Technical Report LPT–2002–22

Projection Based Model Reduction for Dynamic Optimization

M. Schlegel[1], J. v.d. Berg[2], W. Marquardt[1], O.H. Bosgra[2]

August 2002

Contribution to:
AIChE Annual Meeting 2002,
Indianapolis, 3–8.11.2002.

[1] Lehrstuhl für Prozesstechnik
RWTH Aachen, D–52056 Aachen

[2] Systems and Control Group
Delft University of Technology
Mekelweg 2, NL–2626 CD Delft

Enquiries should be addressed to:

Lehrstuhl für Prozesstechnik
RWTH Aachen
Templergraben 55
D–52056 Aachen

Tel.:	+49 / 241 / 80 94668
Fax:	+49 / 241 / 80 92326
E–Mail:	secretary@lfpt.rwth–aachen.de

Projection based nonlinear model reduction for dynamic optimization*

M. Schlegel[1], J. van den Berg[2], W. Marquardt[1], O. H. Bosgra[2]

[1]Lehrstuhl für Prozesstechnik, RWTH Aachen University, Turmstr. 46, 52064 Aachen, Germany

[2]Systems and Control Group, Mech. Eng. TU Delft, Mekelweg 2, 2628 CD Delft, The Netherlands

Abstract

In this work we investigate the applicability of projection based model reduction techniques in the context of dynamic optimization. Especially, we focus on the question, whether dynamic optimization with a reduced model can give the same solution in less computation time, as compared to optimization with the original model.

Keywords: dynamic optimization, model reduction, projection methods, POD

Prepared for Presentation at the AIChE Annual Meeting Indianapolis November 3-8, 2002, Nonlinear Model Reduction

*This work has been funded by the European Commission under grant G1RD–CT–1999–00146 in the "INCOOP" project.

1 Introduction

Dynamic optimization becomes increasingly important to enhance operation of chemical processes. Examples include the design of operational strategies for batch and semi-batch reactors or for continuous processes during transient phases such as grade transitions, start-up or shut-down on the one hand, and real-time applications for optimization-based monitoring and control on receding horizons on the other hand. Large-scale and strongly nonlinear problems with differential-algebraic as well as path and endpoint constraints often have to be considered resulting in demanding computational problems. Obviously, the computation time required by the solution of dynamic optimization problems strongly depends on the size and the structure of the process model considered. This leads to the fact that for large-scale processes the optimization problems often still cannot be solved sufficiently fast, especially not in real-time applications. To cope with this problem, in principal the problem can be approached from two sides: On the on hand, the algorithms for dynamic optimization solvers are subject to improvement. On the other hand, one could apply model reduction and model simplification techniques in order to reduce the computational effort which is related to the size of the model. In this work we investigate the benefit of model reduction for dynamic optimization. Since in transient plant operation the operational envelope of a process usually covers a large region of the state space, the process dynamics cannot be represented adequately by a linear model. Therefore, nonlinear models and the use of nonlinear model reduction techniques is required. In the literature, many approaches for nonlinear model reduction techniques can be found (see Marquardt (2001) for a review). Model order reduction techniques are among the most popular approaches. Usually, model order reduction is carried out by using some projection technique. By means of projection, the original state space is transformed into a state space better revealing the important process dynamics. To achieve model order reduction, the transformed space is decomposed into two complementary subspaces. The reduction is finally achieved by truncation of the non-dominant states or by residualization. Various projection techniques have been suggested, such as nonlinear balancing or proper orthogonal decomposition (POD), and often quite significant reductions in model order have been reported, mainly for applications with distributed parameter systems.

However, for the application in dynamic optimization, some issues are quite important, which usually have not been considered in the context of projection based model reduction. What matters at first is really a reduction in computational complexity, not in model order. Afanasiev and Hinze (2001) successfully used proper orthogonal decomposition for an optimal control application in fluid dynamics, envolving distributed parameters systems. For lumped parameter systems, e.g. those arising in chemical engineering applications, projection techniques tend to yield smaller, but more complex models, which do not retain the structure of the original model. As we will show, this is a major bottleneck of these techniques. Also, it is desirable that optimization with a reduced model yields a solution, which is as close as possible to the solution obtained with the full model regarding optimality and feasibility. Such a behavior is not necessarily guaranteed. In this contribution we will show the application of a projection method for nonlinear reduction to a chemical process model. The reduced model will be used in a dynamic optimization algorithm using the so-called sequential approach. The results are compared to those obtained with the full model regarding both, computational and economic performance. Benefits and drawbacks of nonlinear model reduction for this application are discussed.

2 Dynamic optimization

2.1 Problem formulation

We focus on the use of dynamic optimization for optimal trajectory generation and consider an optimal control problem formulation of the following form:

$$\min_{\boldsymbol{u}(t),\boldsymbol{p},t_f} \; \Phi\left(\boldsymbol{x}\left(t_f\right),\boldsymbol{y}\left(t_f\right)\right) \tag{P1}$$

$$\begin{aligned}
\text{s.t.} \quad \dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{u},\boldsymbol{p},t)\,, \quad t \in [t_0,t_f]\,, && (1)\\
0 &= \boldsymbol{g}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{u},\boldsymbol{p},t)\,, \quad t \in [t_0,t_f]\,, && (2)\\
0 &= \boldsymbol{x}(t_0) - \boldsymbol{x}_0\,, && (3)\\
0 &\geq \boldsymbol{h}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{u},\boldsymbol{p},t)\,, \quad t \in [t_0,t_f]\,, && (4)\\
0 &\geq \boldsymbol{e}\left(\boldsymbol{x}\left(t_f\right),\boldsymbol{y}\left(t_f\right)\right)\,. && (5)
\end{aligned}$$

In this formulation, $\boldsymbol{x}(t) \in \mathbb{R}^{n_x}$ denote the differential state variables with initial conditions $\boldsymbol{x}_0$. $\boldsymbol{y}(t) \in \mathbb{R}^{n_y}$ are the algebraic state variables. The variables to be determined by the optimization procedure for minimization of the objective function Φ are the control vector $\boldsymbol{u}(t) \in \mathbb{R}^{n_u}$, the unknown time–independent parameters $\boldsymbol{p} \in \mathbb{R}^{n_p}$, as well as the final time t_f. The differential–algebraic (DAE) model is given by the equation system (1), (2). We only consider DAE systems with an differential index of less than or equal to one. Furthermore, path constraints (4) can be applied to the states, control variables and time–independent parameters. Finally, endpoint constraints (5) on the state variables can be employed.

