DIVA – A Simulation Environment for Chemical Engineering Applications

Mohl, K.D., Spieker, A., Köhler, R., Gilles, E.D., Zeitz, M.
Institut für Systemdynamik und Regelungstechnik, Universität Stuttgart
email: mohl@isr.uni-stuttgart.de, fax: ++49-711-685-6371

Abstract

The detailed modelling and simulation of single process units as well as integrated production plants are an issue of growing importance. As a software tool addressing this issue the simulation environment DIVA is introduced. DIVA is an integrated tool because several numerical methods like simulation, continuation, and dynamic optimization are available. In this paper, the model representation and model implementation as well as the numerical methods employed in DIVA will be introduced. Several chemical engineering processes have been investigated with DIVA. The utilization of the numerical methods will be highlighted by an example from reactive distillation.

1 Introduction

The efficient use of resources, improved process, safety and environmental aspects lead to stronger integration of production processes. The close interaction within the same or among different process units can result in complex process dynamics. Therefore, suitable tools to deal with design problems and process operation of such complex processes are needed. Analysis and synthesis of the dynamic behaviour of integrated processes can be studied by application of dynamic simulation, nonlinear systems analysis, and dynamic optimization. The integrated simulation environment DIVA [1] (Dynamischer Simulator Verfahrenstechnischer Anlagen) [16, 10, 4, 1, 13, 5, 9], which is mainly being developed at the Institut für Systemdynamik und Regelungstechnik combines these aspects in a single tool.

Dynamic models of process units and process plants are frequently represented by ordinary differential systems (ODE) or differential algebraic systems (DAE) and typically result in highly nonlinear large sparse systems with complex structure due to recycle streams and feedback control. Further complexity is added by process models including discrete components like valves, discrete input functions, and bounded outputs of controllers. Thus, proper numerical methods are required to face these challenging tasks.

2 Model Representation and Program Architecture

In order to make process modelling and model implementation easy, the natural structure of a process plant as a network of interconnected process units is used. These individual process units are modeled in form of independent process unit models.

[1]Besides the authors, the following persons have contributed to DIVA or the code generator over the years: M. Bär, E. Dieterich, V. Gehrke, A. Gerstlauer, C. Gouma, M. Groebel, A. Helget, P. Holl, G. Hutzl, A. Kienle, A. Kröner, G. Lauschke, C. Majer, M. Mangold, W. Marquardt, T. Obertopp, S. Räumschüssel, H. Rettinger, J. Unger, H. Wieland.

For simple assembly of process unit models to plant models, all unit models are represented in the same form. In DIVA, the dynamic process unit models are represented in form of ODE (index zero) or linear-implicit DAE with a differential index of one [21]:

$$\begin{aligned} B_k(x_k, u_k, p_k, t) \cdot \frac{dx_k}{dt} &= f_k(x_k, u_k, p_k, t) \qquad t > t_0, \\ y_k = H_k \cdot x_k, & \qquad x_k(t=0) = x_k^0 \end{aligned} \tag{1}$$

process unit model index k	$\in$	I	left side matrix B_k	$\in$	$R^{n \times n}$
states x_k	$\in$	R^n	function vector f_k	$\in$	R^n
inputs u_k	$\in$	R^m	outputs y_k	$\in$	R^r
parameters p_k	$\in$	R^p	output matrix H_k	$\in$	$R^{r \times n}$
time t	$\in$	R^1	initial values x_k^0	$\in$	R^n

For distributed systems like tubular reactors, a discretization of the spatial domain using the method of lines with finite differences, finite volumina, or collocation methods has to be performed to obtain an ODE or DAE formulation. The representation of the unit model in form of a DAE (1) is not always optimal for all kinds of process units, because a numerical treatment using adaptive methods for the discretization would result in smaller computation times. But a simulation in context with other units is only possible then, by using complex numerics and software tools [8].

