MOSAIC: An Online Modeling Platform Supporting Automatic Discretization of Partial Differential Equation Systems

Erik Esche*, David Müller, Gregor Tolksdorf, Robert Kraus, Günter Wozny

Chair of Process Dynamics and Operation, Berlin University of Technology, Sekr. KWT-9, Str. des 17. Juni 135, D-10623 Berlin, Germany
erik.esche@tu-berlin.de

Abstract

Partial differential algebraic equation systems (PDAE) frequently appear in chemical engineering and their discretization is most often an issue when preparing a simulation wherein they appear. In this contribution, an algorithm is presented and implemented facilitating the general analysis of PDAE systems appearing often in chemical engineering problems and their discretization via orthogonal collocation on finite elements. The recognition of differentiating variables, the application of varying boundary conditions, and the subdivision into independent PDAE systems as well as the implementation of the actual discretization is discussed.

Keywords: PDAE, Automatic Discretization, Orthogonal Collocation, Code Generation

1. Introduction

An inherent issue in the preparation of simulations is the discretization of differential algebraic equation systems (DAEs) or even partial differential algebraic equation systems (PDAEs). Tools like Mathworks' Matlab® or PSE's gPROMS® can handle DAEs which can easily be turned into ordinary differential equation systems (ODEs), but are generally unable to process systems of a higher order or even PDAE systems without further user input. In chemical engineering, PDAE systems appear whenever the multidimensionality of any piece of equipment needs to be described. Be it axial and radial flow of a tubular reactor or the axial profiles of an absorption column changing with time. Many different mathematical tools exist to rewrite PDAE systems as fully discretized algebraic equation systems (AE). Among the more prominent of those are the method of lines, finite differences, finite elements, finite volumes, and orthogonal collocation on finite elements (OCFE) (Finlayson, 1980, Scherer, 2013). So far, only very little effort has been made towards a software-based facilitation of the discretization of partial differential equations, e.g. (Sincovec et al., 1975) and (Yamabe et al., 1990). In this contribution, an algorithm is presented, which facilitates the OCFE of complex systems in a user interface. The algorithm has already been implemented and tested in Matlab® and is currently in the process of being integrated into the online modeling, simulation, and optimization platform MOSAIC (Kuntsche et al., 2011).

2. Status Quo

The online platform MOSAIC is a tool for modeling and simulating phenomena, units, or whole chemical processes. The current version of the tool is already able to automatically identify a user-defined equation system. Depending on whether it is a pure algebraic equation system (AE), ordinary differential equation system (ODE), or a differential algebraic equation system (DAE), MOSAIC has varying specifications for handling it. Among these specifications is the treatment of initial values, ranges of differentiating variables, and the choice of solvers the equation systems can be exported for. On top of that, DAE systems consisting of ODE and AE parts can be semi-automatically reformulated using first order Lagrangian collocation or Euler methods to automatically provide fully discretized AE systems. All of these equations can be transferred to any desired programming language, be it Fortran, C++, Java, Python, Matlab, gPROMS, AMPL, or GAMS.

3. Algorithm for Automatic Discretization of PDAE Systems

The following paragraphs discuss in detail how an initially supplied PDAE system is analyzed and subsequently fully discretized using OCFE. Required user input is noted whenever several choices exist to proceed. This algorithm is not meant as the “sine qua non” solution to discretizing any PDAE system at all. Instead, it is meant to tackle the discretization of most systems appearing in chemical engineering. Any system, in which state variables are differentiated with respect to other state variables, are neglected at this stage as these do not commonly appear in this field. To further reduce the complexity for this contribution, consistency tests for boundary condition definitions will be skipped and only systems with first and second order derivatives are discussed.

3.1. Prerogatives

Firstly, it is assumed that the PDAE is supplied as given in Eq. (1).

$$\vec{g}(\vec{x}, \frac{\partial \vec{x}}{\partial t}, \frac{\partial \vec{x}}{\partial z}, \frac{\partial \vec{x}}{\partial r}, \frac{\partial^2 \vec{x}}{\partial r^2}, \ldots, t, z, r) = \vec{0} \tag{1}$$

This should, however, not be a limitation to the order of the partial differential equation system (PDAE) or the number of differentiating variables, or subsystems therein.