2.2 Sequential solution approach

In the sequential approach (Kraft, 1985) the control profiles $u_i(t), i = 1, ..., n_u$ have to be approximated, and often piecewise polynomial expansions of the form

$$u_i(t) \approx \overline{u}_i(\boldsymbol{c}_i, t) = \sum_{k \in \Lambda_i} c_{i,k} \phi_{i,k}(t), \tag{6}$$

are used, where Λ_i denotes the index set of the chosen parameterization functions $\phi_{i,k}(t)$ and the vector $\boldsymbol{c}_i$ contains the corresponding parameter vector. For brevity, in this paper we only consider piecewise constant functions $\phi_{i,k}(t) := 1 \quad \forall\, t_k \leq t \leq t_{k+1}$, otherwise $\phi_{i,k}(t) = 0$, though an extension to higher–order polynomials is possible. The grid points for each u_i are contained in the mesh $\Delta_{\Lambda_i} := \{t_k | k \in \Lambda_i\}$.

By discretization of the control variables, the dynamic optimization problem (P1) can be reformulated as an NLP:

$$\min_{\boldsymbol{c},\boldsymbol{p},t_f} \Phi \left(\boldsymbol{x}\left(\boldsymbol{c}, \boldsymbol{p}, t_f\right), \boldsymbol{y}\left(\boldsymbol{c}, \boldsymbol{p}, t_f\right)\right) \tag{P2}$$

$$\begin{aligned} \text{s.t.} \quad 0 &\geq \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{c}, \boldsymbol{p}, t_i), \quad \forall t_i \in \Delta_\Lambda, & (7)\\ 0 &\geq \boldsymbol{e}\left(\boldsymbol{x}\left(t_f\right), \boldsymbol{y}\left(t_f\right)\right). & (8) \end{aligned}$$

The path constraints (7) are now evaluated on the unified mesh of all control variables $\Delta_\Lambda := \bigcup_{i=1}^{n_u} \Delta_{\Lambda_i}$. The DAE model (1), (2), (3) is not present directly in the NLP problem, rather it is solved by numerical integration in each function evaluation step of the NLP solver to determine $\boldsymbol{x}(\boldsymbol{c}, \boldsymbol{p}, t)$ for given $\boldsymbol{c}$ and $\boldsymbol{p}$ and therefore present implicitly. Algorithms for the solution of this NLP, typically SQP methods, require gradient information of the constraints and the objective function with respect to the decision variables. There are several possibilities to obtain these gradients. Here, we consider the explicit solution of the arising sensitivity equation systems, which is the method of choice in most sequential approach dynamic optimization algorithms.

The sensitivity systems can be solved by numerical integration together with the original DAE system. Efficient algorithms are available for this purpose, which exploit the special properties of the sensitivity system (e.g. Feehery *et al.* (1997)). Nevertheless, the by far largest amount of computation time in sequential approach dynamic optimization (typically more than 90 %) is spent on the numerical integration of states and sensitivities.

Therefore it is natural to think about applying some sort of model reduction to the DAE system. Ideally, a reduced model should give similar results to the full model in a much shorter computation time.

3 Projection based model reduction

3.1 Basic procedure

Projection methods have been suggested in a great variety in the recent literature. A generic procedure can be formulated as follows:

1. Transform the original state space into a state space better revealing the important contributions to process dynamics, i.e.

$$\boldsymbol{x} - \boldsymbol{x}^* = \mathbf{U}\boldsymbol{z}\,, \tag{9}$$

with a homomorphic transformation $\mathbf{U}$ and the transformed state vector $\boldsymbol{z} \in R^{n_x}$. The reference state $\boldsymbol{x}^*$ is often a non-zero nominal operating point.

2. To achieve order reduction we decompose the transformed space into two complementary subspaces with state vectors $\boldsymbol{z}_1 \in \mathbb{R}^m$ and $\boldsymbol{z}_2 \in \mathbb{R}^{n_x-m}$, respectively. Hence, we obtain

$$\mathbf{U}\boldsymbol{z} = \mathbf{U}_1\boldsymbol{z}_1 + \mathbf{U}_2\boldsymbol{z}_2\,, \quad \mathbf{U} = [\mathbf{U}_1, \mathbf{U}_2]\,. \tag{10}$$

We call $\boldsymbol{z}_1$ the dominant states and refer to $\boldsymbol{z}_2$ as the non-dominant states. Note that the new states are linear combinations of the original states in the linear case. After transformation and subsequent decomposition into two subsystems, we obtain

$$\begin{aligned} \dot{\boldsymbol{z}}_1 &= \mathbf{U}_1^T \boldsymbol{f}(\boldsymbol{x}^* + \mathbf{U}_1\boldsymbol{z}_1 + \mathbf{U}_2\boldsymbol{z}_2, \boldsymbol{y}, \boldsymbol{u})\,, && (11)\\ \dot{\boldsymbol{z}}_2 &= \mathbf{U}_2^T \boldsymbol{f}(\boldsymbol{x}^* + \mathbf{U}_1\boldsymbol{z}_1 + \mathbf{U}_2\boldsymbol{z}_2, \boldsymbol{y}, \boldsymbol{u})\,, && (12)\\ \boldsymbol{z}_1(0) &= \mathbf{U}_1^T(\boldsymbol{x}_0 - \boldsymbol{x}^*)\,, && (13)\\ \boldsymbol{z}_2(0) &= \mathbf{U}_2^T(\boldsymbol{x}_0 - \boldsymbol{x}^*)\,, && (14)\\ 0 &= \boldsymbol{g}(\boldsymbol{x}^* + \mathbf{U}_1\boldsymbol{z}_1 + \mathbf{U}_2\boldsymbol{z}_2, \boldsymbol{y}, \boldsymbol{u}) && (15) \end{aligned}$$