The connections among the process units are extracted from a flowsheet information file and represented in form of a coupling matrix $C = \{C_{ij}\}$, that connects the internal outputs y_i of the process unit j to the internal inputs u_i of the process unit i.

$$u_i = C_{ij} \cdot y_j, \qquad C_{ij} \in R^{m_i \times r_j} \tag{2}$$

By using the coupling matrix, the unit model DAEs are combined to form the plant model DAE, which has again the structure denoted in (1). Thus the model equations of the plant are treated simultaneously.

The individual process unit models, can be taken either from the library of process unit models or from user provided models that are implemented with the code generator of DIVA (see Sec. 3).

Fig. 1 further shows that due to the modular structure of the simulation environment all numerical methods available in DIVA can be applied to the plant model. The user may switch methods interactively. Additionally, the modular structure greatly simplifies the addition of numerical methods. Moreover, the possibility of processing one plant model using diverse numerical methods like continuation or optimization in an integrated simulation environment like DIVA reduces model implementation efforts because the model only has to be implemented once in the tool. This is particularly important because the modelling and model implementation steps are still the most time consuming in computer aided process engineering.

The simulation environment DIVA is implemented in FORTRAN77. At present it contains about 900 routines, and runs on the following hardware platforms, VAX/AXP under Open VMS, SUN under Solaris 2.x, and PC under OS/2 and Linux.

3 Model Implementation

DIVA contains a model library of generic process unit models. Examples of pre-defined process unit models are: distillation columns, condensers, reboilers, CSTRs, tubular reactors, valves, pumps, etc. The DIVA user selects individual process unit models from the model library and aggregates these to plant models by specifying plant flow sheets.