3.2. Subdivision of Differentiating Variables

In this step, the PDAE is scanned for all appearances of derivatives and the denominators of those are identified as differentiating variables. Within the scope of Eq. (1) this would be *t*, *r*, and *z*. Any PDAE system might consist of PDAE subsystems, which could and should be discretized separately as there is no connection between the respective differentiating variables, although their physical meaning might be identical. An example might be a reactor network, in which reactors are described both axially and radially and are bound to the network by inlet conditions and averaging of the outlets. Within MOSAIC the variables for each reactor would each have different namespaces, which are prefixes to the variables, e.g. *e0_z* and *e1_z* for the axial coordinate. Once all differentiating variables are identified, the user is required to specify ranges for each: $t_{min} < t_{max}$, $z_{min} < z_{max}$ etc.

3.3. Analysis of State Variables

As a next step the numerators of said (partial) derivatives are scanned to generate a comprehensive list (*DS*, differentiated states) of all differentiated state variables *x*, which appear therein. For each *x* in *DS* the (partial) derivatives are stored in a second list called *PDL(x)* (partial derivatives list). Based on this new list, dimensions and derivative orders are determined for each *x* as explained in Tab. 1. For the multidimensional cases, in which *x* depends on more than one differentiating variable the next step allows for a splitting of the ranges. In those cases, the user is asked to enter a number of sections for each differentiating variable and a length for each. n_t is the number of sections for variable *t* and □t_i the set of section lengths. Of course, the boundary condition options in Tab. 1 multiply with the number of subdivision of each differentiating variable.

The last step in the analysis of the state variables is the actual definition of boundary conditions. At this point there are three possibilities:

1. Either a constant value is assigned to the boundary condition.
2. Or a statement for the boundary condition is supplied, which is solely a function of the differentiating variables.
3. Or an additional MOSAIC equation is supplied as a boundary condition, which contains the state or a derivative of the state in an implicit form and could even connect to other model parts of the whole MOSAIC equation system. In case this is already included in the PDAE system, this needs to be marked by the user and the range of validity needs to be specified as required.

Table 1. User options for defining boundary conditions for different sets of *PDL(x)*.

PDL(x) for a single *x*	Required Initial or Boundary Conditions
dx/dt	Select one of: $x(t_{\mathrm{min}}), x(t_{\mathrm{max}})$
d^2x/dt^2	Select two out of: $x(t_{\mathrm{min}}), x(t_{\mathrm{max}}), dx/dt\|_{t_{\mathrm{min}}}, dx/dt\|_{t_{\mathrm{max}}}$
$dx/dt, d^2x/dt^2$	Select two out of: $x(t_{\mathrm{min}}), x(t_{\mathrm{max}}), dx/dt\|_{t_{\mathrm{min}}}, dx/dt\|_{t_{\mathrm{max}}}$
Derivatives with only r, or z	*Equivalent to t*
$dx/dt, dx/dr$	Select one of: $x(t_{\mathrm{min}}, r), x(t_{\mathrm{max}}, r)$ and one of: $x(t, r_{\mathrm{min}}), x(t, r_{\mathrm{max}})$
$dx/dt, d^2x/dr^2$	Select one of: $x(t_{\mathrm{min}}, r), x(t_{\mathrm{max}}, r)$ and two out of: $x(t, r_{\mathrm{min}}), x(t, r_{\mathrm{max}}), dx/dt\|_{r_{\mathrm{min}}}, dx/dt\|_{r_{\mathrm{max}}}$

3.4. Discretization of PDAE Systems

Once all the boundary conditions are set, the overall PDAE system needs to be decomposed into independent subsystems. This means systems which are only attached to the overall system through initial or boundary conditions and can therefore be discretized independently from everything else. The identification of these subsystems will be performed in the following Steps:

1. The first element of *DS* is taken and all the equations of the PDAE system are collected containing it.
2. All state variables of this collection of equations are identified and equations are added wherein they appear. This step is repeated until no new equations are added.

