Automatic Model Reduction of Differential Algebraic Systems by Proper Orthogonal Decomposition

Dmytro Khlopov[a,*], Michael Mangold[b]

[a] *Max Planck Institute for Dynamics of Complex Technical Systems Sandtorstraße 1, 39106 Magdeburg, Germany*
[b] *Technische Hochschule Bingen, Berlinstr. 109, 55411 Bingen*

Abstract

Proper Orthogonal Decomposition (POD) is an attractive way to obtain nonlinear low-dimensional models. This article reports on the automatization of the mentioned reduction method. An automatic procedure for the reduction of differential algebraic systems is presented, which is implemented in the modeling and simulation environment ProMoT/Diana. The software tool has been applied to a nonlinear heat conduction model and a continuous fluidized bed crystallizer model. The automatically generated reduced models are significantly smaller than the reference models, while the loss of accuracy is negligible.

Keywords: nonlinear model reduction, proper orthogonal decomposition, empirical interpolation, computer aided modeling, differential algebraic systems

1. Introduction

Many modern mathematical models of real-life processes impose difficulties when it comes to their numerical solution. This holds especially for models represented by nonlinear distributed parameter systems, which are frequent in engineering. Usually, for the numerical solution of distributed parameter systems the original system of infinite order is approximated by one with a finite

*Corresponding author
Email addresses: khlopov@mpi-magdeburg.mpg.de (Dmytro Khlopov), mangold@mpi-magdeburg.mpg.de (Michael Mangold)

system order by a semi-discretization, which results in a system of differential algebraic equations. The resulting number of degrees of freedom is usually very high and makes the use of the discretized model inconvenient for model-based process design, process control and optimization [1]. Thus there is a need for reduced models. Through model reduction, a small system with reduced number of equations is derived. The numerical solution of reduced models should be much easier and faster than the solution of the original problem. On the other hand, the reduced model should be able to reproduce the system behavior with sufficient accuracy in the relevant window of operation conditions and in the relevant range of system parameters.

Various methods for nonlinear and linear model reduction have been proposed, particularly in the areas of electrical and mechanical engineering, control design and computational fluid dynamics. Some of them are based on physical simplifications like assumption of perfect mixing, introduction of compartments, equilibrium assumptions, etc. This approach requires physical insight of the modeler and hence is hard to automatize. Another successful approach, which may also be considered as a physical model reduction method, is based on nonlinear wave propagation theory [2, 3]. It produces reduced model by approximation of the spatially distributed solution by profile with a given shape. As in the previous case, this method requires physical process understanding from the user and can be applied only for special systems. The generalized method of moments [4, 5] is a widely used mathematical reduction technique for population balance equations. In this case, the reduced model does not preserve full information on spatial profile. Another mathematical possibility to obtain reduced models is to separate fast and slow subsystems. Slow manifold approximation [6] requires complicated symbolic operations, which impose difficulties on the automatization of this method. To sum up, widely used methods for nonlinear model reduction require experienced user; automatic application and integration in a simulation tool is a difficult and challenging task, which has hardly been attempted to our knowledge. On the other hand, there are linear model reduction techniques like balanced truncation [7, 8], which are applicable to high order systems and can be automatized quite easily. However, the resulting linear reduced models are only valid locally and not able to capture nonlinear properties of the original system.

In this work Proper Orthogonal Decomposition (POD) [9, 10, 11, 12] is used for the development of an automatic procedure for model reduction. This method has been successfully applied for numerous problems in the fields of fluid dynamics, optimal control, and for population balance systems like crystallizers [13], [14], and granulators [15]. To put it in other words, the model reduction by POD is a proven approach. Nevertheless, applying model reduction by POD manually to complex engineering models is a challenging and tedious task. The idea of this work is to provide a software environment that performs the model reduction by POD automatically with minimal additional input from the user.

The work is structured as follows. Section 2 discusses the model reduction method. Technical details of the developed software tool for automatic model reduction are described in Section 3. Section 4 shows the developed software tool in action by applying it to two test models: a nonlinear heat conductor and a continuous fluidized bed crystallizer.

2. Mathematical model reduction method

2.1. Reference model representation

Before applying a reduction procedure to the reference model, it has to be transformed into a spatially discretized form by applying the method of lines [16]. Discretization results in a system of differential algebraic equations, which may be written as

$$B\frac{dx}{dt}(t) = f(x(t)) = Ax(t) + c + g(x(t)), \tag{1}$$

where $x(t)$ is the discretized state vector, B and A are the system matrices, where B may be singular, c is a constant vector, and $g(x(t))$ is a function that comprises the nonlinearities of the system.

2.2. POD method

In this work the Proper Orthogonal Decomposition method [9, 10, 11, 12] is used for the development of an automatic procedure for the model reduction. The basic idea of this method is to approximate the model solution by a linear combination of time independent basis functions weighted by time dependent coefficients. The basis functions are constructed from numerical simulation results of the detailed reference model. Applying Galerkin's method of weighted residuals produces the reduced model equations. At this point the offline phase of the reduction procedure ends, which can be extremely computationally intensive depending on the complexity of the reference model. But these efforts pay off in the second fast and cheap step, the online phase. In the online phase only a differential algebraic system of low order has to be solved.

As a starting point of the offline phase, the detailed reference model has to be solved numerically. Snapshots for the model states $x(t_1), x(t_2), ...$ and for the right-hand sides $f(t_1), f(t_2), ...$ are stored in matrices $X = (x(t_1), x(t_2), ...)$ and $F = (f(t_1), f(t_2), ...)$, correspondingly.

A reduced basis for the snapshots vectors is constructed from the singular value decomposition (SVD) of X with

$$X = U\Sigma V^T, \tag{2}$$

where U is a unitary matrix containing the left singular vectors or POD modes, which are already ordered by the singular values, V^T is a unitary matrix containing the right singular vectors and Σ is a pseudo-diagonal matrix with the descending singular values as entries. The singular values are a measure for the truncation error and hence determine the order of the reduced model.

Consequently the basis vectors for the orthogonal projection are taken as

$$\Psi_i^x = U_i, i = 1, ..., N^x, \tag{3}$$

where U_i denotes the i th column of U, and N^x is the dimension of the reduced basis and correspondingly the order of the resulting reduced model.

The state vector $x(t)$ is approximated by the following expression:

$$x(t) \approx \Psi^x \phi^x(t), \tag{4}$$

where $\Psi^x = (\Psi_1^x, ..., \Psi_{N^x}^x)$, and $\phi^x(t)$ is the coefficient vector of the reduced basis and the state of the reduced model.

In order to obtain equations for $\phi^x(t)$, the approximation for the state vector (4) is inserted into the discretized differential equation (1). To make the projection of the residuals on the reduced basis vanish, Galerkin's method of weighted residuals is applied, which leads to

$$\underbrace{\Psi^{xT} B \Psi^x}_{=:B_{red}} \frac{d\phi^x}{dt}(t) = \underbrace{\Psi^{xT} A \Psi^x}_{=:A_{red}} \phi^x(t) + \underbrace{\Psi^{xT} c}_{=:c_{red}} + \Psi^{xT} g(\Psi^x \phi^x(t)) \tag{5}$$

The matrices B_{red}, A_{red} and the vector c_{red} from (5) have to be evaluated only once for a fixed reduced basis, because they do not depend on the reduced state vector $\phi^x(t)$.