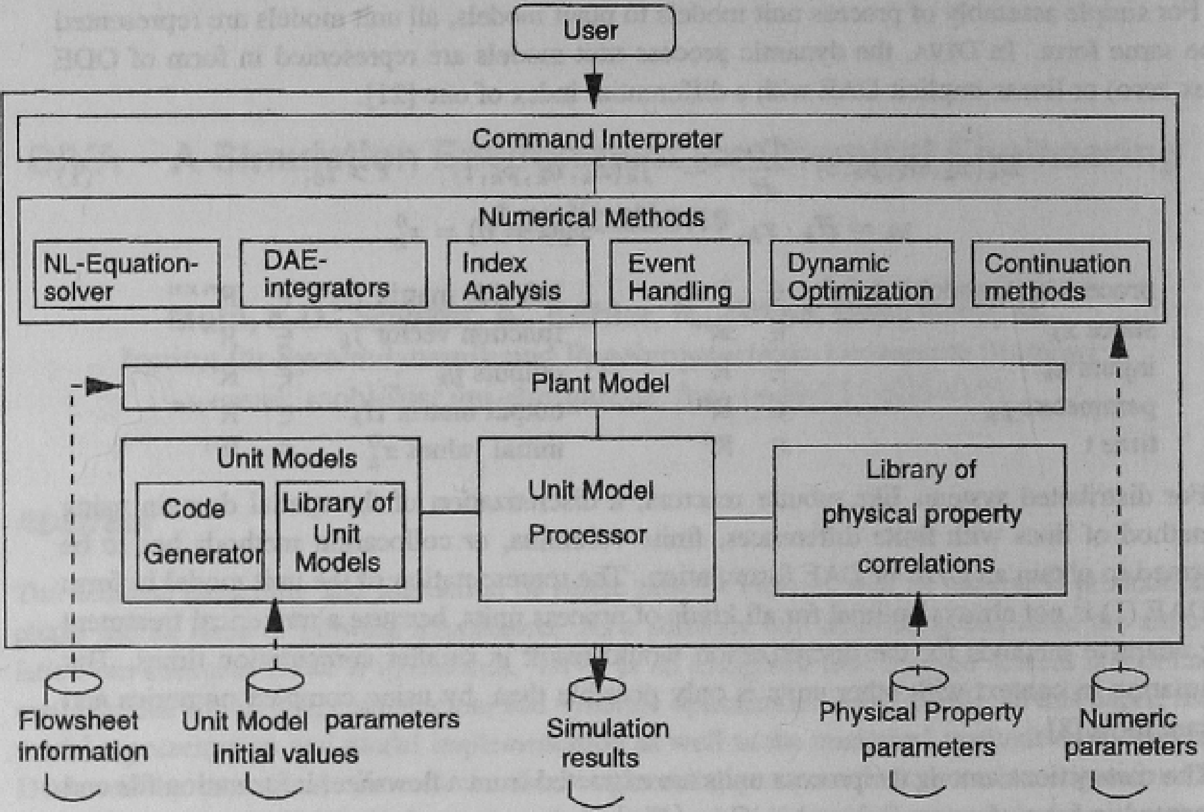

Figure 1: Architecture of the DIVA simulation environment

The process unit models are implemented as several FORTRAN subroutines in a well defined form. For each unit model (1), there is one subroutine for the initialization. The main tasks of this subroutine consist of reading the unit model parameters p_k and initial values x_k^0 from file, initializing the patterns of the Jacobians $\partial(B_k \cdot \dot{x}_k)/\partial x_k$, $\partial(B_k \cdot \dot{x}_k)/\partial u_k$, $\partial f_k/\partial x_k$, $\partial f_k/\partial u_k$ and doing storage management. A second subroutine is called by the numerical methods and evaluates the left side matrix B_k and the function vector f_k. A third subroutine is handling the output of simulation results to the computer console and to file. These three subroutines are obligatory. If the unit model has to account for discontinuities or if analytical Jacobians are used, additional subroutines are needed for the event handling and the evaluation of the Jacobians, respectively. Note that every unit model can be used several times in one plant. This is possible by using different names for the process unit in the plant and its representation in model equations. For each process unit of the plant, the unit model parameters and initial values are given in an individual file (Fig.1).

For the implementation of DIVA process unit models in the model library, the DIVA code generator can be used [19]. The code generator was designed to relieve the user of the time spending and error prone coding of the FORTRAN subroutines that represent the process unit models in the DIVA model library. Only a single input file containing the whole model information in symbolic form must be written by the modeler using the code generator input language. The code generator translates a given model into the required three FORTRAN subroutines for initialization, function evaluation, and management of the simulation output data. The user of the code generator does not need to have intimate knowledge about the structure of the generated FORTRAN subroutines.

The definition of a process unit model in the code generator language is divided into several parts. One of them is a general model description needed for documentation. The definition of the parameters p_k is distinguished by their type (structural, integer or real) and contains the names and default values. Therefore the parametrization of the process unit model is included. Further, the definitions of the input variables u_k and the output variables y_k as well as state variables x_k with initial values x_k^0 are parts of the code generator input file. A special construction is provided by auxiliary variables. They can be used for subterms appearing more than once in the equations. This increases the simulation performance because the assignments of these variables are just evaluated once instead of multiple evaluations of equation subterms. Auxiliary variables can be reused by assigning them several times, because the equations and assignments are evaluated in the order of their definition. This is minimizing storage consumption in connection with avoiding of alternative usage of additional state variables extending the equation system. An additional effect of auxiliary variables is a shorter and better readable unit model representation. Another part of the input file is the definition of process values to be stored as simulation data. Finally, the assignments of auxiliary variables and of model equations specifying the entities of the left side matrix B_k and the function vector f_k are given.

In order to generate the FORTRAN subroutines containing address calculations, storage management, analytical patterns of Jacobians, and function evaluations, the code generator must analyse and translate the model definition with respect of producing efficient code and avoiding multiple pattern references. Therefore, powerful algorithms are implemented. The FORTRAN code created by the code generator has a standard form and provides detailed comments. So it is well readable, an important fact for changes or extensions, e.g., for handling of discrete events, which are yet not supported by the code generator. Besides this, the code generator can be used to generate the DIVA input files, containing the unit model parameters and initial values.

An important task related to the implementation of a process model is the documentation. To achieve consistency in model formulation, parametrization, and documentation, an automatic text documentation was implemented in the code generator to depict the model information of the input file in a LaTeX document.

The code generator is implemented in the programming language LISP and runs on the following hardware platforms, SUN under Solaris 2.x and PC under OS/2 and Linux.

