\ell,k}) = \begin{pmatrix} F^y(\mathrm{y}^{\ell,k}, \mathrm{p}^{\ell,k}, \mathrm{q}^{\ell,k}) \\ F^p(\mathrm{y}^{\ell,k}, \mathrm{p}^{\ell,k}, \mathrm{q}^{\ell,k}) \\ F^q(\mathrm{y}^{\ell,k}, \mathrm{p}^{\ell,k}, \mathrm{q}^{\ell,k}) \end{pmatrix} \tag{4.3}$$

with

$$\begin{aligned} F^y(\mathrm{y}^{\ell,k}, \mathrm{p}^{\ell,k}, \mathrm{q}^{\ell,k}) &= (M_1^{\ell_y} + \delta \mathrm{t} S_{c_1}^{\ell_y})\mathrm{y}^{\ell,k} + \delta \mathrm{t}\mathcal{N}^{\ell_y}(\mathrm{y}^{\ell,k}, \mathrm{p}^{\ell,k}, \mathrm{q}^{\ell,k}; \mu) - M_1^{\ell_y}\mathrm{y}^{\ell,k-1}, \\ F^p(\mathrm{y}^{\ell,k}, \mathrm{p}^{\ell,k}, \mathrm{q}^{\ell,k}) &= S_{c_2(\mathrm{y}^{\ell,k})}^{\ell_p}\mathrm{p}^{\ell,k} + \mathcal{N}^{\ell_p}(\mathrm{y}^{\ell,k}, \mathrm{p}^{\ell,k}, \mathrm{q}^{\ell,k}; \mu), \\ F^q(\mathrm{y}^{\ell,k}, \mathrm{p}^{\ell,k}, \mathrm{q}^{\ell,k}) &= S_{c_3}^{\ell_q}\mathrm{q}^{\ell,k} - \mathcal{N}^{\ell_q}(\mathrm{y}^{\ell,k}, \mathrm{p}^{\ell,k}, \mathrm{q}^{\ell,k}; \mu) - \mathcal{I}^{\ell_q}(\mathrm{t}). \end{aligned}$$

The Newton method requires the derivative of F. Hence the next step is to compute the Jacobian $J : \mathbb{R}^{\ell_y+\ell_p+\ell_q} \to \mathbb{R}^{(\ell_y+\ell_p+\ell_q)\times(\ell_y+\ell_p+\ell_q)}$ given as

$$J(\mathrm{y}^{\ell,k}, \mathrm{p}^{\ell,k}, \mathrm{q}^{\ell,k}) = \begin{pmatrix} \frac{\partial F^y}{\partial \mathrm{y}^k} & \frac{\partial F^y}{\partial \mathrm{p}^k} & \frac{\partial F^y}{\partial \mathrm{q}^k} \\ \frac{\partial F^p}{\partial \mathrm{y}^k} & \frac{\partial F^p}{\partial \mathrm{p}^k} & \frac{\partial F^p}{\partial \mathrm{q}^k} \\ \frac{\partial F^q}{\partial \mathrm{y}^k} & \frac{\partial F^q}{\partial \mathrm{p}^k} & \frac{\partial F^q}{\partial \mathrm{q}^k} \end{pmatrix} \tag{4.4}$$

with

$$\begin{aligned} \frac{\partial F^y}{\partial \mathrm{y}^k} &= M_1^{\ell_y} + \delta \mathrm{t} S_{c1}^{\ell_y} + \delta t M_{\frac{\partial \mathcal{N}}{\partial \mathrm{y}^k}}^{\ell_y} && \in \mathbb{R}^{\ell_y \times \ell_y} \\ \frac{\partial F^y}{\partial \mathrm{p}^k} &= \delta \mathrm{t} M_{\frac{\partial \mathcal{N}}{\partial \mathrm{p}^k}}^{\ell_y,\ell_p} && \in \mathbb{R}^{\ell_y \times \ell_p} \\ \frac{\partial F^y}{\partial \mathrm{q}^k} &= \delta \mathrm{t} M_{\frac{\partial \mathcal{N}}{\partial \bar{q}^k}}^{\ell_y,\ell_q} && \in \mathbb{R}^{\ell_y \times \ell_q} \\ \frac{\partial F^p}{\partial \mathrm{y}^k} &= C_{\frac{\partial c_2}{\partial \mathrm{y}^k}}^{\ell_p,\ell_y} + M_{\frac{\partial \mathcal{N}}{\partial \mathrm{y}^k}}^{\ell_p,\ell_y} && \in \mathbb{R}^{\ell_p \times \ell_y} \\ \frac{\partial F^p}{\partial \mathrm{p}^k} &= S_{c_2}^{\ell_p} + M_{\frac{\partial \mathcal{N}}{\partial \mathrm{p}^k}}^{\ell_p} && \in \mathbb{R}^{\ell_p \times \ell_p} \\ \frac{\partial F^p}{\partial \mathrm{q}^k} &= M_{\frac{\partial \mathcal{N}}{\partial \mathrm{q}^k}}^{\ell_p,\ell_q} && \in \mathbb{R}^{\ell_p \times \ell_q} \\ \frac{\partial F^q}{\partial \mathrm{y}^k} &= -M_{\frac{\partial \mathcal{N}}{\partial \mathrm{y}^k}}^{\ell_q,\ell_y} && \in \mathbb{R}^{\ell_q \times \ell_y} \\ \frac{\partial F^q}{\partial \mathrm{p}^k} &= -M_{\frac{\partial \mathcal{N}}{\partial \mathrm{p}^k}}^{\ell_q,\ell_p} && \in \mathbb{R}^{\ell_q \times \ell_p} \\ \frac{\partial F^q}{\partial \mathrm{q}^k} &= S_{c_3}^{\ell_q} - M_{\frac{\partial \mathcal{N}}{\partial \mathrm{q}^k}}^{\ell_q} && \in \mathbb{R}^{\ell_q \times \ell_q} \end{aligned}$$

and

$$\left(\left(C_f^{\ell_g,\ell_h}\right)\right)_{ij} = \int_\Omega f(x)\psi_j^g(\psi_i^h)' \,\mathrm{d}x, \quad \text{for } 1 \le i \le \ell_g \text{ and } 1 \le i \le \ell_h.$$

Then the Newton correction δd^k can be computed as

$$J(\mathrm{y}^{\ell,k}, \mathrm{p}^{\ell,k}, \mathrm{q}^{\ell,k})\delta d^k = -F(\mathrm{y}^{\ell,k}, \mathrm{p}^{\ell,k}, \mathrm{q}^{\ell,k}) \tag{4.5}$$

and the update is given as

$$(\hat{\mathrm{y}}^{\ell,k}, \hat{\mathrm{p}}^{\ell,k}, \hat{\mathrm{q}}^{\ell,k}) = (\mathrm{y}^{\ell,k}, \mathrm{p}^{\ell,k}, \mathrm{q}^{\ell,k}) + \delta d^k, \tag{4.6}$$

where $\hat{\mathrm{y}}$, $\hat{\mathrm{p}}$ and $\hat{\mathrm{q}}$ indicate the Newton update. The method is iterative and hence steps (4.5) and (4.6) have to be repeated until a desired stopping criteria is satisfied.