3. Finally, we have to deduce a nonlinear dynamic model for the dominant states. Here, one can apply *truncation* of the transformed state by setting $\boldsymbol{z}_2 = 0$:

$$\begin{aligned} \dot{\boldsymbol{z}}_1 &= \mathbf{U}_1^T \boldsymbol{f}(\boldsymbol{x}^* + \mathbf{U}_1\boldsymbol{z}_1, \boldsymbol{y}, \boldsymbol{u})\,, && (16)\\ \boldsymbol{z}_1(0) &= \mathbf{U}_1^T(\boldsymbol{x}_0 - \boldsymbol{x}^*)\,, && (17)\\ 0 &= \boldsymbol{g}(\boldsymbol{x}^* + \mathbf{U}_1\boldsymbol{z}_1, \boldsymbol{y}, \boldsymbol{u}) && (18) \end{aligned}$$

Alternatively, a *residualization* can be applied by setting $\dot{\boldsymbol{z}}_2 = 0$, leading to the following reduced model:

$$\begin{aligned}
\dot{\boldsymbol{z}}_1 &= \mathbf{U}_1^T \boldsymbol{f}(\boldsymbol{x}^* + \mathbf{U}_1 \boldsymbol{z}_1 + \mathbf{U}_2 \boldsymbol{z}_2, \boldsymbol{y}, \boldsymbol{u})\,, && (19)\\
0 &= \mathbf{U}_2^T \boldsymbol{f}(\boldsymbol{x}^* + \mathbf{U}_1 \boldsymbol{z}_1 + \mathbf{U}_2 \boldsymbol{z}_2, \boldsymbol{y}, \boldsymbol{u})\,. && (20)\\
\boldsymbol{z}_1(0) &= \mathbf{U}_1^T (\boldsymbol{x}_0 - \boldsymbol{x}^*)\,, && (21)\\
0 &= \boldsymbol{g}(\boldsymbol{x}^* + \mathbf{U}_1 \boldsymbol{z}_1 + \mathbf{U}_2 \boldsymbol{z}_2, \boldsymbol{y}, \boldsymbol{u}) && (22)
\end{aligned}$$

In contrast to the truncated model, the residualized model is a differential-algebraic model with the same number of equations and variables as the original model, but less differential and more algebraic equations. The residualized model has the feature of being steady-state accurate, which is not necessarily the case for a truncated model.

4. Approximate differential states $\tilde{\boldsymbol{x}}$ of the original system can be easily computed from

$$\tilde{\boldsymbol{x}} = \boldsymbol{x}^* + \mathbf{U}_1 \boldsymbol{z}_1 + \mathbf{U}_2 \boldsymbol{z}_2\,. \tag{23}$$

The variants of projection methods differ mainly in steps 1 – 3, and especially in the way, how the transformation matrix $\mathbf{U}$ is generated. In this work, we apply *proper orthogonal decomposition* (POD) for obtaining the projection.

3.2 Proper orthogonal decomposition

Proper orthogonal decomposition (POD) is a projection method, which has found many applications especially in fluid dynamics (e.g. Sirovich (1987)) or chemical vapor deposition processes (e.g. Baker and Christofides (1999)). The method comes in a number of variants. In this subsection, we will summarize one of them following the presentation of Ravindran (1999).

Starting point is a representative trajectory of (1)–(2) for a certain initial condition $\boldsymbol{x}_0$ and control $\boldsymbol{u}(t)$ defined on a finite time interval $[t_0, t_f]$. The trajectory is uniformly sampled for simplicity to form the ensemble $\mathcal{S} = \{\boldsymbol{x}(t_k) - \boldsymbol{x}^*\}_{k=1}^p = \{\Delta\boldsymbol{x}(t_k)\}_{k=1}^p$ containing p data sets of length n which are often called *snapshots*. As before, $\boldsymbol{x}^*$ is the reference which can be either a steady-state or the ensemble average of the snapshots. We are interested in a unit vector $\boldsymbol{d}$ which is in some sense close to the snapshots in $\mathcal{S}$. We may request that $\boldsymbol{d}$ is as parallel as possible to all the snapshots. This requirement leads to the optimization problem

$$\max \frac{1}{p} \sum_{k=1}^{p} \frac{(\Delta\boldsymbol{x}(t_k)^T \boldsymbol{d})^2}{\boldsymbol{d}^T \boldsymbol{d}} \quad \text{s.t.} \quad \boldsymbol{d}^T \boldsymbol{d} = 1\,. \tag{24}$$

We assume $\boldsymbol{d}$ to be a linear combination of the data, i.e.

$$\boldsymbol{d} = \sum_{k=1}^{p} w_k \Delta \boldsymbol{x}(t_k) \ , \tag{25}$$

and determine the weights w_k to solve the optimization problem (24). Solving this optimization problem is the same as finding the eigenvectors of the *correlation matrix* $\boldsymbol{N}$ with elements

$$N_{i,j} = \Delta \boldsymbol{x}(t_i)^T \Delta \boldsymbol{x}(t_j) \ . \tag{26}$$