2.3. Empirical Interpolation

The nonlinear term on the right-hand side of (5) still depends on the high order state vector of the reference model, bringing additional complexity during the runtime of the reduced model. Clearly, more efficient approaches are needed. There are several methods in literature on how to handle the nonlinear terms in the context of POD model reduction effectively, whose basic idea is to approximate also the nonlinearities by basis vectors constructed from snapshots [17, 18].

In this work the Empirical Interpolation method (EI) [17] is used. Its algorithm uses specially selected interpolation indices to specify an interpolation-based projection instead of a more costly orthogonal projection. Thus, the nonlinearity is projected onto a subspace spanned by a basis, which approximates the solution space of the nonlinearity. The basis vectors $\Psi_i^g, i = 1, ..., N^g$ for the available snapshots $g(t_i) = f(t_i) - (Ax(t_i) + c)$ are constructed by the iterative procedure in [17]. During runtime of the reduced model, the nonlinearity is approximated as a linear combination of time independent basis functions

$\Psi^g = (\Psi_1^g, ..., \Psi_{N^g}^g)$ weighted by time dependent coefficients $\phi^g(t)$, which follow from the linear equation system

$$\underbrace{\Psi_k^g}_{=:D_{red}} \phi^g(t) = f_k(x(t)) - \big(\underbrace{A_k \Psi^x}_{=:E_{red}} \phi^x(t) + c_k\big) \tag{6}$$

The indices k from (6) are the output of the EI algorithm described in [17] and chosen in such a way that the approximation error is minimized. This is achieved by placing new interpolation points where the residual between the input basis and its approximation by former interpolation points is largest.

In summary, the resulting reduced model consists of the differential equations

$$\underbrace{\Psi^{xT} B \Psi^x}_{=:B_{red}} \frac{d\phi^x}{dt}(t) = \underbrace{\Psi^{xT} A \Psi^x}_{=:A_{red}} \phi^x(t) + \underbrace{\Psi^{xT} c}_{=:c_{red}} + \underbrace{\Psi^{xT} \Psi^g}_{=:G_{red}} \phi^g(t) \tag{7}$$

in combination with the linear algebraic equations (6). To sum up, the offline phase comprises the computation of snapshots $x(t_i)$ and $f(t_i)$ by numerical solution of the reference model, the generation of reduced basis Ψ^n and Ψ^g, and evaluation of the numerical data like B_{red}, A_{red}, c_{red}, G_{red}, D_{red}, E_{red}. The online phase is the solution of the N^x differential equations (7) and the N^g algebraic equations (6), which requires much less effort compared to the reference model.

The main task of the model reduction tool is to construct the reduced model equations (6) and (7) in symbolic form from an arbitrarily structured reference model.

3. Software implementation

The automatic procedure for the model reduction is implemented in the modeling and simulation environment ProMoT/Diana [19]. ProMoT is a modeling tool written in Common Lisp with a graphical user interface written in Java [20]. ProMoT supports the structured implementation of dynamic models described by systems of nonlinear implicit differential algebraic equations. ProMoT itself is a purely symbolic modeling tool and hence has no restriction with respect to numerical properties of the models. On the ProMoT level the idea is to keep the model formulation separate from numerical requirements. It translates symbolic model information into simulation code for a number of numerical simulation programs, one of which is Diana.

Diana [21] is a simulation tool for the solution and nonlinear analysis of differential algebraic systems, as they typically result from first principle modeling of chemical engineering systems and biochemical systems. The numerical core of Diana is written in C++ in order to ensure fast and efficient numerical solutions. Model equations also have to be implemented in C++ as an equation set object (ESO) using CAPE-OPEN standard interfaces. Usually, the model implementation is done automatically by ProMoT. For the numerical analysis, the modeler accesses Diana via scripts written in the scripting language Python. The advantage is that Python is more user friendly than C++ code.

The developed software tool for model reduction is a part of the ProMoT project and hence is written in Common Lisp. One uses the Diana simulation tool only as an intermediate step for the numerical solution of the reference model. The main parts of the software tool are the snapshots generator, the symbolic transformator, the generator of numerical data and the builder of reduced model. The structure of the tool is sketched in Fig. 1.

3.1. Snapshots generator

ProMoT provides a general text based modeling language MDL. In order to start the model reduction, the user has to provide the detailed reference model written in this language. Also, the user has to provide the name of a Python script which contains all information about simulation conditions like definition of model parameter values, a time range, and an output time interval for collecting snapshots. The background is that currently no systematic mathematical procedure exists for the choice of optimal conditions for generating snapshots. At this point, physical understanding of the user is required to choose simulation conditions that lead to typical spatial profiles of the solution.

The snapshots generator translates the provided reference model into the

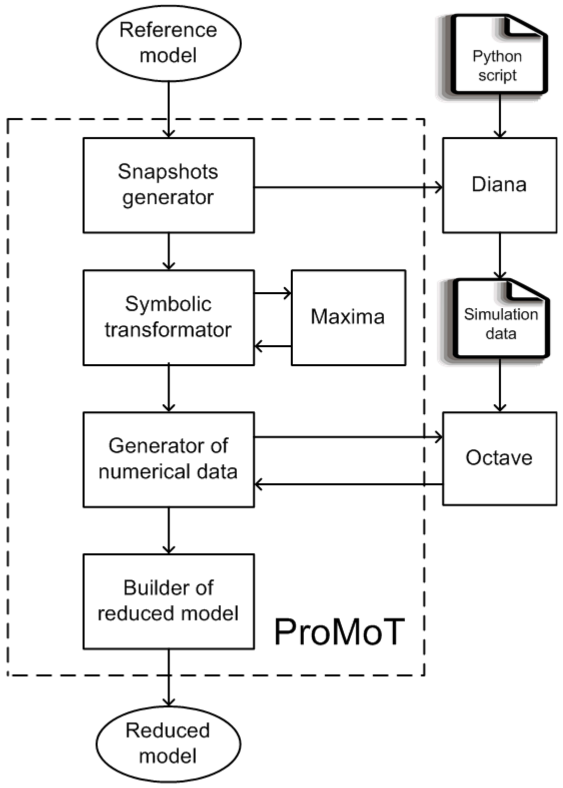


Figure 1: Structure of automatic tool for model reduction.

corresponding C++ code and runs Diana to yield snapshots. When the numerical computation is completed, Diana produces an output file, which contains the snapshots matrices $X = (x(t_1), x(t_2), ...)$ and $F = (f(t_1), f(t_2), ...)$.

150 *3.2. Symbolic transformator*

If the reference model held the required form (1), i.e. with the right-hand side explicitly separated into a linear and a nonlinear part, all the steps of reduction procedure described in Section 2 could be easily implemented using just a numerical tool like Matlab. But the reference model provided by the user usually has an arbitrary structure, which is a set of differential algebraic equations and can be written as

$$B\frac{dx}{dt}(t) = f(x(t)) \tag{8}$$

The symbolic transformation of the reference model into the appropriate form is one of the most difficult tasks in the present work. Splitting of the right-hand sides of model equations into linear and nonlinear parts boils down to the calculation of the system matrix A and the constant vector c from (1). Currently, the tool provides two approaches for accomplishment of this task.