For the convergence of the Newton method we refer the reader, e.g., to [7]. Note that for the Newton method to converge good initial conditions are needed. Here we use a semi-implicit time step, i.e., implicit time discretization with explicit evaluated nonlinear terms. Further to globalize the Newton method we apply a damped Newton iteration. Error orientated framework are known to overcome situations, where the Jacobian matrices are ill-conditioned. Hence the Newton updates satisfy the natural monotonicity test and global convergence can be achieved [7].

Note that there are mixed superindices in the matrices M, C and S, i.e., ℓ_g and ℓ_h with $g \neq h$. This corresponds to the fact that the test and ansatz functions are different discretizations of the space V. This means that in the Jacobian the off diagonal elements arise from a Petrov-Galerkin scheme and hence the resulting matrices can be rectangular. The reason for choosing different ansatz functions for each of the variables lies in the dynamics of the nonlinear system. In order to have the best approximate for each variable, different discretizations of the underlying solution spaces are used.

4.2. **The semi-discrete POD method.** Let us now compute the ansatz functions ψ^g, $g = \{y, p, q\}$, for the ROM. For this we want to apply the POD method from Section 3. Here we assume that solutions to (2.1) are given in discrete form. This solution is considered as the data of which the basis for the ROM will be constructed using the POD method. This discrete solution is usually the solution of a high dimensional discretization like the finite element, finite volume or finite difference method. We here consider a solution given with N_x discretization points in the space domain and N_t discretization points in the time domain for each variable. Let us focus only on the variable y since for p and q the results are obtained analogously. Then the numerical solution for the variable y is denoted as $\mathrm{y}^{N_x,k} \in \mathbb{R}^{N_x}$ for $k = 0, \ldots, N_t$. Further we denote by $\mathrm{x} = (\mathrm{x}_1, \ldots, \mathrm{x}_{N_x})$ and $\mathrm{t} = (\mathrm{t}_0, \ldots, \mathrm{t}_{N_t})$ the spatial and temporal grid points. We use a numerical interpolation method to generate an infinite dimensional representation of the solution. This makes the method independent of the original discretization type since they are all handled in the same way. More details will be given on this in Section 5. We here use the notation $y^{\mathrm{pol}}(t, x)$ for the (piecewise) polynomial representation of the snapshots with the interpolation property

$$y^{\mathrm{pol}}(\mathrm{t}_k, \mathrm{x}_i) = \mathrm{y}_i^{N_x,k} \quad \text{for} \quad i = 1, \ldots, N_x \quad \text{and} \quad k = 0, \ldots, N_t.$$

Here t_k represents the discretization points in time and x_i the discretization points in space. Hence it is guaranteed that the information of the data can be extracted exactly from the polynomials. As snapshots for the POD computation we utilize the solutions to the nonlinear system for the N_t discrete time instance. For that purpose we reformulate (3.5) in the semi-discrete form:

$$\begin{cases} \min\limits_{\{\psi_i^y\}_{i=1}^{\ell_y}} \displaystyle\sum_{k=0}^{N_t} \alpha_k \Big\| y^{\mathrm{pol}}(\mathrm{t}_k, \cdot\,) - \sum_{i=1}^{\ell_y} \langle y^{\mathrm{pol}}(\mathrm{t}_k, \cdot\,), \psi_i^y(\cdot\,) \rangle_{\mathcal{H}}\, \psi_i^y(\cdot\,) \Big\|_{\mathcal{H}}^2 \\ \text{subject to} \quad \langle \psi_i^y(\cdot\,), \psi_j^y(\cdot\,) \rangle_{\mathcal{H}} = \delta_{ij} \text{ for } 1 \leq i, j \leq \ell, \end{cases} \tag{4.7}$$

with α_k nonnegative weights. We here set the α_k to the trapezoidal weights for the integration over the discretized time interval, i.e.,

$$\alpha_0 = \frac{\mathrm{t}_1 - \mathrm{t}_0}{2}, \quad \alpha_k = \frac{\mathrm{t}_{k+1} - \mathrm{t}_{k-1}}{2}, \quad k = 1, \ldots, N_t - 1, \quad \text{and} \quad \alpha_{N_t} = \frac{\mathrm{t}_{N_t} - \mathrm{t}_{N_t-1}}{2}.$$

In our application we have $N_t \ll N_x$. Therefore, we determine the POD basis by using the $N_t \times N_t$ correlation matrix

$$\mathcal{K}_{ij} = \sqrt{\alpha_i}\sqrt{\alpha_j}\langle y^{\mathrm{pol}}(\mathrm{t}_j, \cdot\,), y^{\mathrm{pol}}(\mathrm{t}_i, \cdot\,)\rangle_{\mathcal{H}}.$$

We assume the ℓ_y largest eigenvalues of $\mathcal{K}$ are given in the form $\lambda_1^y \geq \ldots \geq \lambda_{\ell_y}^y > 0$ and hence the POD basis is given by

$$\psi_i^y(x) = \frac{1}{\sqrt{\lambda_i^y}} \sum_{k=0}^{N_t} \sqrt{\alpha_k}(v_i^y)_k y^{\mathrm{pol}}(\mathrm{t}_k, x), \tag{4.8}$$

where $v_i^y \in \mathbb{R}^{N_t}$ are the eigenvectors of $\mathcal{K}$ to the corresponding eigenvalues λ_i^y. Note that the POD basis is given as a linear combination of polynomials and hence are also polynomials. The decay of the eigenvalues is essential since the approximation error for this approach is given by

$$\sum_{k=0}^{N_t} \alpha_k \Big\| y^{\mathrm{pol}}(\mathrm{t}_k, \cdot\,) - \sum_{i=1}^{\ell_y} \langle y^{\mathrm{pol}}(\mathrm{t}_k, \cdot\,), \psi_i^y(\cdot\,)\rangle_{\mathcal{H}} \psi_i^y(\cdot\,) \Big\|_{\mathcal{H}}^2 = \sum_{i=\ell_y+1}^{N_t} \lambda_i^y, \tag{4.9}$$

compare to Section 3. Hence if the decay is too slow the resulting ROM will be of large dimension or one will obtain large approximation errors. There are different approaches of computing the POD basis. In the case of $N_x \ll N_t$ the approach using the operator $\mathcal{R}$ can be used as introduced in Section 3. Different techniques for the computation of the POD basis will be given in Section 5.

4.3. **The reduced order model (ROM).** By applying POD approach (4.7) to each variable y, p and q one obtains the basis functions $\{\psi^y\}_{i=1}^{\ell_y}$, $\{\psi_p\}_{i=1}^{\ell_p}$ and $\{\psi^q\}_{i=1}^{\ell_q}$. Note that the number of computed basis functions can be different for each of the variables. Using these in the Galerkin ansatz for the discrete system (4.2), (4.1b) and (4.1c) we obtain the ROM. When compared to the finite element or finite volume discretization we have a significant decrease in degrees of freedom in the numerical approximation. For all three variables together we obtain $\ell_y + \ell_p + \ell_q$ spatial degrees of freedom. Comparing this, for example, to a finite element discretization with N_x spatial degrees of freedom for each variable we obtain $\ell_y + \ell_p + \ell_q \ll 3N_x$. To illustrate the efficiency of the ROM let us have a look at the decay of the singular values for y, p and q. In Figure 4.1 (left) the decay of the eigenvalues for the three variables is shown together with the first four POD basis functions for the variable y (right). The settings for these results are given in Section 6, Run 1. Since only the $\ell_g, g \in \{y, p, q\}$ largest eigenvalues and corresponding eigenvectors are used it can be seen that the obtained dimension will be low since a fast decay can be observed. Note that the POD basis functions are global compared to the local structure of a finite element basis function. As a result the obtained matrices are full. Hence it is important that the eigenvalues decay fast and the degree of the ROM is low.