Since this matrix is nonnegative Hermitian, it has a complete set of orthogonal eigenvectors $\{\boldsymbol{w}_1, \ldots \boldsymbol{w}_p\}$ along with a set of eigenvalues $\lambda_1 \geq \lambda_2 \cdots \geq \lambda_p$. We can now construct an orthogonal basis span$\{\boldsymbol{d}_1, \ldots, \boldsymbol{d}_p\}$ by means of (25) with

$$\boldsymbol{d}_i = \frac{1}{\sqrt{\lambda_i}} \sum_{k=1}^{p} w_{i,k} \, \Delta \boldsymbol{x}(t_k);, \quad i = 1, \ldots p \tag{27}$$

where $w_{i,k}$ denote the elements of the eigenvector $\boldsymbol{w}_i$. It can be shown that any approximation of $\boldsymbol{x}(t_k)$ in a subspace spanned by the first $p_1 < p$ basis vectors $\boldsymbol{d}_i$ maximizes the captured energy $\boldsymbol{x}(t_k)^T \boldsymbol{x}(t_k)$ of the data set. Due to this property, we may just use a reduced basis span$\{\boldsymbol{d}_1, \ldots, \boldsymbol{d}_{p_1}\}$ with $p_1 \ll p$ to obtain sufficient approximation quality. The value of p_1 is determined after some experimentation. The ratio

$$\kappa = \frac{\sum_{k=1}^{p_1} \lambda_k}{\sum_{k=1}^{p} \lambda_k} \tag{28}$$

indicates the percentage of energy contained in the first p_1 basis vectors. Obviously, this ratio should be close to unity. Note that we therefore do not have to match the number of snapshots (or basis vectors) p to the dimension n of the dynamic system. Often, we want to use $p \ll n$ for convenience, if very large-scale systems are considered, which, for example, may arise after discretization of a distributed parameter system.

There are at least two common ways of determining the basis vectors. Banerjee and Arkun (1998) employ a singular value decomposition and construct the basis from the left singular vectors of $\boldsymbol{N}$. An alternative approach does not rely on the correlation matrix $\boldsymbol{N}$ but on the $n \times p$ *snapshot matrix*

$$\boldsymbol{X} = [\Delta \boldsymbol{x}(t_1), \Delta \boldsymbol{x}(t_2), \ldots \Delta \boldsymbol{x}(t_p)] \tag{29}$$

the columns of which are the snapshots $\Delta \boldsymbol{x}(t_k)$ at t_k (Aling *et al.*, 1996; Shvartsman and Kevrekidis, 1998). Again, the basis is formed by those

$p_1 \ll p$ left singular vectors of $\boldsymbol{X}$ which are associated with the largest singular values and hence capture most of the energy in the data set.

The basis constructed from the correlation or snapshot matrices gives rise to a representation of an approximate solution $\tilde{\boldsymbol{x}}$ by a linear combination of the basis vectors. Hence,

$$\tilde{\boldsymbol{x}} - \boldsymbol{x}^* = \sum_{k=1}^{p_1} z_k \, \boldsymbol{d}_k \, . \tag{30}$$

If the expansion coefficients z_k and the basis vectors $\boldsymbol{d}_k$ are collected in a vector $\boldsymbol{z}_1 = [z_1, z_2, \ldots z_{p_1}] \in R^{p_1}$ and a matrix $\mathbf{U}_1 = [\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots \boldsymbol{d}_{p_1}] \in R^{n \times p_1}$ we can rewrite this equation as

$$\tilde{\boldsymbol{x}} = \boldsymbol{x}^* + \mathbf{U}_1 \boldsymbol{z}_1 \tag{31}$$

which has the same structure as Eq. (23) in case of truncating the non-dominant states. The reduced model is

$$\begin{aligned} \dot{\boldsymbol{z}}_1 &= \boldsymbol{U}_1^T \boldsymbol{f}(\boldsymbol{x}^* + \boldsymbol{U}_1 \boldsymbol{z}_1, \boldsymbol{u}) \, , && (32) \\ \boldsymbol{z}_1(0) &= \boldsymbol{U}_1^T (\boldsymbol{x}_0 - \boldsymbol{x}^*) \, , && (33) \end{aligned}$$

which has exactly the same appearance as the truncated model (16)–(17) resulting from model reduction by projection.

Alternatively, the full basis spanned by $p < n$ vectors can be employed by summing to p instead to p_1 in the approximation (30). This approach would result in a model structure completely analogous to the system (19)– (21). Residualization by setting $\dot{\boldsymbol{z}}_2$ to zero can then be used to potentially reduce the computational effort.

The choice of the initial $\boldsymbol{u}(t)$ plays an important role in POD. The frequency content of the input signal determines, which dynamics are excited and therefore identified by the POD.

4 Implementation

If a projection technique is applied to the process model, the dynamic optimization problem P1 to be solved turns into

$$\min_{\boldsymbol{u}(t), \boldsymbol{p}, t_f} \; \Phi\left(\boldsymbol{x}\left(t_f\right), \boldsymbol{y}\left(t_f\right)\right) \tag{P3}$$

$$\begin{aligned}
\text{s.t.}\quad \dot{\boldsymbol{z}}_1 &= \mathbf{U}_1^T \boldsymbol{f}(\boldsymbol{x}^* + \mathbf{U}_1 \boldsymbol{z}_1, \boldsymbol{y}, \boldsymbol{u})\,, && (34)\\
0 &= \boldsymbol{g}(\boldsymbol{x}^* + \mathbf{U}_1 \boldsymbol{z}_1, \boldsymbol{y}, \boldsymbol{u})\,, && (35)\\
\boldsymbol{z}_1(0) &= \mathbf{U}_1^T(\boldsymbol{x}_0 - \boldsymbol{x}^*)\,, && (36)\\
0 &\geq \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{u}, \boldsymbol{p}, t)\,, \quad t \in [t_0, t_f]\,, && (37)\\
0 &\geq \boldsymbol{e}\left(\boldsymbol{x}\left(t_f\right), \boldsymbol{y}\left(t_f\right)\right)\,, && (38)
\end{aligned}$$

in case of truncation, and

$$\min_{\boldsymbol{u}(t), \boldsymbol{p}, t_f} \Phi\left(\boldsymbol{x}\left(t_f\right), \boldsymbol{y}\left(t_f\right)\right) \qquad \text{(P4)}$$