3. This new subset of equations is considered to be an independent PDAE system and all variables therein need to be discretized equally. The new subsystem is yet again analyzed with respect to the (partial) derivatives to determine the required order of the discretization method. This procedure is outlined in Tab. 2. As said above, boundary conditions are considered to be the borderlines of each independent subsystem. The only exception is subsystems which are bound together by coupling equations. At this point, additional user input is required to pair certain differentiating variables and bind these coupled subsystems together.
4. Once the collocation specifications are set according to Step 3, the number of finite elements for each variable needs to be specified by the user. An important constraint herein is the definition of sections of boundary conditions for multidimensional systems. The number of finite elements for each differentiating variable should at least be greater or equal to the number of boundary sections for the same value. For *t* this would mean $FE_t \geq n_t$. In addition to the number, the length of the finite elements also needs to be set. The notation and specification is shown in Fig. 2, wherein □t_{ij} refers to the length of finite element *j* for boundary condition section *i* of differentiating variable *t*.

Table 2. Examples for user options for applying orthogonal collocation on systems with various dimensions of partial differentials.

Appearing (Partial) Derivatives	Examples for the Collocation Specifications
d/dt	Linear combination of Lagrangian polynomials on finite elements
$d/dt, d/dr$	Bilinear combination of two sets of Lagrangian polynomials on finite segments
$d/dt, d^2/dr^2$	Bilinear combination of Lagrangian polynomials (for *t*) and Hermite polynomials (for *r*) on finite segments
$d/dr, d^2/dr^2, d/dz, d^2/dz^2$	Bilinear combination of two sets of Hermite polynomials on finite segments
$d/dt, d/dr, d^2/dr^2, d/dz, d^2/dz^2$	Trilinear combination of Lagrangian polynomials (for *t*) and two sets of Hermite polynomials (for *r* and *z*) on finite volumes.

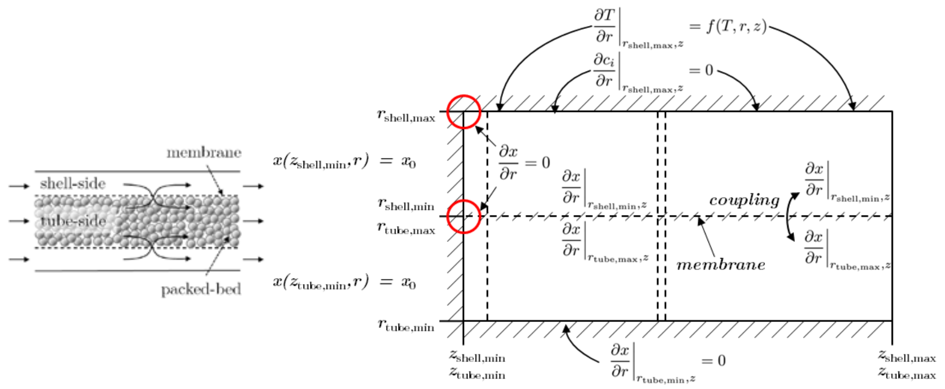
5. After the definition of all finite elements, user input is required again. The order of the collocating polynomials needs to be chosen as well as the position of the roots within each finite element (shifted Radau, Legendre etc.). Based on these choices the values and the derivatives (first and second order depending on the system) of the collocating polynomials are automatically calculated at all collocation positions (roots) within MOSAIC and saved as parameter values.
6. In this step the actual discretization is performed. New indices for the boundary condition sections, the finite elements, and the collocation positions therein are appended to all state variables in the PDAE subsystem. The new generic form of the system is then reinstantiated by applying the ranges of the newly added indices. The previously defined initial values for all states are reused for their discretized sets; the same is true for previously set lower and upper bounds. The (partial) derivatives within the equations are replaced by variables, for which additional equations are added to calculate and to

relate them to their respective states. In addition, all boundary conditions are applied and added as additional equations to the system as well as expressions for calculating all differentiating variables at all positions.

7. The non-discretized subsystem is removed from the whole PDAE system and replaced by its fully discretized version. MOSAIC connectors, which tethered this subsystem to the overall PDAE, are reapplied and connected to the correct bounds.
8. Finally all states appearing in the discretized subsystem are removed from *DS* and all steps starting with Step 1 are repeated until *DS* is finally empty.