5. Numerical realization

In this section we give details on the realization of the proposed method. Different methods for computing the POD basis are described. The focus is especially on a non-intrusive way of the construction of the POD basis and the ROM. The goal is to avoid any dependence of the ROM to the original problem except for the space

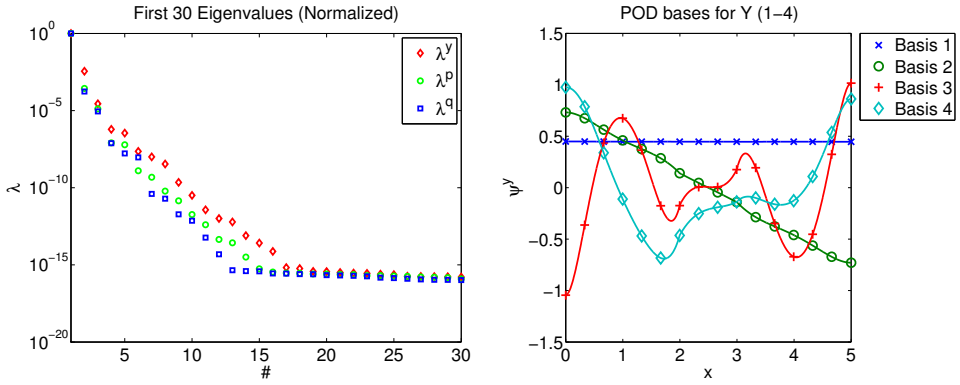

FIGURE 4.1. Decay of the eigenvalues for the three variables y, p and q (left) and the first four POD basis functions for the variable y (right).

and time discretization. The ROM should be constructed entirely from the snapshots. Furthermore, to evaluate the nonlinearities efficiently in the reduced-order approach an empirical interpolation method (EIM) is introduced.

5.1. **POD basis and ROM computation.** In Section 4.2 we introduced the discrete POD method. There the spatial discretization is eliminated using a (piecewise) polynomial representation of the snapshots. In this paper cubic splines are utilized. There are different variants of interpolation methods to avoid oscillation. In the numerical experiments we utilize the implementation of MATLAB using the routine `interp1` with the option `pchip` and the implementation in the GNU SCIENTIFIC LIBRARY (GSL) using the `akima` routines. The MATLAB implementation is a realization of the piecewise cubic Hermite interpolation while the GSL implementation realizes a non-rounded corner algorithm for the cubic spline interpolation. The advantage of the GSL variant is that it can also provide derivatives of the polynomials. The representation of the data by polynomials allows the construction of the semi-discrete POD method since the variables are infinite dimensional in space. This is the first step to make the ROM independent from the discretization of the data.

To obtain fully discrete POD method we have to introduce a discretization of the inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$. This can be done by applying numerical integration and one obtains for $\mathcal{H} = H$

$$\langle y^{\mathrm{pol}}(\mathrm{t}_j, \cdot), y^{\mathrm{pol}}(\mathrm{t}_j, \cdot) \rangle_H = y^{\mathrm{pol}}(\mathrm{t}_j, \mathrm{x}^{\mathrm{int}})^\top W y^{\mathrm{pol}}(\mathrm{t}_j, \mathrm{x}^{\mathrm{int}}) = \sum_{k=1}^{N_{\mathrm{int}}} w_k y^{\mathrm{pol}}(\mathrm{t}_j, \mathrm{x}_k^{\mathrm{int}})^2,$$

where $\mathrm{x}^{\mathrm{int}}$ are appropriately chosen points for the numerical integration of the polynomials and

$$W = \mathrm{diag}(w_1, \ldots, w_{N_{\mathrm{int}}})$$

the weight matrix containing the N_{int} associated integration weights w_i in the diagonal. For the integration we use the Gauss-Legendre quadrature. Note that we obtain an equality since the numerical integration of polynomials can be carried out exactly. In our case we use piecewise cubic polynomials and hence a Gauss

quadrature of order four is required. Note that this approach is not limited to $\mathcal{H} = H$ but also $\mathcal{H} = V$ is possible. In this case additionally the derivatives of the polynomials are included in the inner products.

To compute the POD basis the results from Section 4.2 can be directly applied. Again for the data it is assume that $N_t \ll N_x$. The correlation matrix is given as

$$\mathcal{K} = D^{\frac{1}{2}} Y^\top W Y D^{\frac{1}{2}} \in \mathbb{R}^{N_t \times N_t},$$

where $Y = [y^{\mathrm{pol}}(\mathrm{t}_0, \mathrm{x}^{\mathrm{int}}), \dots, y^{\mathrm{pol}}(\mathrm{t}_{N_t}, \mathrm{x}^{\mathrm{int}})]$ and $D = \mathrm{diag}(\alpha_0, \dots, \alpha_{N_t})$ a diagonal matrix containing the weights for the trapezoidal integration over time. Note that the matrix $\mathcal{K}$ is symmetric and positive definite. Further only the first ℓ_y eigenvalues are needed, hence efficient iterative eigenvalue solvers can be used. The POD basis is then obtained by

$$\psi_i^y(\mathrm{x}^{\mathrm{int}}) = \frac{1}{\lambda_i} Y D^{\frac{1}{2}} v_i \quad \text{for } i = 1, \dots, \ell_y,$$

where v_i is the i-th eigenvector of $\mathcal{K}$ corresponding to the i-th largest eigenvalue λ_i. In the case of $N_x \ll N_t$ a similar approach using the operator $\mathcal{R}$ is used. We then get

$$\mathcal{R} = W^{\frac{1}{2}} Y D Y^\top W^{\frac{1}{2}} \in \mathbb{R}^{N_x \times N_x}.$$

Then the POD basis is obtain by

$$\psi_i^y(\mathrm{x}^{\mathrm{int}}) = W^{-\frac{1}{2}} u_i \quad \text{for } i = 1, \dots, \ell_y,$$

where u_i is the i-th eigenvector of $\mathcal{R}$ corresponding to the i-th largest eigenvalue λ_i. Again $\mathcal{R}$ is symmetric and positive definite. Note that the computation of $W^{\frac{1}{2}}$ and $W^{-\frac{1}{2}}$ is very cheap since these matrices are diagonal matrices. Further both approaches are linked by the singular value decomposition of $\overline{Y} = W^{\frac{1}{2}} Y D^{\frac{1}{2}}$ with the property $\overline{Y} v_i = \sigma_i u_i$ and $\sigma_i = \sqrt{\lambda_i}$. For a comparison of the different approaches the reader is referred, e.g., to [16].