$$\begin{aligned}
\dot{\boldsymbol{z}}_1 &= \mathbf{U}_1^T \boldsymbol{f}(\boldsymbol{x}^* + \mathbf{U}_1 \boldsymbol{z}_1 + \mathbf{U}_2 \boldsymbol{z}_2, \boldsymbol{y}, \boldsymbol{u})\,, && (39)\\
0 &= \mathbf{U}_2^T \boldsymbol{f}(\boldsymbol{x}^* + \mathbf{U}_1 \boldsymbol{z}_1 + \mathbf{U}_2 \boldsymbol{z}_2, \boldsymbol{y}, \boldsymbol{u})\,, && (40)\\
0 &= \boldsymbol{g}(\boldsymbol{x}^* + \mathbf{U}_1 \boldsymbol{z}_1 + \mathbf{U}_2 \boldsymbol{z}_2, \boldsymbol{y}, \boldsymbol{u})\,, && (41)\\
\boldsymbol{z}_1(0) &= \mathbf{U}_1^T(\boldsymbol{x}_0 - \boldsymbol{x}^*)\,, && (42)\\
0 &\geq \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{u}, \boldsymbol{p}, t)\,, \quad t \in [t_0, t_f]\,, && (43)\\
0 &\geq \boldsymbol{e}\left(\boldsymbol{x}\left(t_f\right), \boldsymbol{y}\left(t_f\right)\right)\,, && (44)
\end{aligned}$$

if residualization is chosen.

As dynamic optimizer, the tool *ADOPT* (Schlegel *et al.*, 2001) is used. It is an implementation of a sequential approach dynamic optimizer and is interfaced to the dynamic modeling and simulation package *gPROMS* (gPROMS, 2002), which is used as the model source. For convenience, both, the original model as well as the projected model have been implemented in gPROMS. In concrete terms, the gPROMS model looks as follows:

$$\begin{aligned}
\boldsymbol{x_d} &= \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{u}, \boldsymbol{p}, t)\,, && (45)\\
0 &= \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{u}, \boldsymbol{p}, t)\,, && (46)\\
\boldsymbol{x} &= \mathbf{U}\boldsymbol{z} + \boldsymbol{x}^*\,, && (47)\\
\dot{\boldsymbol{z}} &= \mathbf{U}^T \boldsymbol{x}_d\,, && (48)\\
0 &= \boldsymbol{z}(t_0) - \mathbf{U}^T(\boldsymbol{x}_0 - \boldsymbol{x}^*)\,. && (49)
\end{aligned}$$

Truncation can be realized by replacing $\mathbf{U}$ by $\mathbf{U}_1$ in above equations, which corresponds to replacing $\boldsymbol{z}$ by $\boldsymbol{z}_1$. For residualization, the equations

$$0 = \mathbf{U}_2 \boldsymbol{x}_d \qquad (50)$$

have to be added to the model. Note that the equations (47)–(48) increase the size of the model without containing physical information. They are just added for the convenience of retaining the original model (45)–(46) in gPROMS format. Therefore, in the subsequent case study, we refer to the original, i.e. un-projected model by setting $\mathbf{U} = \mathbf{I}$ (identity matrix) in above equations, in order to enable a fair comparison of original and projected models in terms of number of equations and variables.

5 Case study

As a test case, we consider a chemical plant consisting of a reactor, a distillation column and a recycle. The flowsheet of the plant is depicted in Figure 1. Initially, the plant is filled with pure component A, which undergoes the irreversible reaction $A \rightarrow B$.

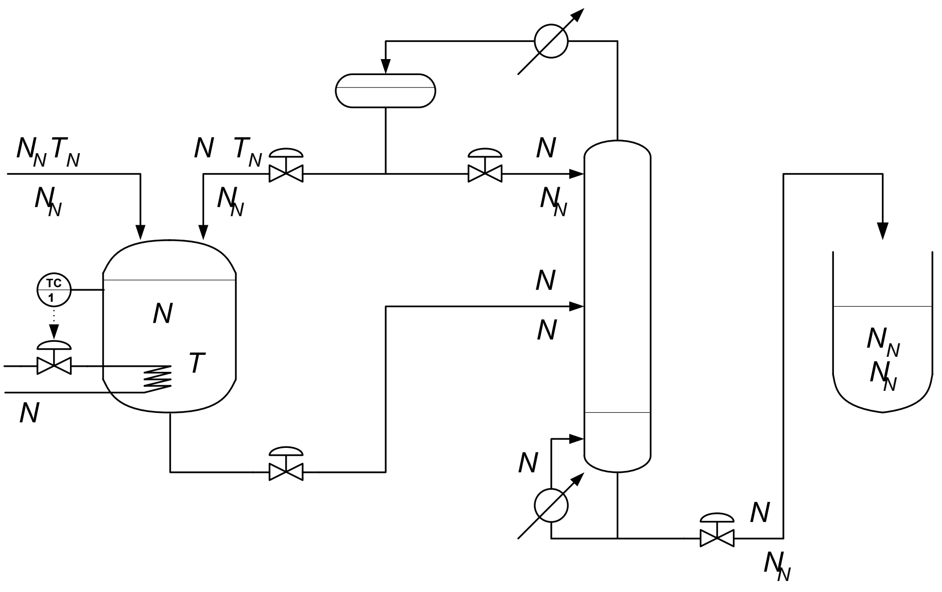


Figure 1: Flowsheet of the case study plant.