3.2.1. Analytical Jacobian

The first approach is based on the calculation of the analytical Jacobian matrix and its use for the system matrix A. The constant vector c is filled with zero values and has no particular meaning in this approach. It is only needed to preserve the generality of model reduction procedure. The general form of an element of the Jacobian matrix can be expressed by the following equation:

$$A_{i,j} = \frac{\partial f_i(x(t))}{\partial s_j}, \tag{9}$$

where $A_{i,j}$ is an element of the system matrix at i th row and at j th column. $f_i(x(t))$ is the right-hand side of the i th differential or algebraic equation, s_j is a symbolic name of the j th state variable. The above operation has to be performed for expressions in symbolic form.

ProMoT is a symbolic tool and hence allows to treat all the modeling entities like model equations and variables in symbolic form. To perform such mathematical operations like differentiation over ProMoT symbolic expressions in a way which is similar to the traditional manual computations, the computer algebra system Maxima [22] is used. Since it is written in Common Lisp and can be called directly from Lisp code, Maxima is embedded into the ProMoT core. For convenience of use of the computer algebra system a program interface between ProMoT and Maxima has been developed. The interface allows to convert internal data structures of ProMoT into corresponding Maxima representation and vice versa.

The main advantage here is that this approach turns out to be very cheap with respect to computational time as well as allocated memory during the offline phase, because the Jacobian matrix is treated as a sparse matrix with low occupancy rate. On the other hand, this approach produced rather poor numerical results of the model reduction for the example systems considered. One reason might be that the choice of a suitable reference state x, around which the linearization is carried out, is not obvious.

3.2.2. Linear regression

The idea of the second approach is to calculate the system matrix A and the constant vector c from (1) numerically from the available matrices with snapshots $X = (x(t_1), x(t_2), ...)$ and $F = (f(t_1), f(t_2), ...)$. In order to achieve this, the following linear regression problem has to be solved

$$\arg\min_{A,c} \sum_{i=1}^{N_d} \Big[(Ax(t_i) + c - f(t_i))^{\mathsf{T}}(Ax(t_i) + c - f(t_i))\Big], \tag{10}$$

where N_d denotes the number of generated snapshots from numerical solution of reference model.

After some mathematical manipulations the system matrix A can be calculated from the following system of linear algebraic equations

$$\begin{aligned} &A\Big\{\sum_{i=1}^{N_d} x(t_i)x(t_i)^{\mathsf{T}} - \sum_{i=1}^{N_d} x(t_i)\Big[\sum_{i=1}^{N_d} x(t_i)\Big]^{\mathsf{T}}\Big\} \\ &= \sum_{i=1}^{N_d} f(t_i)x(t_i)^{\mathsf{T}} - \sum_{i=1}^{N_d} f(t_i)\Big[\sum_{i=1}^{N_d} x(t_i)\Big]^{\mathsf{T}} \end{aligned} \tag{11}$$

When the matrix A is known, the constant vector c is obtained as

$$c = \Big(\sum_{i=1}^{N_d} f(t_i) - A\sum_{i=1}^{N_d} x(t_i)\Big)\frac{1}{N_d} \tag{12}$$

Since this approach is applied to already generated numerical data and makes no assumptions on the linearization point x, it provides much better results on model linearization while keeping the nonlinearities of the reference model as small as possible. For numerical computations a specialized external software tool is used, which will be described later. The main disadvantage here is high memory usage that is needed to solve a linear equation system of very high order with dense matrices.

As a final step, the model reduction tool can easily construct the nonlinearity $g(x(t))$ of the reference model in symbolic form as

$$g(x(t)) = f(x(t)) - \big(Ax(t) + c\big). \tag{13}$$

This information combined with corresponding snapshots is used to approximate the nonlinearity of the reference model by the Empirical Interpolation method.

3.3. Generator of numerical data

Generating the equations of the reduced model requires various numerical linear algebra computations in the offline phase, in particular the solution of linear equations, singular value decomposition for computing the reduced basis, and the computation of the system matrices of the reduced model. An advanced numerical apparatus is needed to accomplish this. For these purposes it was decided to use a specialized software as an external tool. GNU Octave [23] is a high-level interpreted language primarily intended for numerical computations. Octave is freely available, easy to use, convenient for development of model reduction tool because of the ability to work in an interactive mode, but has limitations with respect to very large matrices. Due to modular structure of the reduction tool, Octave could be replaced by other linear algebra packages in the future.

To use this tool externally a program interface between ProMoT and Octave has been developed. ProMoT can send commands to Octave and receive its responses via the special input and output streams. A typical interaction scenario starts with sending some numerical data to Octave, then applying a mathematical function, and requesting an output result back to ProMoT. All conversions between ProMoT data and corresponding Octave representation are made by the developed programming interface.

3.4. Builder of reduced model

After completion of the above parts it is possible to calculate all the numerical matrices and symbolic expressions needed for the reduced model in the form of (6) and (7). The builder of the reduced model creates a new modeling file into which it writes the following system of equations

$$\begin{cases} \sum_{j=1}^{N^x} B_{redi,j} \frac{d\phi_j^x}{dt}(t) = \sum_{j=1}^{N^x} A_{redi,j}\phi_j^x(t) + c_{redi} + \sum_{j=1}^{N^g} G_{redi,j}\phi_j^g(t) \\ \sum_{j=1}^{N^g} D_{redk,j}\phi_j^g(t) = f_k(x(t)) - \big(\sum_{j=1}^{N^x} E_{redk,j}\phi_j^x(t) + c_k\big) \end{cases} \tag{14}$$

where N^x is the number of ordinary differential equations of the reduced model and N^g denotes the number of algebraic equations for handling of the nonlinearities. For the reconstruction of the states of the reference model $x(t_1), x(t_2), ...$ one has to evaluate equation (4).

4. Case studies

4.1. Heat conductor

One of the first spatially distributed chemical engineering models to which POD was applied is a nonlinear heat conduction system defined on a two-dimensional plane [10]. In [10], the model reduction was done manually, separating the system into a part with homogeneous boundary conditions and another one with inhomogeneous boundary conditions. This separation is quite tedious. Therefore, the model is a nice test example for the developed automatic model reduction tool. The system geometry is shown in Fig. 2. It is a square with a quarter removed. The system boundaries (I),(II),(III),(IV) and (V) have the boundary temperature of zero; the boundary temperature T_f of the upper boundary (VI) takes arbitrary values between 0°C and 50°C.

The governing equation of the system reads:

$$\frac{\partial T}{\partial t} = \nabla \cdot (\kappa(T)\nabla T) \tag{15}$$

with the following temperature dependence of the thermal diffusivity:

$$\kappa(T) = k_1 + k_2 T + k_3 T^2 \tag{16}$$

where k_1, k_2 and k_3 are constants with values taken from [10].

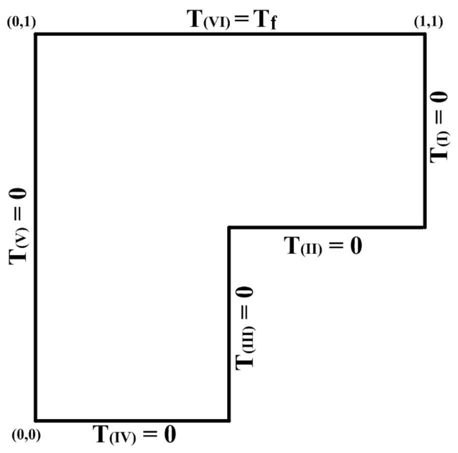


Figure 2: Heat conduction system under consideration.