The POD basis $\{\psi_i^y\}_{i=1}^{\ell_y}$ obtained by the above introduced methods are evaluation of polynomials at the integration points. Using equation (4.8) the polynomial representation can also be obtained. Due to the polynomial origin of the basis functions the derivatives can be computed easily. Using this we can now generate the discrete ROM from the system given by equations (4.2), (4.1b) and (4.1c) by evaluating the corresponding integrals. These integrals can again be computed by introducing integration points and corresponding weights. This construction makes the ROM independent of the original discretization. The ROM is computed entirely from the data provided through the snapshots. This is in contradiction to other approaches which often require the availability of the system matrices from the full discretization to perform a projection, compare [5, 16], where the standard POD method is applied to similar systems.

5.2. **Multiple snapshot sets.** Let us now consider the case that multiple snapshot sets are available. Here different values for the parameter μ are considered. This is the case if already several simulations of the nonlinear system have been carried out for different parameter settings of μ. Now all this data should be used to generate the best possible ROM. To the given data we can apply the interpolation as introduced previously. We then can denote the snapshot sets as $Y_k = [y_{\mu_i}^{\mathrm{pol}}(\mathrm{t}_0, \mathrm{x}^{\mathrm{int}}), \dots, y_{\mu_i}^{\mathrm{pol}}(\mathrm{t}_{N_t}, \mathrm{x}^{\mathrm{int}})]$ for $k = 1, \dots, N_s$. Note that we here indicate the dependency of y^{pol} on the parameter μ explicitly by a subindex for clarity.

Let us assume that N_t is the same for all the snapshot sets. This is not a constraint but more of a simplification of notation. We want to look at two approaches for the computation of the POD basis. One approach is to apply the POD method straight to the data while the other uses a greedy algorithm, motivated by the reduced basis method [4, 10, 11, 12].

Let us start with the application of the POD method applied to multiple snapshot sets. The idea is to treat all the sets equally by considering them as one set. For this let us introduce $\mathbb{Y} = [Y_1, \dots, Y_{N_s}]$. Then the correlation matrix can be set up as

$$\tilde{\mathcal{K}} = \tilde{D}^{\frac{1}{2}} \mathbb{Y}^\top W \mathbb{Y} \tilde{D}^{\frac{1}{2}} \in \mathbb{R}^{N_s N_t \times N_s N_t},$$

where $\tilde{D} = \operatorname{diag}(D, \dots, D) \in \mathbb{R}^{N_s N_t \times N_s N_t}$ is a block diagonal matrix and D as previously introduced. The matrix $\tilde{\mathcal{K}}$ can become very large if N_s is large and $N_x \ll N_t N_s$ holds. For this reason using the approach

$$\tilde{\mathcal{R}} = W^{\frac{1}{2}} \mathbb{Y} \tilde{D} \mathbb{Y}^\top W^{\frac{1}{2}} \in \mathbb{R}^{N_x \times N_x},$$

can be a better choice and lead to smaller dimensions. Alternatively also a singular value decomposition can be applied as described previously. The POD basis will be obtained as described before. Both these approaches suffer from the problem that computing the matrix $\tilde{\mathcal{K}}$ or $\tilde{\mathcal{R}}$ is computationally very expensive and might therefore not be feasible in some cases.

Let us now look at the greedy type approach. Computing POD basis functions for the ROM utilizing the greedy method is a standard approach in the reduced basis method for parametrized systems. In the case of time dependent problems this technique was combined with the POD method [11]. In the case here the greedy algorithm will be used to extract a basis from the given snapshot sets. This procedure belongs to the group of *strong* greedy procedures. Compared to the more frequently used *weak* greedy procedures this method does not benefit from rapid computable error estimators [4]. In the approach presented here the true projection error is used as an error indicator. The advantage of this strategy is that the equation under investigation can easily be exchanged without changing the algorithm. Further since it is used to extract a basis of already computed snapshots it is also computationaly efficient. The presented approach is similar to [12]. The main difference to the standard reduced basis method is that the snapshot sets are given and are not computed during the procedure. By using a greedy procedure problem of computing the basis is split up in subproblems and the basis $\{\psi_i^y\}_{i=1}^{\ell_y}$ will be build iteratively. Given N_s snapshot sets, in every iteration the worst approximated snapshot set is chosen to enrich the POD basis. As an indicator we define the projection error over the space-time cylinder, i.e., $Q = \Omega \times [\mathrm{t}_0, \mathrm{t}_{N_x}]$. This approach has the advantage that no further information other than the given data is required.

The detailed algorithm is given in Algorithm 1. By $\Pi_{V^{\ell_y}}$ the orthogonal projection on V^{ℓ_y} with respect to the inner product $\mathcal{H}$ is denoted. Further the POD_L routine extracts the first L POD basis functions obtained by the best fit property (4.7). Here either of the approaches introduced in Section 5.1 can be used to obtain the POD basis. Algorithm 1 always return an orthonormal basis and no further orthogonalization has to be applied since the basis is computed of the projection

Algorithm 1 (The POD-Greedy Algorithm)

Require: N_s snapshot sets $\mathrm{Y}_k = [\mathrm{y}^{N_x,0}, \ldots, \mathrm{y}^{N_x,N_t}]$ associated to different parameter settings, tolerance ϵ^{POD} for the projection error

1: $\Psi \leftarrow POD_L(\mathrm{Y}_1)$, $V^{\ell_y} = \mathrm{span}(\Psi)$

2: Compute projection error for all N_s snapshot sets

$$E_k = \|\mathrm{Y}_k - \Pi_{V^{\ell_y}}(\mathrm{Y}_k)\|^2_{\mathcal{H}(Q)} = \sum_{i=0}^{N_x} \alpha_i \|y^{\mathrm{pol}}_{\mu_k}(\mathrm{t}_i, \cdot) - \Pi_{V^{\ell_y}}(y^{\mathrm{pol}}_{\mu_k}(\mathrm{t}_i, \cdot))\|^2_{\mathcal{H}}$$

3: **while** $\max(E_k) > \epsilon^{\mathrm{POD}}$ **do**

4: $\quad k \leftarrow \arg\max_{k=1,\ldots,N_s} E_i$

5: $\quad \overline{\Psi} \leftarrow POD_L(\mathrm{Y}_k - \Pi_{V^{\ell_y}}(\mathrm{Y}_k))$, $\quad \Psi \leftarrow \Psi \cup \overline{\Psi}$, $\quad V^{\ell_y} = \mathrm{span}(\Psi)$

6: $\quad$ Recompute projection errors E_k for all N_s snapshot sets using Ψ

7: **end while**

8: **return** POD basis Ψ

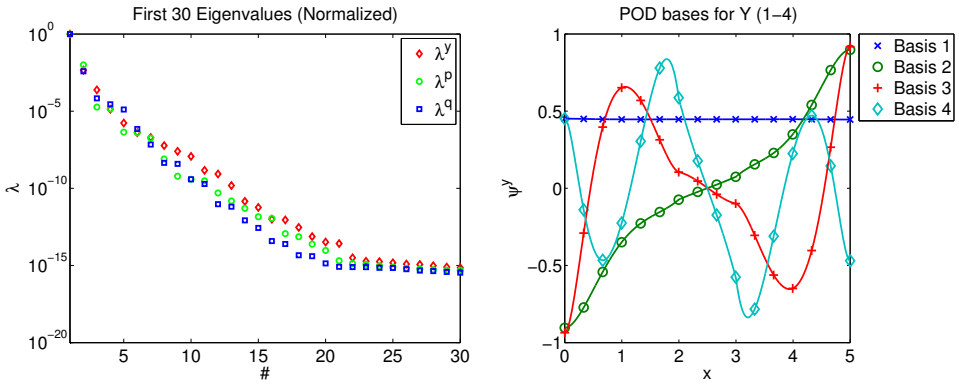

FIGURE 5.1. Decay of the eigenvalues for the three variables y, p and q (left) and the first four POD basis functions for the variable y (right).

residuals which are orthogonal to the previously obtained basis. The orthonormality of the basis is even preserved in the case that a snapshot set is selected more than once. Note that the computation of the projection error in Algorithm 1 can be implemented very efficiently since each E_k only depends on Y_k and hence the evaluation can be done in parallel. Moreover, the dimension of the POD basis is determined by the algorithm and does not have to be set priorly.