The plant is operated in semi-batch mode. The dynamic optimization problem considered in the case study is to minimize the time of operation t_f, which is required for yielding a pre-specified amount of product N_p with a defined composition x_p. In concrete terms, we employ the two endpoint constraints

$$N_p(t_f) = 2000\,\text{mol}, \tag{51}$$

$$x_p(t_f) = 0.01\,. \tag{52}$$

Additionally, we impose a path constraint on the reactor hold-up:

$$350\text{ mol} \leq N(t) \leq 600\text{ mol}. \tag{53}$$

The operational degree of freedom, whose optimal profile over time is to be found by the dynamic optimization, is the reflux ratio $R = L/D$. The distillation column has 41 trays. The original model of this plant, in the form of

Eqns. (1)-(2) has $n_x = 47$ differential and $n_y = 48$ algebraic variables. Besides the composition on the 41 trays, the 47 differential states comprise the composition, hold-up and temperature in the reactor as well as the product amount and concentration, and one additional state representing the process time.

5.1 Nominal solution

As nominal solution, we consider the optimal solution found by performing a dynamic optimization run with the full model using a discretization of R according to Eq. (6) with 20 piecewise constant elements. The optimal profile of R is shown in Figure 2. The profiles of the constrained quantities x_p and N are depicted in Figure 3 (a) and (b). Note that the reflux is first at the lower bound for quality and then raised for maximum product amount, but limited by the path constraint on N.

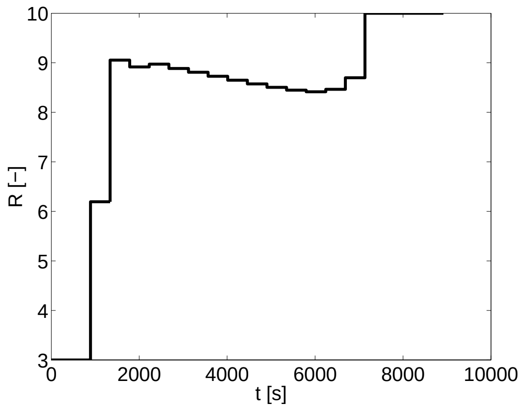
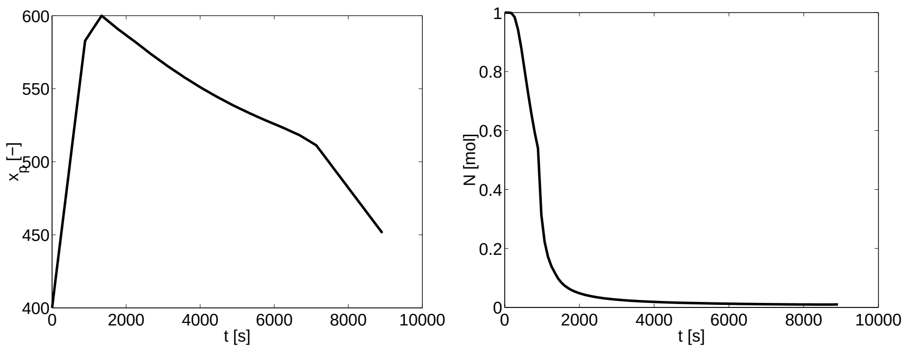


Figure 2: Optimal trajectory for reflux ratio R (nominal case).

The optimal solution was obtained by employing the software setup as described in Section 4. The integration tolerance for the numerical integrator has been set to 10^{-6}, the optimality and feasibility of the NLP solver to 10^{-5}.

The minimum final time (objective function value), which can be achieved by meeting all constraints is 8918.6 seconds. The solution took 169 CPU seconds on a 1.2 GHz PC. The NLP solver required 31 major iterations, and in total 73 numerical integrations (for states and sensitivities) were needed for solving this problem.

As mentioned before, the nominal model considered here includes a projection with the unity matrix. In the following studies with projected models



(a) (b)

Figure 3: Constrained quantities reactor hold-up N and product quality x_p (nominal case).

the projection will be solely applied to the column part of the model. Therefore, the vector z contains at most $n_z = 41$ elements, corresponding to the 41 trays in the column. For this reason, an extra 82 equations of type (47)-(48) are added to the model, so that the nominal model final comprises 47 differential and 130 algebraic equations and variables. As a basis for subsequent comparisons, the sparsity pattern of the Jacobian matrix of the full model is shown in Figure 4. It shows 502 nonzero entries for 177 equations and variables.

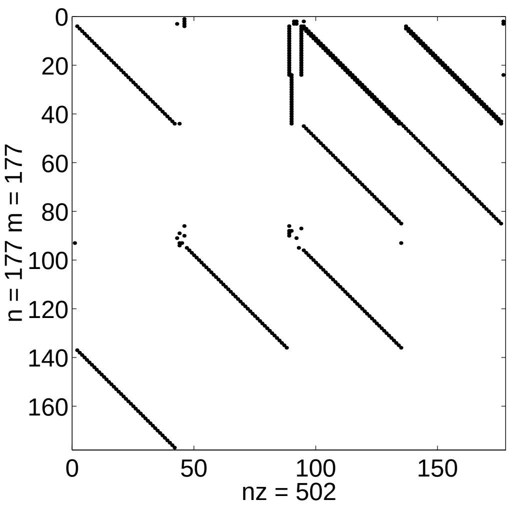


Figure 4: Sparsity pattern of the Jacobian (nominal case).

5.2 Solutions with projected models

Consecutively, the same problem has been solved by applying different kinds of projections to the model. As mentioned before, only the column model has been used in the projection, therefore 41 of the 47 differential states are effected by this.

The snapshot matrix for generating the POD projection matrix $\mathbf{U}$ has been generated based on a simulation with the optimal nominal trajectory for $\boldsymbol{u}(t)$ according to Figure 2.

In the following we show, how truncation and residualization with different levels of order reduction effect the numerical and computational performance of the optimization with the projected model.