4.1.1. Spatial discretization

The method of lines is used to convert the partial differential equation (15) into a set of ordinary differential equations that can be solved numerically. A finite volume scheme is applied with volume elements as shown in Fig. 3.

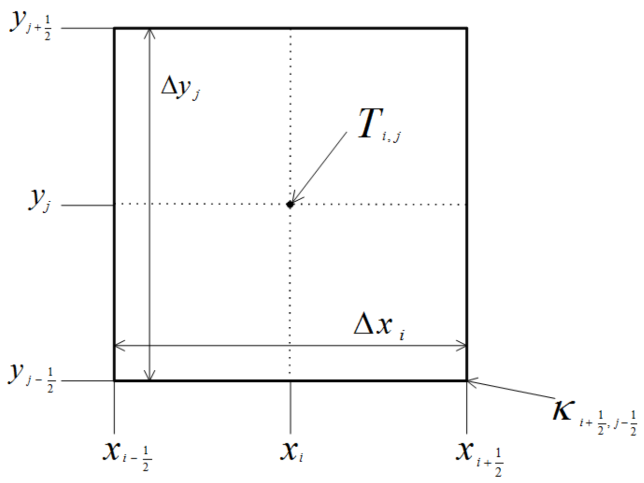


Figure 3: Volume element for the spatial discretization of the heat conduction model.

Equidistant grids are used in both the x and y directions. The discretization is straight-forward and done as follows. Firstly, Eq. (15) can be rewritten as

$$\frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\Big(\kappa(T)\frac{\partial T}{\partial x}\Big) + \frac{\partial}{\partial y}\Big(\kappa(T)\frac{\partial T}{\partial y}\Big) \tag{17}$$

Integration of Eq. (17) over a volume element gives

$$\begin{aligned}\int\limits_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}}\int\limits_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \frac{\partial T}{\partial t}\bigg|_{x,y,t} dxdy &= \int\limits_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}}\int\limits_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \frac{\partial}{\partial x}\bigg(\kappa(T)\frac{\partial T}{\partial x}\bigg|_{x,y,t}\bigg)dxdy \\ &+ \int\limits_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}}\int\limits_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \frac{\partial}{\partial y}\bigg(\kappa(T)\frac{\partial T}{\partial y}\bigg|_{x,y,t}\bigg)dxdy\end{aligned} \tag{18}$$

The integral on the left-hand side of Eq. (18) is averaged in both directions. The integrals on the right-hand side are first solved in x and y directions correspondingly and averaged in other directions:

$$\frac{dT_{i,j}}{dt}\Delta x_i \Delta y_j = \Delta y_j \left[\kappa(T)\frac{\partial T}{\partial x}\bigg|_{x,y_j,t}\right]_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} + \Delta x_i \left[\kappa(T)\frac{\partial T}{\partial y}\bigg|_{x_i,y,t}\right]_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}} \tag{19}$$

$$\begin{aligned}\frac{dT_{i,j}}{dt} &= \frac{1}{\Delta x_i}\left(\kappa_{i+\frac{1}{2},j}\frac{\partial T}{\partial x}\bigg|_{x_{i+\frac{1}{2}},y_j} - \kappa_{i-\frac{1}{2},j}\frac{\partial T}{\partial x}\bigg|_{x_{i-\frac{1}{2}},y_j}\right) \\ &+ \frac{1}{\Delta y_j}\left(\kappa_{i,j+\frac{1}{2}}\frac{\partial T}{\partial y}\bigg|_{x_i,y_{j+\frac{1}{2}}} - \kappa_{i,j-\frac{1}{2}}\frac{\partial T}{\partial y}\bigg|_{x_i,y_{j-\frac{1}{2}}}\right)\end{aligned} \tag{20}$$

Approximation of the remaining derivatives gives

$$\begin{aligned}\frac{dT_{i,j}}{dt} &= \frac{1}{\Delta x_i}\left(\kappa_{i+\frac{1}{2},j}\frac{T_{i+1,j}-T_{i,j}}{\Delta x_i} - \kappa_{i-\frac{1}{2},j}\frac{T_{i,j}-T_{i-1,j}}{\Delta x_i}\right) \\ &+ \frac{1}{\Delta y_j}\left(\kappa_{i,j+\frac{1}{2}}\frac{T_{i,j+1}-T_{i,j}}{\Delta y_j} - \kappa_{i,j-\frac{1}{2}}\frac{T_{i,j}-T_{i,j-1}}{\Delta y_j}\right)\end{aligned} \tag{21}$$

with

$$\kappa_{i+\frac{1}{2},j} = \frac{1}{2}\Big(\kappa(T_{i,j}) + \kappa(T_{i+1,j})\Big) \tag{22}$$

In this example, 120 grid points are chosen in both directions, resulting in an equation system of 10800 ordinary differential equations. It is obvious from (21) that a manual separation of the right-hand sides into a linear and a nonlinear part would be quite cumbersome. An automatization of this step, as done by the developed tool, simplifies the generation of the reduced model considerably.

4.1.2. Simulation scenario 1: single change of boundary temperature

To perform the model reduction procedure, the dynamic characteristics of the reference model have to be obtained in form of snapshots. The boundary temperature T_f is considered as system input. The dynamics of the system with respect to changes of T_f is to be analyzed. Thus, the snapshots have been collected in the following way. The initial temperature across the plate is equal to 0°C. Then the plate heats up by increasing the upper boundary temperature T_f to 50°C until a new steady state is reached. For the numerical solution by Diana the IDA solver [24] is used, which varies the time step Δt dynamically according to user defined tolerances. A new steady state is reached after 0.5 s and during this time the integrator takes 319 steps, at which the system solutions are collected as snapshots.

In order to start model reduction, the user has to provide a MDL file with the reference model written in modeling language MDL and a Python script with information about the simulation scenario. Also, the user has to specify the truncation errors indicating what fraction of the least significant basis functions is to be neglected. In the following, the truncation error for the POD modes and for the basis functions derived by empirical interpolation are designated as e_x and e_g, correspondingly. Using the linear regression method for model linearization and specifying the truncation errors $e_x = 10^{-5}$ and $e_g = 10^{-1}$, the tool generated the reduced model with only 19 ordinary differential equations and 1 algebraic equation, compared to 10800 ordinary differential equations of the reference model. Both models agree very well, as is illustrated by Fig. 4. It shows the relative total error $\|x(t) - \hat{x}(t)\|/\|x(t)\|$, where $\hat{x}(t)$ is the approximation of the reduced model. The error takes the largest value at the initial stage when there are very steep temperature gradients at the boundary and it reduces to a small value as the system reaches the steady state.