Let us shortly compare the obtained POD basis from the two described methods. The settings to obtain these results are given in Section 6, Run 2. In Figure 5.1 (left) the decay of the eigenvalues for the three variables y, p and q is shown together with the first four POD basis for the variable y. In Figure 5.2 (left) the decay of the projection error for the variable y is shown for different choices of L. It can be seen that the projection error decays equally fast for all the three choices. When computing the basis a larger L can achieve a faster basis computation but can also

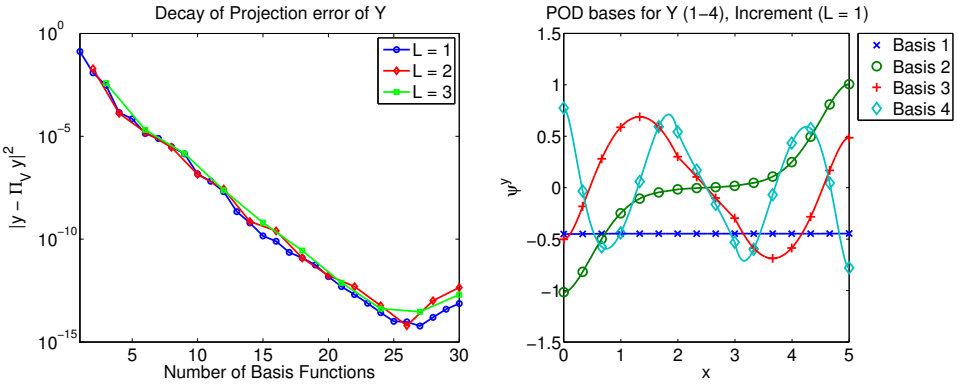

FIGURE 5.2. Decay of the projection error in y for different values of L (left) and the first four POD basis functions obtained by the greedy algorithm using $L = 1$ (right).

lead to a larger basis. The obtained POD basis for $L = 1$ is shown in Figure 5.2 (right). It can be seen that both methods deliver a very similar shaped POD basis. Note that by using a different L in Algorithm 1 the basis might be obtained in a different order.

5.3. **Empirical interpolation methods (EIM).** When applying the POD basis to nonlinear problem one retains the dependency on the dimension of the provided data, the snapshots. When looking at the nonlinear term $\mathcal{N}^{\ell_g}$ for $g \in \{y, p, q\}$ we observe that the integral has to be evaluated on the whole domain Ω, i.e on the number of discretization points in space N_x. Further the nonlinearity is not evaluated at the variables y^ℓ, p^ℓ and q^ℓ but rather at y^ℓ, p^ℓ and q^ℓ. This means that one has to expand the reduced solution to the full discretization of the snapshots in order to integrate the nonlinear term. Hence the ROM is not independent of the dimension of the provided discretization and one obtains a high computational complexity. In the case of nonlinear terms this means that expensive evaluations of nonlinear terms have to be carried out throughout the simulation process. Therefore the empirical interpolation method (EIM) is utilized. This method was introduced in [3] and is often used in combination with the reduced basis approach [10]. The EIM algorithm is based on the greedy algorithm. We will now discuss the basic idea and its application to the nonliearities in this application. The goal is to develop a method to efficiently evaluate the integral

$$\left(\mathcal{N}^{\ell_g}(\mathrm{y}^\ell(t), \mathrm{p}^\ell(t), \mathrm{q}^\ell(t); \mu)\right)_i = \int_\Omega \mathcal{N}(x, y^\ell(t), p^\ell(t), q^\ell(t); \mu)\psi_i^g \,\mathrm{d}x \tag{5.1}$$

for $g \in \{y, p, q\}$. Let us define

$$n(t; \mu) = \mathcal{N}(x, y^\ell(t, x), p^\ell(t, x), q^\ell(t, x); \mu).$$

Using the EIM, the goal is to approximate $n(t; \mu)$ by a Galerkin ansatz of the form

$$n(t; \mu) \approx \sum_{k=1}^{\ell_{\mathrm{EIM}}} \phi_k c_k(t; \mu),$$

where ϕ_k is a linear independent basis, c_k are coefficients depending on t and μ, and ℓ_{EIM} is the dimension of the EIM basis. Inserting this ansatz into (5.1) the nonlinear term $\mathcal{N}^{\ell_g}$ can be approximated as

$$\mathcal{N}^{\ell_g}(\mathrm{y}^\ell(t), \mathrm{p}^\ell(t), \mathrm{q}^\ell(t); \mu) \approx \sum_{k=1}^{\ell_{\mathrm{EIM}}} \phi_k^{\ell_g} c_k(t;\mu) \quad \text{with} \quad \left(\phi_k^{\ell_g}\right)_i = \int_\Omega \phi_k \psi_i^{\ell_g} \,\mathrm{d}x$$

for $i = 1, \ldots, \ell_g$ and $g \in \{y, p, q\}$. The $\phi_k^{\ell_g}$ can be precomputed and hence make ROM independent of the dimension of the space discretization from the given data.

The computation of the basis ϕ_k, $k = 1, \ldots, \ell_{\mathrm{EIM}}$, is based on the greedy algorithm [3]. The EIM method returns the basis ϕ_k and associated interpolation points $\mathrm{x}_k^{\mathrm{EIM}}$. We use the EIM method in a discrete version, i.e., on the evaluation of the nonlinear term at the discrete data. Hence the interpolation points $\mathrm{x}^{\mathrm{EIM}} = (\mathrm{x}_1^{\mathrm{EIM}}, \ldots, \mathrm{x}_{\ell_{\mathrm{EIM}}}^{\mathrm{EIM}})$ are points chosen from the space discretization $\mathrm{x} = (\mathrm{x}_1, \ldots, \mathrm{x}_{N_x})$ and given as indices $\wp_k^{\mathrm{EIM}}$. Since the EIM is an interpolation method the interpolation property is fulfilled, i.e.,

$$Pn(t;\mu) = P\left(\sum_{k=1}^{\ell_{\mathrm{EIM}}} \phi_k c_k(t;\mu)\right),$$

where the matrix $P = (e_{\wp_1^{\mathrm{EIM}}}, \ldots, e_{\wp_{\ell_{\mathrm{EIM}}}^{\mathrm{EIM}}})$ with $e_{\wp_i^{\mathrm{EIM}}} = (0, \ldots, 0, 1, 0, \ldots, 0)^\top \in \mathbb{R}^{N_x}$ a vector with all zeros and at the $\wp_i^{\mathrm{EIM}}$-th row a one. From this property and the fact that the nonlinearity is evaluated pointwise we can compute the coefficient as