5.2.1 Truncation

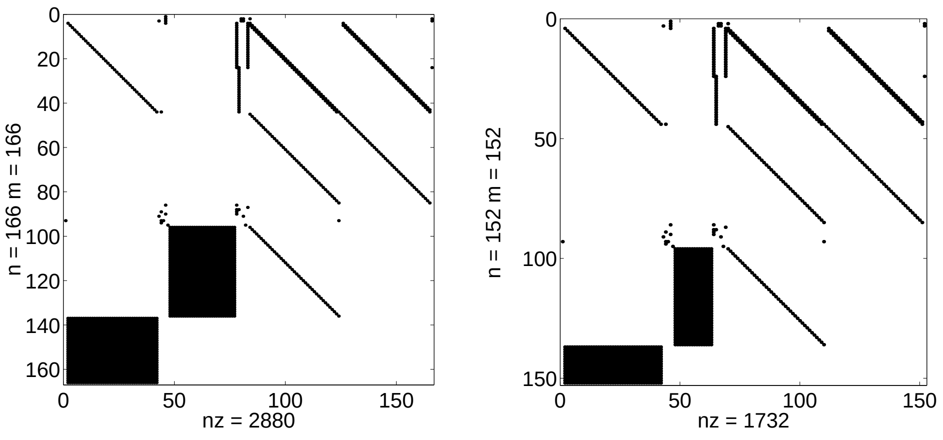
First, we investigate the effect of applying truncation to the projected model. For this purpose, five different levels of reduction have been explored, which are $n_z = \{30, 20, 18, 16, 8\}$. The same dynamic optimization problem has been solved with increasingly reduced models. The computational statistics are summarized in Table 1.

	model order	number of iterations	number of integrations	obj. fun. value	CPU time [s]
nominal	41	31	73	8918.618	169.7
truncation	30	31	73	8918.622	758.1
	20	30	68	8921.933	562.4
	18	27	54	8889.999	397.6
	16	25	46	8958.987	317.5
	8	*problem infeasible*			
residuali-zation	30	31	73	8918.618	975.6
	16	32	77	8918.614	1083.2
	8	31	74	8916.406	1025.8

Table 1: Computational statistics: comparison of optimization runs with the nominal model and with truncated and residualized models of different size.

It can be seen, that the case $n_z = 30$ gives almost exactly the same solution as the nominal model. However, with 758 CPU seconds the computation time is more than four times longer than with the nominal model. The reason for this behavior becomes obvious by looking at Figure 5 (a), which shows the Jacobian pattern for this truncated model. As compared to the pattern of the nominal model there are less equations and variables, but much more

Jacobian entries. This is due to the full blocks in the lower right corner of the matrix, which are the result of the projection equations (47)-(48), since **U** here is not the unity matrix anymore, but a dense projection matrix.



(a) $n_z = 30$ (b) $n_z = 16$

Figure 5: Sparsity pattern of the Jacobian (truncated models).

This observation highlights one major bottleneck of the projection techniques. The reduced model is smaller in dimension, but much more dense than the original model. Sparsity structures, which are common e.g. in chemical engineering models, are partly destroyed. This has a bad impact on the computational performance, if sparse solvers are used for solving the linear algebra problems in the numerical integration algorithms. In our case, such a sparse solver is used for this purpose in order to enable the solution of large-scale systems.

This effect becomes less severe, the more the model is reduced. The matrix fill-up becomes smaller, as can be seen e.g. for the case $n_z = 16$ in Figure 5 (b). Table 1 shows a computation time of 317 CPU seconds, which is much faster than the case $n_z = 30$. However, even this is almost the double of the reference computation time for the nominal solution. As a rough estimate, it can be stated that the computational effort with sparse linear algebra is proportional to the number of nonzero elements in the Jacobian pattern (and to the number of integrations needed for the optimization). Consequently, even for the case $n_z = 16$, the computation time has to be much higher than for the nominal case, because there are still more Jacobian entries.

The quality of the numerical solution is the second important aspect

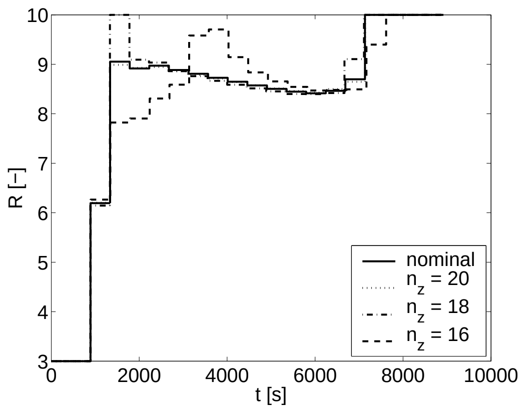


Figure 6: Optimal trajectory of reflux ratio R (truncated models of different order).

besides the computational performance, when investigating reduced models. As mentioned before, the truncated model with 30 states in the column model yields the same solution. This, however, is not the case anymore, if the model is reduced further. This becomes clear by looking at the Figures 6 and 7. They show the optimal trajectory for the reflux ratio and the constrained quantities found by optimizations with the different reduced models, and the nominal solution for comparison. The profiles in Figure 7 have been obtained by simulating the corresponding optimal trajectories in the original model.

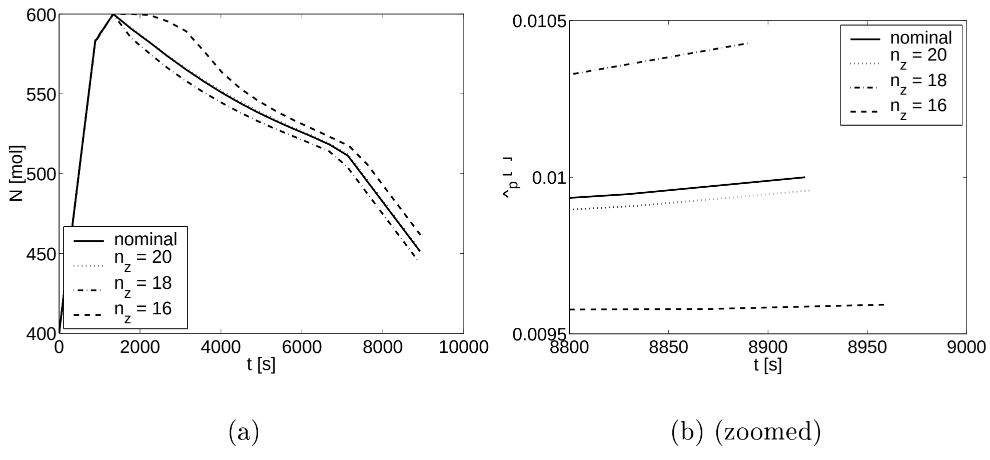


Figure 7: Constrained quantities reactor hold-up N and product quality x_p (truncated models of different order).