In comparison to the first approach, the second approach based on calculation of the analytical Jacobian produces less efficient reduced models. The reference state x, around which the linearization has been carried out, is cho-

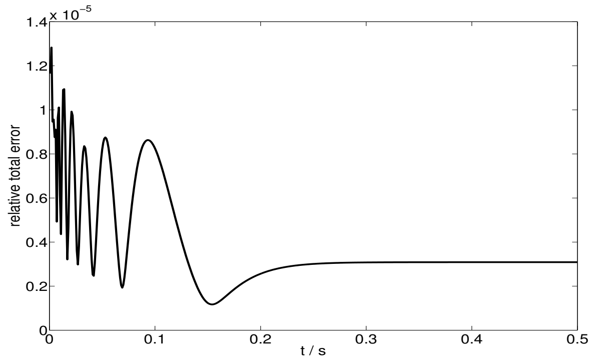


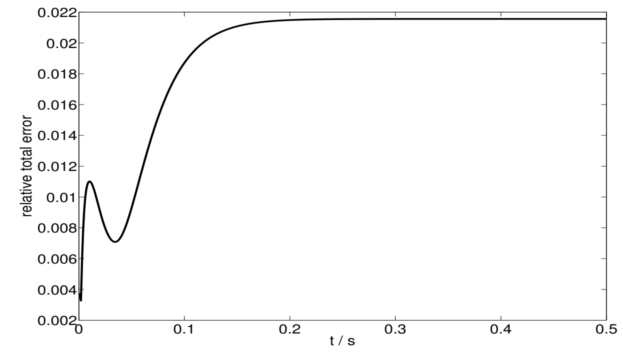
Figure 4: Simulation results of the reduced heat conductor model using linear regression.

sen as average value among all the generated snapshots. Fig. 5 shows how the truncation error e_g affects the accuracy of the reduced models.

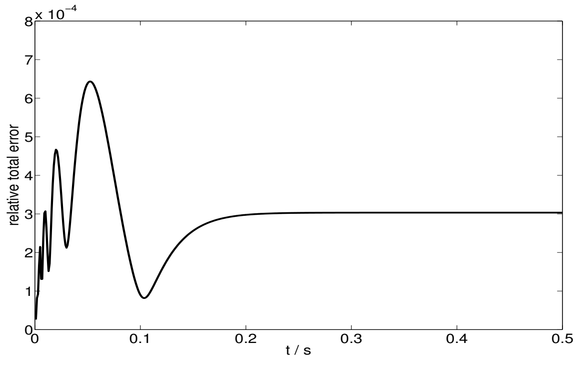
In order to achieve the same approximation accuracy using this approach, 25 algebraic equations are needed, compared to only 1 algebraic equation by using linear regression method.

4.1.3. Simulation scenario 2: randomly changing boundary temperature

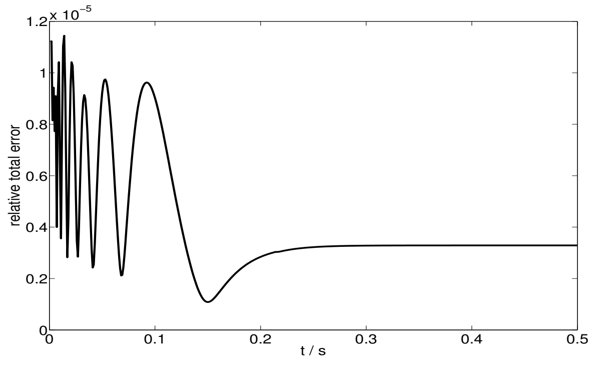
In this section the case is considered where the boundary temperature T_f changes randomly. It is known that the system parameter T_f takes values between 0°C and 50°C, thereby narrowing the variety of possible dynamic characteristics of the reference system. Trying to collect the most significant ones, the following simulation scenario has been performed for the generation of snapshots. As typical values of the boundary temperature T_f only the multiples of 10 between 0 and 50 are considered. In turn, for each of these values T_{f_i} the following actions are to be made. At first, the steady-state temperature distribution when the upper boundary temperature is set to T_{f_i} is taken as an initial temperature distribution. Next, a series of simulations are being performed from this steady state by setting the upper boundary temperature to other typical values T_{f_j} one by one except the considered one T_{f_i}. Each such simulation takes 0.01 s of simulation time. As in the previous case by using the IDA solver [24] with the varying step size Δt, Diana generated 5645 snapshots.



(a) 19 differential equations and 6 algebraic equations ($e_x = 10^{-5}$ and $e_g = 10^{-3}$).



(b) 19 differential equations and 16 algebraic equations ($e_x = 10^{-5}$ and $e_g = 10^{-5}$).



(c) 19 differential equations and 25 algebraic equations ($e_x = 10^{-5}$ and $e_g = 10^{-7}$).

Figure 5: Simulation results of the reduced heat conductor model using analytical Jacobian.

Using the linear regression method for model linearization and specifying the truncation errors $e_x = 10^{-5}$ and $e_g = 10^{-6}$, the software tool produced the reduced model with 66 ordinary differential equations and 76 algebraic equations, compared to 10800 ordinary differential equations of the reference model. The reduced model has been solved when the boundary temperature T_f changes randomly between 0°C and 50°C at every 0.01 s and compared with the exact solution. Fig. 6 shows a random variation of the boundary temperature T_f constructed by a random number generation code. Fig. 7 shows that both solutions agree very well. The error increases when a new value of the boundary temperature T_f appears and goes down towards a steady state.

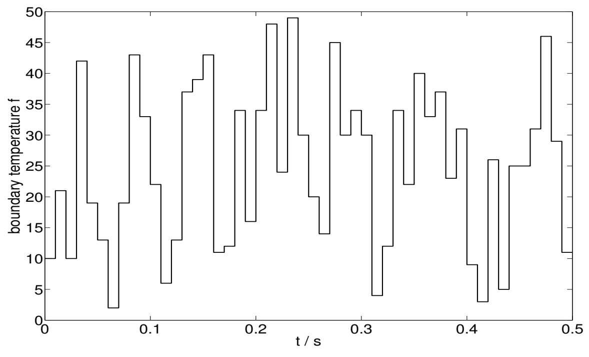


Figure 6: The temporal variation of boundary temperature f (random variation).

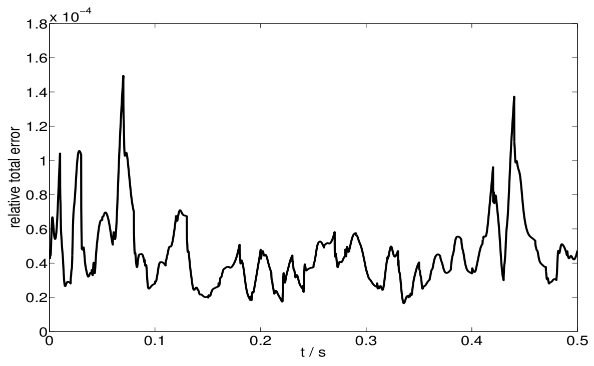


Figure 7: Simulation results of the reduced heat conductor model (simulation scenario 2).

4.2. Fluidized bed crystallizer

As the second case study a model of fluidized bed crystallizer sketched in Fig. 8 will be considered. The crystallizer aims at separation of a mixture by selectively growing crystals of one component in the mixture. Selective crystallization is achieved by providing seeding crystals of the derived species and by keeping the supersaturation of the liquid in a range that prevents nucleation of new crystals [25]. The crystallizer has the shape of a cylinder whose diameter narrows towards the crystallizer's bottom from d_{top} to d_{bottom}. An input volume flow of the fluid comes from outside and enters the bottom of the crystallizer. The fluid flow goes from bottom to top, which drags small particles upwards. Larger particles sink to the bottom due to gravity. A mixture of solvent and particles leaves the crystallizer at the top. An additional fluid flow near the crystallizer's bottom transports particles to an ultrasonic attenuator where they are broken into smaller fragments. The fragments are sent back to the crystallizer.