$$(P\Phi)\, c = \mathcal{N}(\mathrm{x}^{\mathrm{EIM}}, y^\ell(t, \mathrm{x}^{\mathrm{EIM}}), p^\ell(t, \mathrm{x}^{\mathrm{EIM}}), q^\ell(t, \mathrm{x}^{\mathrm{EIM}})),$$

with $\Phi = (\phi_i, \ldots, \phi_{\ell_{\mathrm{EIM}}})$ and $c = (c_1, \ldots, c_{\ell_{\mathrm{EIM}}})^\top$. Here it is assumed that $(P\Phi) \in \mathbb{R}^{\ell_{\mathrm{EIM}} \times \ell_{\mathrm{EIM}}}$ is invertible. The nonlinearity only has to be evaluated at ℓ_{EIM} points. Depending on the EIM method the matrix $(P\Phi)$ has different properties and hence the computation of the coefficient can be carried out very efficiently. A comparison of different interpolation methods has been investigated in [16].

The EIM method is also applied to the nonlinear diffusion coefficient $c_2(y;\mu)$. Again for c_2 a Galerkin ansatz is carried out and we get

$$S^{\ell_p}_{c_2(\mathrm{y}^\ell;\mu)} \approx \sum_{k=1}^{\ell_{\mathrm{EIM}}} S^{\ell_p}_{\phi_k} c_k.$$

The quantities $S^{\ell_p}_{\phi_k}$ can be precomputed and the computation of the coefficients c_k can be done efficiently as described before. Therefore the obtained ROM again becomes independent from the space discretization of the data. Analogous is the approach for the other nonlinear terms arising in the Newton method.

For completeness in Algorithm 2 the procedure for computing the EIM basis utilized in this article is presented. Note that here the provided data is used to drive the algorithm. When several snapshot sets are available they are treated as a single snapshot set and Algortihm 2 is applied to it. In the algorithm $\Phi_{\wp^{\mathrm{EIM}}}$ denotes the rows of the vector or matrix Φ corresponding to the indices $\wp^{\mathrm{EIM}}$. Again the algorithm is only provided with a tolerance for the interpolation error and returns the basis and points of the required size ℓ_{EIM}. No prior knowledge of the size of ℓ_{EIM} is needed.

Algorithm 2 (The empirical interpolation method (EIM))

Require: Tolerance ϵ^{EIM} and snapshots $\mathcal{N}(\mathrm{x}, \mathrm{y}^{N_x,k}, \mathrm{p}^{N_x,k}, \mathrm{q}^{N_x,k}; \mu)$ for $k = 0, \dots, N_t$

$k \leftarrow \arg\max_{j=0,\dots,N_t} \|\mathcal{N}(\mathrm{x}, \mathrm{y}^{N_x,j}, \mathrm{p}^{N_x,j}, \mathrm{q}^{N_x,j}; \mu)\|_\infty$
$\xi \leftarrow \mathcal{N}(\mathrm{x}, \mathrm{y}^{N_x,k}, \mathrm{p}^{N_x,k}, \mathrm{q}^{N_x,k}; \mu)$
$\text{idx} \leftarrow \arg\max_{j=1,\dots,N_x} |\xi_j|$
$\phi_1 \leftarrow \xi / \xi_{\{\text{idx}\}}$
$\Phi = [\phi_1]$ and $\wp^{\text{EIM}} = \text{idx}$
Solve $\Phi_{\{\wp^{\text{EIM}}\}} c_j = \mathcal{N}(\mathrm{x}, \mathrm{y}^{N_x,j}, \mathrm{p}^{N_x,j}, \mathrm{q}^{N_x,j}; \mu)_{\{\wp^{\text{EIM}}\}}$ for $j = 0, \dots, N_t$
$k \leftarrow \arg\max_{j=0,\dots,N_t} \|\mathcal{N}(\mathrm{x}, \mathrm{y}^{N_x,j}, \mathrm{p}^{N_x,j}, \mathrm{q}^{N_x,j}; \mu) - \Phi c_j\|_\infty$
while $\|\mathcal{N}(\mathrm{x}, \mathrm{y}^{N_x,k}, \mathrm{p}^{N_x,k}, \mathrm{q}^{N_x,k}; \mu) - \Phi c_k\|_\infty > \epsilon^{\text{EIM}}$ **do**
  $\xi \leftarrow \mathcal{N}(\mathrm{x}, \mathrm{y}^{N_x,k}, \mathrm{p}^{N_x,k}, \mathrm{q}^{N_x,k}; \mu)$
  $\text{idx} \leftarrow \arg\max_{j=1,\dots,N_x} |(\xi - \Phi c_k)_{\{j\}}|$
  $\phi_i \leftarrow (\xi - \Phi c_k) / (\xi - \Phi c_k)_{\{\text{idx}\}}$
  $\Phi \leftarrow [\Phi, \phi_i]$ and $\wp^{\text{EIM}} \leftarrow [\wp^{\text{EIM}}, \text{idx}\,]$
  Solve $\Phi_{\{\wp^{\text{EIM}}\}} c_j = \mathcal{N}(\mathrm{x}, \mathrm{y}^{N_x,j}, \mathrm{p}^{N_x,j}, \mathrm{q}^{N_x,j}; \mu)_{\{\mathrm{p}^{EIM}\}}$ for $j = 0, \dots, N_t$
  $k \leftarrow \arg\max_{j=0,\dots,N_t} \|\mathcal{N}(\mathrm{x}, \mathrm{y}^{N_x,j}, \mathrm{p}^{N_x,j}, \mathrm{q}^{N_x,j}; \mu) - \Phi c_j\|_\infty$
end while
return Φ and $\wp^{\text{EIM}}$

5.4. **Comparison.** The presented approach of generating the ROM relies on the weak formulation of the problem and does not require any additional information from the original system. It is non-intrusive since the ROM is entirely generated from the provided snapshots. The only link to the original problem is the discretization which is used for the integration. There are several differences compared to the standard ROM approaches, which are based on the projection of the original system onto the subspace spanned by the POD basis. The most significant difference is that no system matrices are needed from the original discretization since they are computed in the reduced form and are not projected. On the other hand this strategy results in a larger implementation effort, where as the projection approach can be implemented very easy. Another observation is that the presented reduced order modeling technique performs better when no EIM is available. This can be the case when the nonlinear term is not known, i.e., a parameter in an optimization process. The pros and cons are summarized in Table 5.1. With *integration* we refer to the proposed approach while with *projection* we refer to the standard approach which was also used in [16] for similar equations. Further numerical comparisons are carried out in Section 6 to underline the differences.

6. Numerical experiments

In this section some numerical results comparing the proposed approach of *integration* with the more common approach using a *projection* are presented. Errors with respect to a finite element reference solution are presented and the performance gain of the different approaches is compared.