4 Numerical Methods

Based on the model representation of process units and plants, a simultaneous strategy is used for the numerical solution of ODE and DAE in DIVA. Due to the utilization of sparse matrix techniques and standard numerical routines like Harwells and BLAS, DIVA is suitable for large scale systems with as much as 10.000 state-variables with over 100.000 Jacobian entries. Systems of this scale have been realized in DIVA with reasonable computation times. In order to help the user checking for proper model formulation, an algorithm for the structural index analysis and tools for checking the patterns of the plant model are available in DIVA.

For the numerical integration of the plants DAE, several solvers using different algorithms can be used. Namely, the integrators are SDASSL using a Backward-differentiation-method [2], LIMEXS using an Extrapolation method [3], and RADAU5S [6] and SDIRK4 using Runge-Kutta-methods. A special feature of the SDIRK4 integrator is the simultaneous calculation of sensitivities (derivatives of model states with respect to model parameters).

DIVA also provides different methods for solving systems of nonlinear equations which are needed for the consistent initialization of the integration and the calculation of steady state solutions. The solvers implemented in DIVA are a Levenberg-Marquardt-algorithm (Harwell NS13 [7]) and a Newton-method (NLEQ1s [18]).

Besides these standard numerical methods, DIVA provides routines for event handling. This includes explicit events that are triggered at known times like discrete input functions and implicit events, where the date to trigger the event depends on the model states like bounded outputs of controllers.

Further DIVA offers the possibility of a nonlinear analysis of the plant model. A continuation method using an Euler-Newton predictor-corrector algorithm with local parametrization and stepsize control [12] has been implemented in DIVA. This algorithm provides the possibility to calculate parameter dependent stationary solution branches and thus is an excellent tool to examine the parameter dependent steady state behaviour of the process under consideration. Note that not only stable but also unstable steady state solutions may be calculated by using the continuation method. Moreover, this algorithm includes methods for the stability analysis of steady state solutions and for the detection of bifurcation points, e.g., limit points (cf. Sec. 5).

Algorithms for dynamic optimization problems, like sensitivity analysis, parameter identification, experimental design, and trajectory optimization, are currently under development and have been implemented in a test version [15]. For effective realization of these problems the simultaneous computation of sensitivities during the integration is crucial.

The simulator DIVA is connected in several ways with MATLAB 4.2, a commercial software tool for matrix based systems analysis and synthesis. MATLAB is used for the visualization of simulation results (DIVA-Graphik). Besides, it is possible to generate a linearized plant model in MATLAB format that can be used for control analysis and design methods.

5 Applications

Below some applications from process industries that have been realised in DIVA are given. There are production plants with several process units like a Styrene and a Butylacetate production plant. Further there are single process units like (reactive) distillation processes [17], a novel reactor concept (Circulation Loop Reactor) for air pollution abatement [14], simulated-moving-bed chromatography, liquid chromatography, multiphase reaction processes, liquid-liquid-extraction [22] and cristallization. Some of these process units are modeled as aggregation of subunits, e.g. distillation columns can be aggregated using unit models of column sections, reboilers, condensers, valves, etc..

As an example to highlight the advantages of an integrated tool, a reactive distillation column for the production of Methyl *tert*-Butyl Ether (MTBE) is treated more extensively. The column configuration [11], which has been studied in the literature, is shown in Fig. 2. For this config-