The reference model for this process is described in [25]. The main model assumption is that the number of particles is sufficiently high that the particle phase may be described by a particle population with a number size density $n(x, L, t)$ denoting the number of particles with size L per volume at a point x in space. Further, plug flow in axial direction and vanishing gradients in radial direction are assumed. The population balance equation of the system reads:

$$\begin{aligned} A(x)\frac{\partial n}{\partial t}\bigg|_{x,L,t} &= -\frac{\partial}{\partial x}(A(x)v_p(x,L,t)n(x,L,t)) \\ &\quad +D\frac{\partial}{\partial x}\left(A(x)\frac{\partial n}{\partial x}\bigg|_{x,L,t}\right) \\ &\quad -A(x)G(x)\frac{\partial n}{\partial L}\bigg|_{x,L,t} \\ &\quad +\dot{V}_{us}\big(n_{from_{us}}(L)-n(x,L,t)\big)\delta(x-x_{us}) \end{aligned} \tag{23}$$

with boundary conditions

$$v_p(0,L,t)n(0,L,t) - D\frac{\partial n}{\partial x}\bigg|_{0,L,t} = 0 \tag{24}$$

$$\frac{\partial n}{\partial x}\bigg|_{H,L,t} = 0 \tag{25}$$

$$n(x,0,t) = 0 \tag{26}$$

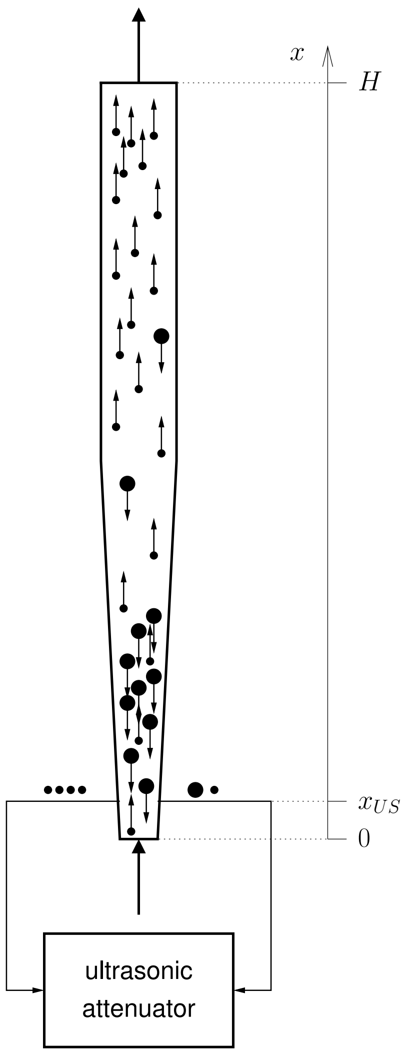


Figure 8: Considered system consisting of a crystallizer and an ultrasonic attenuator.

and initial conditions

$$n(x, L, 0) = n_0(x, L) \tag{27}$$

The first term on the right-hand side of (23) is the advective transport of particles with velocity v_p; $A(x)$ denotes the cross-sectional area of the crystallizer.

The particle velocity v_p from [25] can be expressed as follows

$$v_p(x,L) = \frac{\dot{V}}{A(x)} - v^*_{eq}(x,L), \tag{28}$$

where $\dot{V}$ is the volume flow of the fluid; v^*_{eq} denotes the volumetric fluid flux needed to keep a suspension in equilibrium. It is computed from the Richardson Zaki model as described in [25].

315 The second term on the right-hand side of the population balance equation (23) stands for particle transport by dispersion.

The third term is due to particle growth with the growth rate

$$G(x) = k\frac{c(x) - c_{sat}}{c_{sat}} \tag{29}$$

The dynamic behaviour of the concentration $c(x)$ from (29) is described by the following balance equation of the solute in the liquid phase

$$\frac{\partial}{\partial t}(A_{eff}(x)c) = -\frac{\dot{V}}{A(x)}\frac{\partial c}{\partial x} + A_{eff}(x)D_f\frac{\partial^2 c}{\partial x^2} + \frac{\rho_p}{\rho_f}\int_0^\infty \frac{\pi}{6}G(x)\hat{L}^3\frac{\partial n}{\partial L}dL, \tag{30}$$

where $A_{eff}(x)$ denotes the effective area of the crystallizer.

The last term of (23) describes the effect of the ultrasonic attenuator on the particle population. $\dot{V}_{us}$ is the volume flow to and from the attenuator. The equation for the number size distribution in flow from the ultrasonic attenuator $n_{from_{us}}$ reads:

$$\begin{aligned} \frac{\partial n_{from_{us}}(L)}{\partial t} = \Bigg(& \frac{\dot{V}_{us}}{V_{us}}\Big(n(x_{us},L,t) - n_{from_{us}}(L)\Big) \\ & + \frac{1}{\tau_{us}}\Big(n_{us}(L)k_{us} - n_{from_{us}}(L)\Big)\Bigg) \end{aligned} \tag{31}$$

n_{us} is chosen as

$$n_{us}(L) = exp\Bigg(- \frac{10^{-3}}{L}\frac{(L-L_{us})^2}{2\sigma^2_{us,L}}\Bigg) \tag{32}$$

The scaling factor k_{us} can be calculated as

$$k_{us} = \frac{\int_0^\infty n_{from_{us}}(L)L^3dL}{\int_0^\infty n_{us}(L)L^3dL} \tag{33}$$

The expression

$$\delta(x - x_{us}) = \frac{1}{\sigma_{us,x}\sqrt{2\pi}} exp\left(- \frac{(x - x_{us})^2}{2\sigma_{us,x}^2} \right) \tag{34}$$

approximates the spatial spread of the extraction of particles due to the finite diameter of the connecting tube between crystallizer and attenuator.

The method of lines is used to convert the reference system into a spatially discretized form. For the numerical solution a finite volume scheme is applied. Since the particle velocity v_p may change its sign along the x coordinate, gradients in this direction are approximated by central differences to provide numerical stability under these circumstances. The following discretization grid has been applied: 120 points in the direction of the external coordinate x and 80 points in the direction of the internal coordinate L. In total, the reference model consists of 9800 ordinary differential equations.

As a demonstrative example, the reduced model has to be produced that approximates the following dynamic of the reference model. An initial state of the fluidized bed crystallizer is the stationary state when all system parameters are set to their default values. Then the volume flow of the fluid $\dot{V}$ increases from $2.5 \cdot 10^{-6}$ m^3 s^{-1} to $2.8 \cdot 10^{-6}$ m^3 s^{-1}. A new stationary state is reached after 3000 s. The dynamic behavior of the system during this time is shown in Fig. 9.

To perform the model reduction, snapshots are collected on an equidistant time grid for $t = 0..3000$ s with interval of 1 s. Using the linear regression method for linearization and specifying the truncation errors $e_x = 10^{-6}$ and $e_g = 10^{-2}$, the tool generated the reduced model with 45 ordinary differential equations and 66 algebraic equations, compared to 9800 equations of the reference model. Fig. 10 shows good agreement of approximation with the exact solution. The error takes the largest value at the beginning when the biggest particle population decreases rapidly and it goes down as the system reaches the stationary state.