In the presented numerical results we use the following settings. The system under consideration is (2.1). For the domain Ω the one dimensional interval $[0, 5]$

	Pros	Cons
Integration	• Independent of original discretization • More efficient when no EIM available	• More implementation efforts
Projection	• Easy to implement	• Need of original system matrices ($\Psi^\top S\Psi$) • Inefficient when no EIM available

TABLE 5.1. Comparison of two different reduced order modeling approaches.

is chosen. The linear diffusion coefficients are set to

$$c_1 = \begin{cases} 2 & x \in [0,2] \cup [3,5], \\ 1 & \text{else}, \end{cases} \qquad c_3 = \begin{cases} 1 & x \in [0,2] \cup [3,5], \\ 0.001 & \text{else}. \end{cases}$$

Further the nonlinear diffusion coefficient is chosen as (2.3) and the nonlinearity $\mathcal{N}$ is set to

$$\mathcal{N}(x,y,p,q;\mu) = \begin{cases} \mu_2\sqrt{y}\sinh\left(\mu_1(q-p) - \ln y\right) & x \in [0,2] \cup [3,5], \\ 0 & \text{else}. \end{cases}$$

For the discretization in space we choose 1000 discretization points and second order finite elements. This results in 1999 degrees of freedom. When using the finite element method we use the same ansatz and test functions for each of the variables. The time interval $[0,T]$, $T = 4$, is discretized in 400 equidistant steps. For the time dependent boundary (2.1e) we choose

$$I(t) = \frac{t}{2}\sin(2\pi t).$$

As initial condition for the time dependent variable y we use $y_\circ(x) = 1$. With these settings the snapshots are generated. The values for the parameter μ together with the settings for the POD and EIM method are described for each of the numerical experiments individually.

6.1. **Run 1.** In this numerical test we want to see how good the solution can be recovered by the ROM. For the parameter μ we choose $(1, -1, 0.5)$. We use the finite element solutions to compute the POD basis. This basis is then used to generate the ROM.

To measure the accuracy of the POD method with respect to the finite element method the average relative error is introduced in the form

$$\varepsilon_y = \frac{1}{N_t}\sum_{i=0}^{N_t} \frac{\|y^{N_x}(\mathrm{t}_i) - y^\ell(\mathrm{t}_i)\|_{L^2(\Omega)}}{\|y^{N_x}(\mathrm{t}_i)\|_{L^2(\Omega)}}, \tag{6.1}$$

where N_t is the number of time steps and t_i the discretization points in time. By y^{N_x} the finite element solution is denoted and by y^ℓ the reduced order solution.

In the experiment we use 9 POD basis functions for the variables y and p and 8 POD basis functions for the variable q. Further, we choose $\epsilon^{\text{EIM}} = 10^{-9}$ as tolerance for the interpolation error in Algorithm 2 which results in 31 EIM basis functions for each of the two nonlinear terms. In Table 6.2 the relative average errors in each of the variables is compared for the two different approaches. By *projection* we denote the classical model order reduction approach, where the original problem is projected to obtain a ROM. Note that for this approach the matrices of the original problem are needed. By *integration* we denote the new approach, where the integrals to obtain the the ROM are evaluated exactly with the help of polynomials. It can be seen that both methods deliver the same accuracy. Further in Table 6.2 also the application of the EIM method is investigated and its effect on the relative error. Also here it can be seen that there are no significant differences.

		Projection		Integration	
		L^∞	L^2	L^∞	L^2
ROM	ε_y	4.4290×10^{-5}	2.2605×10^{-5}	7.3189×10^{-5}	3.5286×10^{-5}
	ε_p	3.3028×10^{-5}	2.2785×10^{-5}	4.9109×10^{-5}	3.6869×10^{-5}
	ε_q	3.1712×10^{-5}	2.6428×10^{-5}	7.3370×10^{-5}	6.4670×10^{-5}
ROM-EIM	ε_y	4.4290×10^{-5}	2.2605×10^{-5}	7.3189×10^{-5}	3.5286×10^{-5}
	ε_p	3.3029×10^{-5}	2.2785×10^{-5}	4.9109×10^{-5}	3.6869×10^{-5}
	ε_q	3.1712×10^{-5}	2.6428×10^{-5}	7.3370×10^{-5}	6.4670×10^{-5}

TABLE 6.2. Comparison of the average relative errors when using the two different approaches for the computation of model order reduction and with and without EIM.

To compare the the performance further we have a look at the computational time. Table 6.3 compares the time for the different model order reduction approaches and the application of the EIM method. Moreover, also a comparison to the finite element method is shown. Note that both model order reduction methods perform very similar when EIM is used. Compared to the finite element method a speedup factor of approximately 40 can be achieved. When EIM is not used one can see the advantage of the new approach. Here still a speedup factor of approximately 7 can be achieved using the approach which is independent of the finite element method.

	Projection	Integration	FEM
EIM	2.36	2.33	–
NO EIM	38.1	13.5	92.1

TABLE 6.3. Comparison of the simulation time in seconds needed by the different methods.

6.2. **Run 2.** In the second numerical experiment we focus on the performance of the ROM using the integration approach. For this we introduce a *sample* set for the parameter μ and test how good the approximation of the ROM is when we

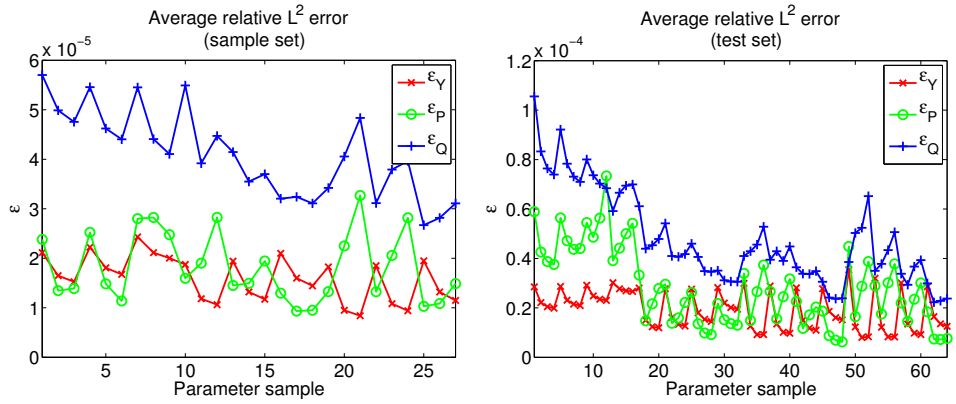

FIGURE 6.1. Run 5.2: Average relative error for the sample (left) and the test (right) set using Greedy-POD with EIM.

choose parameters of a *test* set. Let us start with the settings. As our sample set we choose

$$\mu \in \{1, 2, 3\} \times \{-3, -2, -1\} \times \{0.5, 1, 1.5\}.$$

This gives 27 possible combinations for which we will solve the FE model. Then we use the POD approach to compute a POD basis from the obtained snapshots. Here the two methods described in Section 5.2 will be utilized. In the Greedy-POD approach we set the tolerance for the projection error to $\epsilon^{\text{POD}} = 10^{-10}$ and the increment $L = 2$. This settings lead to 18 POD basis functions for the variable y, 16 for the variable q and 14 for the variable q. For the approach not using the greedy algorithm the same number of POD basis functions are used for comparison. For the EIM method we choose the tolerance $\epsilon^{\text{EIM}} = 10^{-9}$ which results in 48 basis functions for $\mathcal{N}$ and 47 for c_2. To test the obtained results from the ROM we introduce a test set

$$\mu \in \{0.5, 1.5, 2.5, 3.5\} \times \{-3.5, -2.5, -1.5, -0.5\} \times \{0.25, 0.75, 1.25, 1.75\}.$$

The test set gives us 64 possible combinations for the choice of the parameter μ. To measure the performance of the ROM we look at the average relative L^2 error as previously introduced.