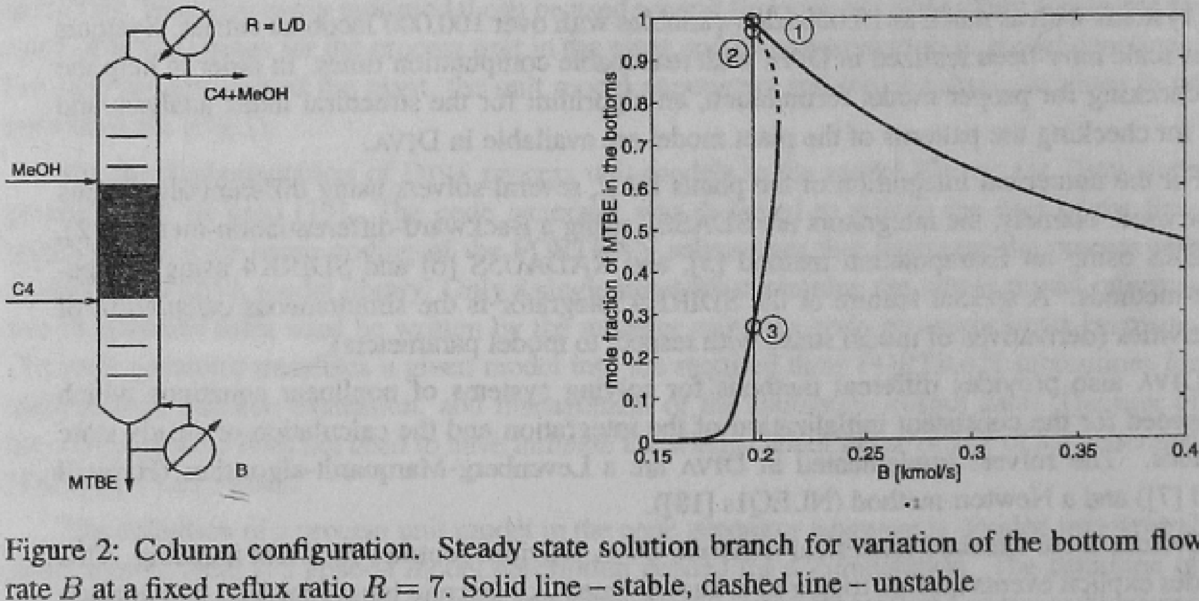

Figure 2: Column configuration. Steady state solution branch for variation of the bottom flow rate B at a fixed reflux ratio $R = 7$. Solid line – stable, dashed line – unstable

uration and the operation conditions given in [11], multiple steady state solutions are obtained as can be seen in Fig. 2, where a solution branch calculated using the continuation algorithm is shown. The three possible solutions for a bottom flow rate $B = 0.197$ kmol/s are marked with circles. From the stability analysis, it is known that the solutions points marked in Fig. 2 as ① and ③ are stable, while ② is unstable.

Using dynamic simulation the domains of attraction for the steady state solutions can be studied, e.g. starting from solution ① a pulse disturbance is introduced to the isobutene (IB) mole fraction in the C4-feed. In Fig. 3 and 4, the transition from the stable solution ① at high mole fraction of desired MTBE in the bottom product to the solution ③ where little MTBE is produced in the column is shown for an increase of IB of 5 % for three hours.