4. Case Study: 2D Discretization of a PDAE System

The procedure above has already been applied to a conventional packed-bed membrane reactor in a reactor network for the oxidative coupling of methane. Fig. 1 shows a sketch of the reactor. CH_4 and N_2 are fed to the tube-side, O_2 diluted with even more N_2 to the shell. The packed-bed holds the catalyst to facilitate the direct conversion of CH_4 to C_2H_4. In addition, both outlet streams contain a number of byproducts, namely: CO, CO_2, H_2, H_2O, and leftover CH_4 as well as N_2. The modeling of the reactor has already been discussed in (Esche et al., 2012) and (Esche et al., 2011).



membrane
shell-side
tube-side
packed-bed
$x(z_{\text{shell,min}}, r) = x_0$
$x(z_{\text{tube,min}}, r) = x_0$
$r_{\text{shell,max}}$
$r_{\text{shell,min}}$
$r_{\text{tube,max}}$
$r_{\text{tube,min}}$
$\left.\frac{\partial T}{\partial r}\right|_{r_{\text{shell,max}},z} = f(T,r,z)$
$\left.\frac{\partial c_i}{\partial r}\right|_{r_{\text{shell,max}},z} = 0$
$\frac{\partial x}{\partial r} = 0$
$\left.\frac{\partial x}{\partial r}\right|_{r_{\text{shell,min}},z}$
$\left.\frac{\partial x}{\partial r}\right|_{r_{\text{tube,max}},z}$
coupling
membrane
$\left.\frac{\partial x}{\partial r}\right|_{r_{\text{tube,min}},z} = 0$
$z_{\text{shell,min}}$
$z_{\text{tube,min}}$
$z_{\text{shell,max}}$
$z_{\text{tube,max}}$

Figure 1. Sketch of the conventional packed-bed membrane reactor (left): N_2 and CH_4 are fed to the catalytic packed-bed. O_2 crosses the non-selective membrane from the shell-side into the packed-bed. Definition of the boundary conditions for the reactor (right).

At this point the model itself will only be sketched very roughly to facilitate the procedure outlined above. The core of the PDAE system is the set of nine component balances and the energy balance describing the concentration and temperature fields:

$$0 = -u_z \cdot \frac{\partial c_i(r,z)}{\partial z} + \mathcal{D}_{i,r} \cdot \left[\frac{\partial^2 c_i}{\partial r^2} + \frac{1}{r} \cdot \frac{\partial c_i}{\partial r}\right] + \dot{cr}_i \tag{2}$$

$$c_{\text{tot}} \cdot c_{p,\,\text{tot}} \cdot u_z \cdot \frac{\partial T}{\partial z} = \lambda \cdot \left[\frac{\partial^2 T}{\partial r^2} + \frac{1}{r} \cdot \frac{\partial T}{\partial r}\right] + \sum_{j=1}^{NR} \left(\varphi_{\text{cat}} \cdot \varrho_{\text{cat}} \cdot \dot{r}r_j \cdot (-\Delta_R H)\right) \tag{3}$$