The example illustrates that the developed software tool is able to handle nonlinear models of high complexity, for which a manual model reduction would

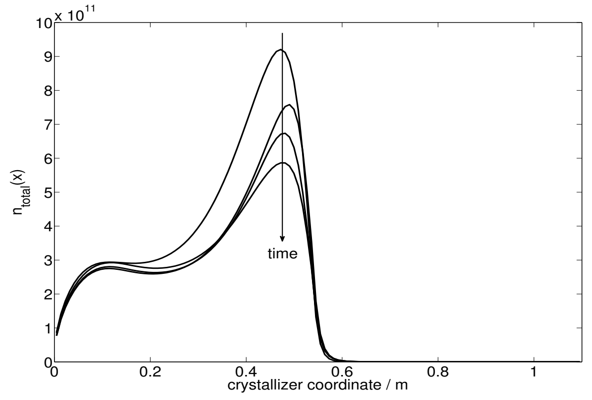


Figure 9: Profiles of total number of particles $n_{total}(x)$ at time points $t = 0$ s, $t = 250$ s, $t = 500$ s, $t = 3000$ s.

be a considerable task.

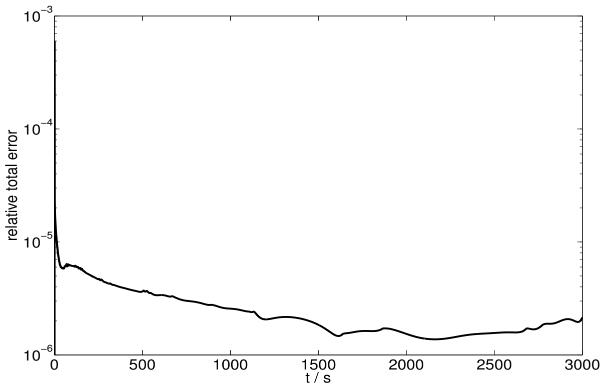


Figure 10: Simulation results of the reduced fluidized bed crystallizer model.

4.3. Case studies: summary

The summary information with all the specifics about the generated reduced models is presented in Table 1. The simulations in this work were carried out on 350 a personal computer with an Intel(R) Core(TM) i5-4590 3.30 GHz and 32 GB RAM running the Ubuntu 12.04.5 LTS. The computational effort is measured with the Python command *clock* from the package *time* as CPU time in seconds.

Table 1: Summary information about the generated reduced models.

	Number of equations		Computational time (seconds)		
Experiment	Reference	Reduced	Offline phase	Reference	Reduced
Conductor (scenario 1) using linear regression	10800 ODEs	19 ODEs + 1 alg	783.65	41.12	5.04
Conductor (scenario 1) using analytical Jacobian	10800 ODEs	19 ODEs + 6 alg	212.09	41.12	7.91
Conductor (scenario 1) using analytical Jacobian	10800 ODEs	19 ODEs + 16 alg	212.00	41.12	7.64
Conductor (scenario 1) using analytical Jacobian	10800 ODEs	19 ODEs + 25 alg	210.79	41.12	7.54
Conductor (scenario 2) using linear regression	10800 ODEs	66 ODEs + 76 alg	7253.60	992.81	546.67
Crystallizer using linear regression	9800 ODEs	45 ODEs + 66 alg	3993.20	1138.84	102.72

5. Conclusions

The automatic tool for the model reduction has been developed by using proper orthogonal decomposition combined with empirical interpolation. For demonstration purposes a virtual machine has been prepared with all the needed software installed. It is freely available for download from `http://promottrac.mpi-magdeburg.mpg.de/dist/pod/promot-pod-reducer-32bit.ova`.

Although the basis functions from snapshots of the reference model give some hints on the accuracy to be expected from the reduced model, depending on many factors the approximation error during runtime of the reduced model can leave the desired range. For this purpose an efficient a-posteriori error estimator proposed in [26] has to be implemented.

Acknowledgements

This work has been supported financially by DFG in the framework of SPP 1679, contract nr. MA 2051/2-1.

List of Symbols

Latin symbols:

symbol	*description*	*unit*	*value*
A	right system matrix of reference model		
A_{red}	right system matrix of reduced model		
$A(x)$	cross-sectional area of crystallizer	m^2	
$A_{eff}(x)$	effective area of crystallizer	m^2	
B	left system matrix of reference model		
B_{red}	left system matrix of reduced model		
c	constant vector of reference model		
c_{red}	constant vector of reduced model		
c_{sat}	saturated solution concentration	1	0.0051
$c(x)$	liquid phase concentration	1	
D	dispersion coefficient in particle phase	$\mathrm{m}^2\,\mathrm{s}^{-1}$	10^{-4}
D_f	dispersion coefficient in liquid phase	$\mathrm{m}^2\,\mathrm{s}^{-1}$	10^{-4}
d	diameter of crystallizer	m	
d_{bottom}	diameter at the bottom of crystallizer	m	$1.5 \cdot 10^{-2}$
d_{top}	diameter at the top of crystallizer	m	$3 \cdot 10^{-2}$
e_x	truncation error for POD modes	1	
e_g	truncation error for basis functions derived by empirical interpolation	1	
$f(x(t))$	right-hand sides of equations		
F	snapshots for $f(x(t))$		
$G(x)$	particle growth rate	$\mathrm{m}\,\mathrm{s}^{-1}$	
$g(x(t))$	nonlinearities of reference model		
H	height of crystallizer	m	1.1
k_{us}	scaling factor for n_{us}	1	
L	internal coordinate / particle diameter	m	
k	growth rate constant	1	$3.37 \cdot 10^{-7}$
k_1	coefficient in thermal diffusivity	1	1

k_2	coefficient in thermal diffusivity	1	10^{-2}
k_3	coefficient in thermal diffusivity	1	10^{-4}
L_{us}	mean value of ultrasonic outlet number size density	m	$5.0 \cdot 10^{-5}$
N_d	number of snapshots	1	
N^x	number of differential equations	1	
N^g	number of algebraic equations	1	
$n(x, L, t)$	number size density	$\mathrm{m}^{-3}\mathrm{m}^{-1}$	
n_{us}	output size distribution	$\mathrm{m}^{-3}\mathrm{m}^{-1}$	
$n_{from_{us}}$	number size distribution in flow from ultrasonic attenuator	$\mathrm{m}^{-3}\mathrm{m}^{-1}$	
s	symbolic name of state variable		
T	temperature	°C	
T_f	temperature of the upper boundary	°C	
t	time	s	
U	POD modes		
$\dot{V}$	fluid volume flow	$\mathrm{m}^3\,\mathrm{s}^{-1}$	$2.5 \cdot 10^{-6}$
$\dot{V}_{us}$	volume flow through attenuator	$\mathrm{m}^3\,\mathrm{s}^{-1}$	$1.17 \cdot 10^{-5}$
V_{us}	volume of ultrasonic attenuator	m^3	10^{-3}
v^*_{eq}	volumetric fluid flux	$\mathrm{m\,s}^{-1}$	
v_p	particle velocity	$\mathrm{m\,s}^{-1}$	
$x(t)$	discretized state vector		
x	space coordinate	m	
x_{US}	position of the connection between crystallizer and ultrasonic attenuator	m	0.025
$\hat{x}(t)$	approximation of reduced model		
X	snapshots for $x(t)$		