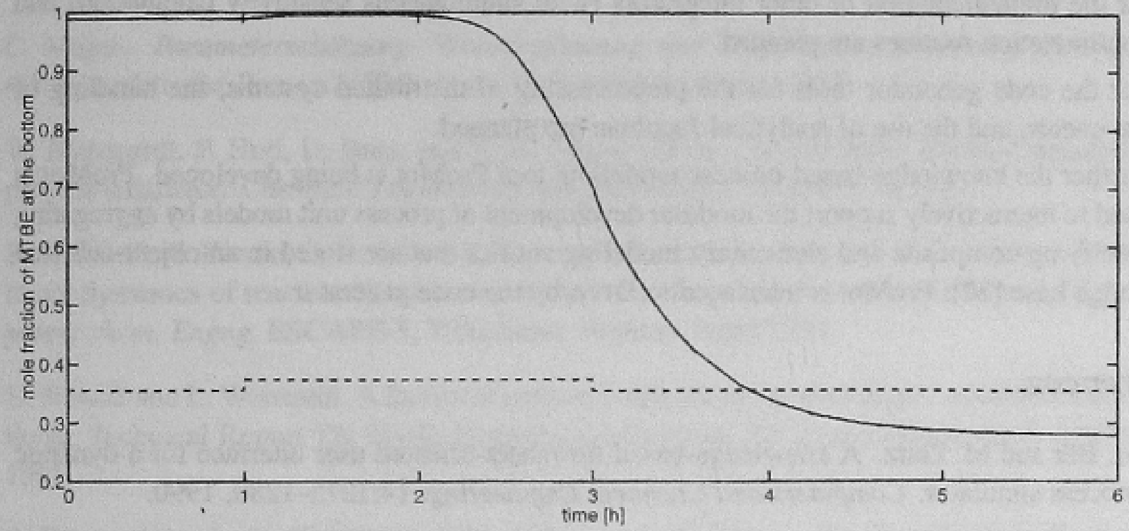

Figure 3: Transition between steady states for a pulse disturbance of the C4 feed. Solid line – mole fraction of MTBE at the bottom, dashed line – IB mole fraction in C4-feed.

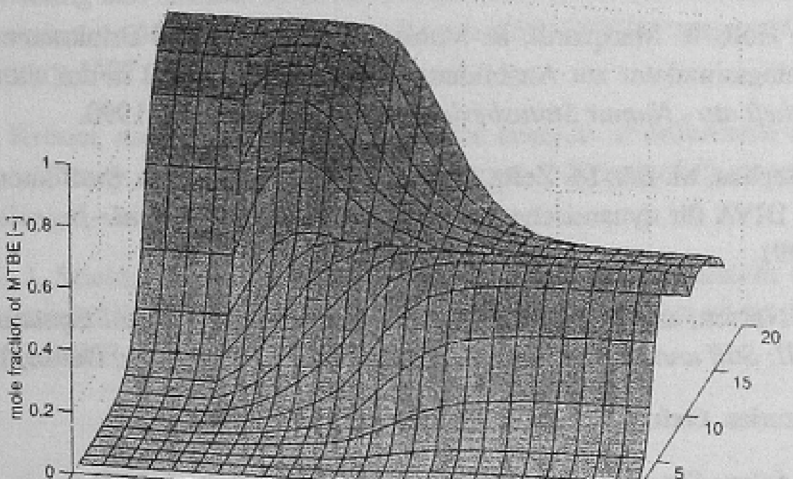

Figure 4: 3D-plot of column profiles for the transition between steady states for a pulse disturbance of the C4 feed.

Besides these chemical engineering problems also control design problems, applications from biochemical engineering and automotive traffic models have been studied using DIVA.

Conclusions and Future Developments

In this paper we have shown the advantages of an integrated and interactive simulation tool, which offers the opportunity to solve various engineering problems in the same environment. This greatly reduces modelling and implementation efforts.

At present an extension of the nonlinear analysis tools with algorithms for the continuation of higher order singularities like limit or Hopf points and periodic solutions is under development. Further the implementation of other integrators (with simultaneous sensitivity calculation) and other optimization routines are planned.

For the code generator tools for the preprocessing of distributed systems, the handling of discrete events, and the use of analytical Jacobian are planned.

Further the knowledge-based process modelling tool ProMot is being developed. ProMot is supposed to interactively support the modular development of process unit models by aggregating and specifying composite and elementary modeling entities that are stored in an object-oriented knowledge base [20]. ProMot is interfaced to DIVA by the code generator.

References

[1] M. Bär and M. Zeitz. A knowledge-based flowsheet-oriented user interface for a dynamic process simulator. *Computers and Chemical Engineering*, 14:1275–1280, 1990.

[2] K. E. Brenan, S. L. Campbell, and L. R. Petzold. *Numerical Solution of Initial Value Problems in Differential-Algebraic Equations*. North Holland, Elsevier Science Publishing Co. Inc., 1989.

[3] P. Deuflhard, E. Hairer, and J. Zugck. One-step and Extrapolation Methods for Differential-Algebraic Systems. *Numer. Math.*, 51:501–516, 1987.

[4] E. D. Gilles, P. Holl, W. Marquardt, R. Mahler, H. Schneider, K. Brinkmann, and K.-H. Will. Ein Trainingssimulator zur Ausbildung von Betriebspersonal in der chemischen Industrie. *Sonderheft atp - Namur Statusbericht '90*, pages 261–268, 1990.