Obviously, the whole reactor is modeled two-dimensionally. The balances contain axial flow (coordinate *z*) and radial diffusion (coordinate *r*). Component specific radial diffusion coefficients $D_{i,r}$, component specific reaction rates cr_i, concentrations c_{tot}, heat capacities $c_{p,tot}$, thermal conductivities □, reaction rates rr_j are calculated in additional equations, which are directly included in the PDAE system. To add to the complexity, the reaction terms cr_i and rr_j need to be neglected for describing the shell-side of the reactor. $□_{cat}\cdot□_{cat}$ represents the catalyst amount. The mass transfer through the membrane is governed by Knudsen diffusion. The reactor is 20 cm long and each 10 cm a different heating segment exists to heat or cool the system. The reactor is part of a larger network of reactors discussed in (Esche et al., 2013). Let it suffice to say, that the shell- and tube-side feeds of the reactor, i.e. the concentrations $c_i(r,z=0)$ and temperatures $T(r,z=0)$, are tethered to the network streams and that the outlet conditions $c_i(r,z=z_{end})$ *and* $T(r,z=z_{end})$ are returned to the network. From here on, each step of the algorithm described above will be applied to the briefly outlined PDAE system. Firstly, all prerogatives are fulfilled and the PDAE system is supplied in an appropriate form. The system contains four differentiating variables as the whole model is split up between tube and shell: r_{tube}, r_{shell}, z_{tube}, z_{shell}. In total, the subdivisions of the differentiating variables are as follows: $r_{tube,min}$ = 0.0cm, $r_{tube,max}$ = 3.5cm, $r_{shell,min}$ = 3.5cm, $r_{shell,max}$ = 5.0cm; $z_{tube,min}$ = $z_{shell,min}$ = 0.0cm, $z_{tube,max}$ = $z_{shell,max}$ = 20.0cm. The analysis of the state variables of the entire system leads to a list *DS* containing component concentrations c_i and temperatures T for shell and tube respectively. The respective *PDL(x)* for each contains the following derivatives: $d/dz, d/dr, d^2/dr^2$. In this case, the user decides to enter concentrations and temperatures at all inlet positions and to supply radial gradient information for both r_{min} and r_{max} at all axial positions as shown in Fig. 1 (right). Given the flux through the membranes, which collides with radially constant feed concentrations, and the two separate heating sections, z_{shell} and z_{tube} are further subdivided. Fig. 1 (right) shows how this subdivision is carried out and presents details on the applied boundary conditions. The boundary conditions for the membrane side are added as additional equations, the same is true for the heat flux at the outer shell of the reactor at $r_{shell,max}$. The supplied boundary conditions make separating shell- and tube-side equations impossible as the membrane closely links them. Hence, in the first step of the discretization, all equations and state variables belonging to the entire reactor are collected. The user needs to interfere at this point and pair z_{shell} and z_{tube} as well as r_{shell} and r_{tube}. This leads to a drastic reduction of the differentials appearing, yet again leading to: $\partial/\partial z, \partial/\partial r, \partial^2/\partial r^2$. For this system, the user chooses to apply Lagrangian polynomials for the axial dependency and Hermite polynomials for the radial. For the Lagrangian polynomials shifted Radau roots and for the Hermite polynomials shifted Legendre roots are applied. During the actual discretization the procedure leads to a set of 130,000 algebraic equations.

5. Conclusions

The presented algorithm is expected to work robustly on most PDAE systems appearing in chemical engineering. The systematic analysis of the system and smaller subsystems is essential for guaranteeing a logical discretization. Up to now the algorithm has only been applied in Matlab and tested on a single, albeit complex, problem. As a next step, a more generic form of the algorithm is implemented in MOSAIC, to ease the user interaction and to allow for an easier set-up of the entire system.

Acknowledgements

The authors acknowledge the support from the Cluster of Excellence “Unifying Concepts in Catalysis” and the Collaborative Research Center SFB/TR 63 InPROMPT “Integrated Chemical Processes in Liquid Multiphase Systems” both coordinated by the Technische Universität Berlin and funded by the German Research Foundation (Deutsche Forschungsgemeinschaft “DFG”).

References

E. Esche, H. Arellano-Garcia, L.T. Biegler, Optimal Operation of a Membrane Reactor Network, AIChE Journal (2013), DOI: 10.1002/aic.14252

E. Esche, H. Arellano-Garcia, G. Wozny, L.T. Biegler, 2012, Optimal Operation of Membrane Reactor Network, Computer Aided Chemcial Engineering, Vol. 31, 1321-1325, DOI: 10.1016/B978-0-444-59506-5.50095-X

B.A. Finlayson, 1980, Nonlinear Analysis in Chemical Engineering, McGraw-Hill Inc., ISBN 0-07-020915-4

S. Kuntsche, T. Barz, R. Kraus, H. Arellano-Garcia, G. Wozny, 2011, MOSAIC a web-based modeling environment for code generation, Computers & Chemical Engineering, Vol. 35, 11, 2257-2273

P.O.J. Scherer, 2013, Computational Physics, Springer International Publishing, ISBN: 978-3-319-00400-6

R.F. Sincovec, N.K. Madsen, 1975, Software for Nonlinear Partial Differential Equations, ACM Transactions on Mathematical Software, Vol. 1, 3, 232-260

M. Yamabe, C. Konno, Y. Umetani, 1990, Automatic generation method of a simulation program for numerically solving a partial differential equation according to a boundary-fitted method, US Patent No. 4972334