It is interesting to note that the optimal profiles deviate the more from the nominal case, the larger the amount of reduction is. Most important is the fact, that the endpoint constraint on the product quality (Figure 7 (b)) is not met. The reason for this lies in the fact, that the reduced models by definition just give approximations of the nominal process behavior. In particular, the truncated model shows a steady-state error, which leads to the deviation in the trajectory of x_p.

For the case $n_z = 8$, the dynamic optimizer even could not find a feasible solution for the problem at all.

5.2.2 Residualization

As an alternative to truncation, also residualization of the reduced model has been considered. As stated above, this does not really lead to a reduced model, rather some differential states are replaced by algebraic ones, keeping the overall size of the model constant. Therefore, the Jacobian pattern does not change with the level of reduction n_z. Figure 8 shows this pattern for a residualized model.

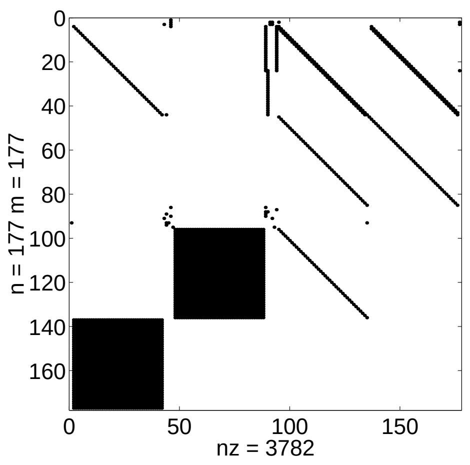


Figure 8: Sparsity pattern of the Jacobian (residualized model).

Due to this fact, the fill-up of the Jacobian caused by this type of model "reduction" is even more severe than in the case of truncation. Consequently, the computation times with residualized models, reported in Table 1 for $n_z = \{30, 16, 8\}$, are all quite closely together and significantly higher than for the truncated models.

However, a different conclusion can be drawn with respect to the solution quality. For all residualized models, even for $n_z = 8$, the obtained solution was almost identically to the nominal case. For this reason, no solution plots for these cases are given here.

6 Conclusions

In this paper, we have investigated the applicability of projection based model reduction to dynamic optimization in chemical engineering problems. Proper orthogonal decomposition (POD) has been chosen as the method for obtaining the required projection matrix. By means of a case study, the effect of using projected models with respect to computational performance and solution quality has been studied.

From our case study we can conclude, that the benefit of using reduced projected models in dynamic optimization applications is limited, at least for lumped parameter systems. The required computation times are much higher than for optimizations with un-reduced models. This is caused by the matrix fill-up, which partly destroys the problem inherent structure. Real-world applications will always show such a structure. Note, that this is principal problem, which holds for all kinds of projection methods, not only for POD.

Besides the computation time issue, truncated models additionally lack of sufficient accuracy, leading to a loss in solution quality, which grows with the degree of reduction. Residualized models do not show this problem, but require even higher computation times.

References

Afanasiev, K. and M. Hinze (2001). Adaptive control of a wake flow using proper orthogonal decomposition. In: *Shape Optimization & Optimal Design, Lecture Notes in Pure and Applied Mathematics 216*. pp. 317–332. 216. Marcel Dekker.

Aling, H., R.L. Kosut, A. Emami-Naeini and J.L. Ebert (1996). Nonlinear model reduction with application to rapid thermal processing. In: *Proceedings of the 35th Conf. of Decision and Control.* Kobe. pp. 4305–4310.

Baker, J. and P.D. Christofides (1999). Output feedback control of parabolic pde systems with nonlinear spatial differential operators. *Ind. Eng. Chem. Res.* **38**, 4372–4380.

Banerjee, A. and Y. Arkun (1998). Model predictive control of plant transitions using a new identification technique for interpolating nonlinear models. *J. Proc. Control* **8**(5/6), 441–457.

Feehery, W., J. Tolsma and P.I. Barton (1997). Efficient sensitivity analysis of large–scale differential–algebraic systems. *Appl. Numer. Math.* **25**, 41–54.

gPROMS (2002). *gPROMS User Guide (Release 2.1.1)*. Process Systems Enterprise Ltd.. London, UK.

Kraft, D. (1985). On converting optimal control problems into nonlinear programming problems. *Comput. Math. Prog.* **15**, 261–280.

Marquardt, W. (2001). Nonlinear model reduction for optimization based control of transient chemical processes. In: *Chemical Process Control-6, Preprints*. Tuscon, Arizona. pp. 30–60.

Ravindran, S.S. (1999). Proper orthogonal decomposition in optimal control of fluids. Technical Report NASA/TM-1999-209113. National Aeronautics and Space Administration NASA. Hampton, Virginia (USA).

Schlegel, M., Th. Binder, A. Cruse, J. Oldenburg and W. Marquardt (2001). Component–based implementation of a dynamic optimization algorithm using adaptive parameterization. In: *European Symposium on Computer Aided Process Engineering – 11* (R. Gani and S.B. Jørgensen, Eds.). Elsevier. pp. 1071–1076.

Shvartsman, Stanislav Y. and Ioannis G. Kevrekidis (1998). Nonlinear model reduction for control of distributed systems: A computer-assisted study. *AIChE J.* **44**(7), 1579–1595.

Sirovich, L. (1987). Turbulence and the dynamics of coherent structures: I, ii and iii. *Quart. Appl. Math.* **XLV**(3), 561.