[5] M. Hager, K. Stephan, M. Bär, M. Zeitz, A. Kröner, and E. D. Gilles. Stoffdatenversorgung des Simulators DIVA für dynamische Prozesse und Anlagen. *Chemie-Ingenieur-Technik*, 63:260–261, 1991.

[6] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations; I: Nonstiff Problems, II: Stiff and Differential-Algebraic Problems*. Springer, Berlin, 1987.

[7] Harwell Laboritories, Oxford, England. *Harwell Library*, 1996.

[8] A. Helget. *Modulare Simulation verfahrenstechnischer Anlagen*. PhD thesis, Universität Stuttgart, 1997.

[9] A. Helget and E. D. Gilles. Dynamische Prozeß– und Anlagensimulation. In Hans Schuler, editor, *Prozeßsimulation*, pages 109–148. VCH, 1994.

[10] P. Holl, W. Marquardt, and E. D. Gilles. DIVA - A powerful tool for dynamic process simulation. *Computers in Chemical Engineering*, 12:421–425, 1988.

[11] R. Jacobs and R. Krishna. Multiple Solutions in Reactive Distillation for Methyl *tert*-Butyl Ether Synhesis. *Ind. Eng. Chem. Res.*, 32(8):1706–1709, 1993.

[12] A. Kienle, G. Lauschke, V. Gehrke, and E. D. Gilles. On the dynamics of the circulation loop reactor – Numerical methods and analysis. *Chem. Engng. Sci.*, 50(15):2361–2375, 1995.

[13] A. Kröner, P. Holl, W. Marquardt, and E. D. Gilles. DIVA – an open architecture for dynamic simulation. *Computers and Chemical Engineering*, 14:1289–1295, 1990.

[14] G. Lauschke and E. D. Gilles. Circulating reaction zones in a packed-bed loop reactor. *Chem. Ing. Sci.*, 49:5359–5375, 1994.

[15] C. Majer. *Parameterschätzung, Versuchsplanung und Trajektorienoptimierung für verfahrenstechnische Prozesse.* PhD thesis, Universität Stuttgart, 1997.

[16] W. Marquardt, P. Holl, D. Butz, and E. D. Gilles. DIVA - A flow-sheet oriented dynamic process simulator. *Chemical Engineering and Technology*, 10:164–173, 1987.

[17] K. D. Mohl, A. Kienle, E. D. Gilles, P. Rapmund, K. Sundmacher, and U. Hoffmann. Nonlinear dynamics of reactive distillation processes for the production of fuel ethers. In *Computers chem. Engng.* ESCAPE-7, Trondheim, Norway, April 1997.

[18] U. Nowak and L. Weimann. A family of newton codes for systems of highly nonlinear equations. Technical Report TR 91-10, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 1991.

[19] S. Räumschüssel, A. Gerstlauer, E.D. Gilles, and M. Zeitz. Ein Präprozessor für den verfahrenstechnischen Simulator DIVA. In G. Kampe and M. Zeitz, editors, *Simulationstechnik, 9. ASIM-Symposium in Stuttgart*, pages 177–182. Vieweg Verlag, Braunschweig/Wiesbaden, 1994.

[20] F. Tränkle, A. Gerstlauer, M. Zeitz, and E. D. Gilles. PROMOT/DIVA: A prototype of a process modeling and simulation environment. In I. Troch and F. Breitenecker, editors, *IMACS Symposium on Mathematical Modelling, 2nd MATHMOD*, pages 341–346, Vienna, Austria, February 1997. ARGESIM Report.

[21] J. Unger, A. Kröner, and W. Marquardt. Structural analysis of differential-algebraic equation systems – Theory and applications. *Computers and Chemical Engineering*, 19(8):867 – 882, 1995.

[22] G. Zamponi, J. Stichlmair, A. Gerstlauer, and E. D. Gilles. Simulation of the transient behavior of a stirred liquid-liquid extraction column. In *Computers chem. Engng.*, volume 20, Suppl., pages 963–968, April 1996.