In Figure 6.1 the results using the Greedy-POD approach is shown. The left plot shows the error for the sample set and the right plot the error for the test set. It can be seen that the ROM performs very well with error in the range of 10^{-5} which is also the setting for the projection error ($\sqrt{\epsilon^{\text{EIM}}}$). The result for the ROM not using the Greedy-POD algorithm for the basis generation are shown in Figure 6.2. It can be seen that both approachs obtain a very similar accuracy measured in average relative errors. In Table 6.4 the computational time for the generation of the ROM is shown. Since the parallel implementation provided by MATLAB can not be apply to this problem in an sophisticated way, there is very little difference in the computation times.

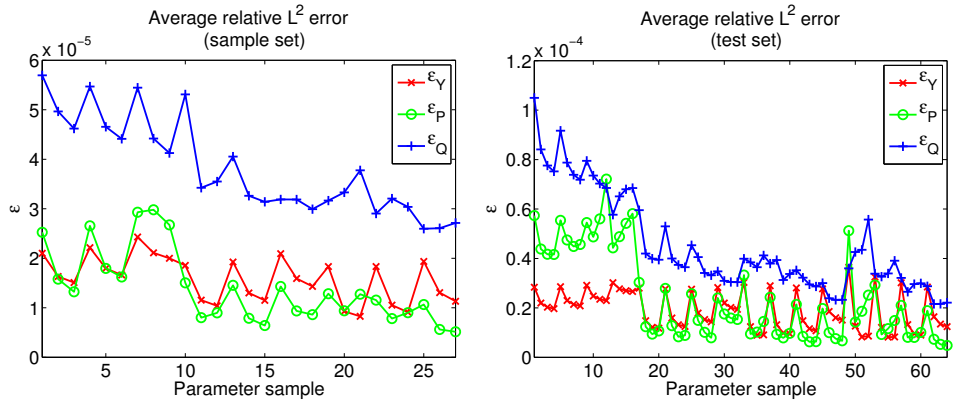

FIGURE 6.2. Run 5.2: Average relative error for the sample (left) and the test (right) set using POD with EIM.

	Greedy-POD	POD	EIM	FEM	ROM	ROM-EIM
CPU time (s)	29	30	45	$\sim 100^*$	$\sim 14^*$	$\sim 2.5^*$

TABLE 6.4. CPU time in seconds for the different basis generation procedures (Greedy-POD, POD, EIM) and for solving the nonlinear system (FEM, ROM, ROM-EIM). (* CPU time varies due to different parameter choices for μ.)

7. Conclusions

In this paper the POD method is utilized to developed a non-intrusive ROM for a nonlinear elliptic-parabolic system. It was shown that the basis computation can be carried out in a black box manner. Compared to the sandard ROM approaches, the presented technique can generated the ROM independetly from the original discretization, i.e., no system matrices from the original discretization are needed. It turns out that the proposed method competes well with the standard implementation of projection. Very good speed up factors can be achieved with and without the empirical interpolation method.

References

[1] R.A. Adams. *Sobolev Spaces*. Pure and Applied Mathematics, Vol. 65. Academic Press, New York-London, 1975.

[2] H.W. Alt. *Lineare Funktionalanalysis. Eine anwendungsorientierte Einführung*. Springer-Verlag, Berlin, 1992.

[3] M. Barrault, Y. Maday, N. C. Nguyen and A. T. Patera. An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations. *Comptes Rendus Mathematique*, 339(9):667-672, 2004.

[4] A. Buffa, Y. Maday, A. T. Patera, C. Prudhomme and G. Turinici. A priori convergence of the Greedy algorithm for the parametrized reduced basis method. *ESAIM: Mathematical Modelling and Numerical Analysis*, 46, 595-603, 2012.

[5] L. Cai and R. E. White. Reduction of Model Order Based on Proper Orthogonal Decomposition for Lithium-Ion Battery Simulations. *Journal of The Electrochemical Society*, 156(3):A154-A161, 2009.

[6] R. Dautray and J.-L. Lions. *Mathematical Analysis and Numerical Methods for Science and Technology. Volume 5: Evolution Problems I*. Springer-Verlag, Berlin, 1992.
[7] P. Deuflhard. *Newton Methods for Nonlinear Problems. Affine Invariance and Adaptive Algorithms*. Series Computational Mathematics 35, Springer, Berlin, 2004.
[8] M. Doyle, T. Fuller and J. Newman. Modeling of galvanostatic charge and discharge of the lithium ion battery/polymer/insertion cell. *Journal of the Chemical Society*, 140(6):1526-1533, 1993.
[9] L.C. Evans. *Partial Differential Equations*. American Mathematical Society, Providence, Rhode Island, 2002.
[10] M. Grepl. Certified reduced basis method for nonaffine linear time-varying and nonlinear parabolic partial differential equations. *WSPC: Mathematical Models and Methods in Applied Sciences*, 22(3), 2012.
[11] B. Haasdonk. Convergence rates of the POD-greedy method. *ESAIM: Mathematical Modelling and Numerical Analysis*, 2012.
[12] B. Haasdonk, J. Salomon and B. Wohlmuth. A Reduced Basis Method for the Simulation of American Options. *ENUMATH Proceedings*, 2012.
[13] P. Holmes, J.L. Lumley, G. Berkooz and C.W. Rowley. *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*. 2nd ed., Cambridge Monographs on Mechanics, Cambridge University Press, 2012.
[14] K. Kunisch and S. Volkwein. Galerkin proper orthogonal decomposition methods for parabolic problems. *Numerische Mathematik*, 90:117-148, 2001.
[15] K. Kunisch and S. Volkwein. Proper orthogonal decomposition for optimality systems. *ESAIM: Mathematical Modelling and Numerical Analysis*, 42:1-23, 2008.
[16] O. Lass and S. Volkwein. *POD Galerkin schemes for nonlinear elliptic-parabolic systems*. Submitted, 2011.
[17] A.T. Patera and G. Rozza. *Reduced Basis Approximation and A Posteriori Error Estimation for Parametrized Partial Differential Equations*. MIT Pappalardo Graduate Monographs in Mechanical Engineering, 2006.
[18] P. Popov, Y. Vutov, S. Mergenov, and O. Iliev. Finite volume discretization of equations describing nonlinear diffusion in li-ion batteries. *Numerical Methods and Applications, Lecture Notes in Computer Science*, 6046:338-346, 2011.
[19] L. Sirovich. Turbulence and the dynamics of coherent structures. Parts I-II. *Quarterly of Applied Mathematics*, XVL:561-590, 1987.
[20] J. Wu, J. Xu, and H. Zou. On the well posedness of mathematical model for lithium-ion battery systems. *Methods and Applications of Analysis*, 13:275-298, 2006.

Oliver Lass, Universität Konstanz, Fachbereich Mathematik und Statistik, Universitätsstrasse 10, D-78457 Konstanz, Germany
E-mail address: Oliver.Lass@uni-konstanz.de