Greek symbols:

symbol	*description*	*unit*	*value*

κ	thermal diffusivity	$\mathrm{W\,m^{-1}\,K^{-1}}$	
Σ	diagonal matrix with singular values		
$\sigma_{US,L}$	standard deviation of ultrasonic outlet number size density	m	10^{-5}
$\sigma_{US,x}$	shaping parameter for exchange flow between crystallizer and attenuator	m	
ϕ^x	state vector of Ψ^x		
ϕ^g	state vector of Ψ^g		
Ψ^g	matrix of reduced basis vectors for $g(x(t))$		
Ψ^x	matrix of reduced basis vectors for $x(t)$		
τ_{us}	characteristic time of attenuation	s	100

References

[1] D. Shi, N. H. El-Farra, M. Li, P. Mhaskar, P. D. Christofides, Predictive control of particle size distribution in particulate processes, Chemical Engineering Science 61 (1) (2006) 268–281. `doi:10.1016/j.ces.2004.12.059`.

[2] W. Marquardt, Traveling Waves in Chemical Processes, ResearchGate 30 (4) (1990) 585–606.

[3] A. Kienle, Low-order dynamic models for ideal multicomponent distillation processes using nonlinear wave propagation theory, Chemical Engineering Science 55 (2000) 1817–1828. `doi:10.1016/S0009-2509(99)00463-7`.

[4] D. L. Marchisio, R. O. Fox, Solution of population balance equations using the direct quadrature method of moments, Journal of Aerosol Science 36 (1) (2005) 43–73. `doi:10.1016/j.jaerosci.2004.07.009`.

[5] N. Lebaz, A. Cockx, M. Sprandio, J. Morchain, Reconstruction of a distribution from a finite number of its moments: A comparative study in the case of depolymerization process, Computers & Chemical Engineering 84 (2016) 326–337. `doi:10.1016/j.compchemeng.2015.09.008`.

[6] P. D. Christofides, P. Daoutidis, Finite-Dimensional Control of Parabolic PDE Systems Using Approximate Inertial Manifolds, Journal of Mathematical Analysis and Applications 216 (2) (1997) 398–420. doi:10.1006/jmaa.1997.5649.

[7] P. Benner, E. S. Quintana-ortí, G. Quintana-ortí, Balanced Truncation Model Reduction of Large-Scale Dense Systems on Parallel Computers, Mathematical and Computer Modelling of Dynamical Systems 6 (4) (2000) 383–405. doi:10.1076/mcmd.6.4.383.3658.

[8] M. Heinkenschloss, T. Reis, A. C. Antoulas, Balanced truncation model reduction for systems with inhomogeneous initial conditions, Automatica 47 (3) (2011) 559–564. doi:10.1016/j.automatica.2010.12.002.

[9] K. Kunisch, S. Volkwein, Galerkin Proper Orthogonal Decomposition Methods for a General Equation in Fluid Dynamics, SIAM Journal on Numerical Analysis 40 (2) (2002) 492–515. doi:10.1137/S0036142900382612.

[10] H. M. Park, D. H. Cho, The use of the Karhunen-Loéve decomposition for the modeling of distributed parameter systems, Chemical Engineering Science 51 (1) (1996) 81–98. doi:10.1016/0009-2509(95)00230-8.

[11] L. Sirovich, Turbulence and the dynamics of coherent structures part i: coherent structures, Quarterly of Applied Mathematics 45 (3) (1987) 561–571.

[12] A. C. Antoulas, Approximation of Large-scale Dynamical Systems, Society for Industrial and Applied Mathematics, 2005.

[13] M. Krasnyk, M. Mangold, Reduction of a Urea Crystallizer Model by Proper Orthogonal Decomposition and Best-Points Interpolation, Industrial & Engineering Chemistry Research 49 (20) (2010) 9887–9898. doi:10.1021/ie901988t.

[14] M. Mangold, L. Feng, D. Khlopov, S. Palis, P. Benner, D. Binev, A. Seidel-Morgenstern, Nonlinear model reduction of a continuous fluidized bed crystallizer, Journal of Computational and Applied Mathematics 289 (2015) 253–266. doi:10.1016/j.cam.2015.01.028.

[15] M. Mangold, Model reduction of batch drum granulator by proper orthogonal decomposition, in: 8th IFAC International Symposium on Advanced Control of Chemical Processes, Singapore, 2012, pp. 856–861. doi:10.3182/20120710-4-SG-2026.00049.

[16] W. E. Schiesser, The Numerical Method of Lines: Integration of Partial Differential Equations, 1st Edition, Academic Press, San Diego, 1991.

[17] M. A. Grepl, Y. Maday, N. C. Nguyen, A. T. Patera, Efficient reduced-basis treatment of nonaffine and nonlinear partial differential equations, ESAIM: Mathematical Modelling and Numerical Analysis 41 (03) (2007) 575–605. doi:10.1051/m2an:2007031.

[18] N. C. Nguyen, A. T. Patera, J. Peraire, A best points interpolation method for efficient approximation of parametrized functions, International Journal for Numerical Methods in Engineering 73 (4) (2008) 521–543. doi:10.1002/nme.2086.

[19] M. Mangold, D. Khlopov, G. Danker, S. Palis, V. Svjatnyj, A. Kienle, Development and Nonlinear Analysis of Dynamic Plant Models in ProMoT /Diana, Chemie Ingenieur Technik 86 (7) (2014) 1107–1116. doi:10.1002/cite.201400003.

[20] M. Ginkel, A. Kremling, T. Nutsch, R. Rehner, E. D. Gilles, Modular modeling of cellular systems with ProMoT/Diva, Bioinformatics 19 (9) (2003) 1169–1176. doi:10.1093/bioinformatics/btg128.

[21] M. Krasnyk, Diana - An object-oriented tool for nonlinear analysis of chemical processes, Ph.D. thesis, Otto-von-Guericke-Universitt, Magdeburg, Germany (2008).

[22] Maxima, a Computer Algebra System, Web page.
URL `http://maxima.sourceforge.net/`

[23] J. W. Eaton, D. Bateman, S. Hauberg, GNU Octave version 3.0.1 manual: a high-level interactive language for numerical computations, CreateSpace Independent Publishing Platform, 2009, ISBN 1441413006.
URL `http://www.gnu.org/software/octave/doc/interpreter`

[24] A. C. Hindmarsh, A. G. Taylor, User documentation for IDA, a differential-algebraic equation solver for sequential and parallel computers, Tech. Rep. UCRL-MA-136910, Lawrence Livermore National Laboratory (1999).

[25] D. Binev, A. Seidel-Morgenstern, H. Lorenz, Continuous Separation of Isomers in Fluidized Bed Crystallizers, Crystal Growth & Design 16 (3) (2016) 1409–1419. `doi:10.1021/acs.cgd.5b01513`.

[26] Y. Zhang, L. Feng, S. Li, P. Benner, An Efficient Output Error Estimation for Model Order Reduction of Parametrized Evolution Equations, SIAM Journal on Scientific Computing 37 (6) (2015) B910–B936. `doi:10.1137/140998603`.