Weierstraß–Institut
für Angewandte Analysis und Stochastik

im Forschungsverbund Berlin e.V.

Preprint ISSN 0946 – 8633

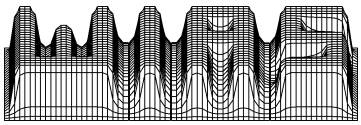
Heterogeneous Dynamic Process Flowsheet Simulation of Chemical Plants

Friedrich Grund, Klaus Ehrhardt, Jürgen Borchardt, Dietmar Horn

submitted: 10th May 2000

Weierstrass Institute
for Applied Analysis
and Stochastics
Mohrenstrasse 39
10117 Berlin
Germany
E-Mail: grund@wias-berlin.de
E-Mail: ehrhardt@wias-berlin.de
E-Mail: borchardt@wias-berlin.de
E-Mail: horn@wias-berlin.de

Preprint No. 576
Berlin 2000



1991 *Mathematics Subject Classification.* 65L05,80A30,65Y05,65H10.

Key words and phrases. Chemical process simulation, Systems of differential–algebraic equations, Large-scale dynamic simulation, Coupled processes, Distributed simulation, Waveform iteration, Broyden update.

This work was supported by the Federal Ministry of Education, Science, Research and Technology, Germany under grant GR7FV1.

Edited by
Weierstraß–Institut für Angewandte Analysis und Stochastik (WIAS)
Mohrenstraße 39
D — 10117 Berlin
Germany

Fax: + 49 30 2044975
E-Mail (X.400): c=de;a=d400-gw;p=WIAS-BERLIN;s=preprint
E-Mail (Internet): preprint@wias-berlin.de
World Wide Web: http://www.wias-berlin.de/

Abstract. For large-scale dynamic simulation problems in chemical process engineering, a heterogeneous simulation concept is described which allows to distribute the solution of the models of coupled dynamic subprocesses to a computer network. The main principle of such a technique is to solve the submodels of an overall model independently of each other on subsequent time intervals. This is done by estimating the vector of input variables of the submodels, calculating the corresponding time behaviour of the output variables concurrently, and matching the time profiles of the interconnecting variables of the process flowsheet iteratively. Therefore, accelerated waveform iteration methods are considered, using Broyden– and block–Broyden–type updates. The simulation concept is investigated especially in the case that the submodels do not provide input-output sensitivities.

1 Distributed Simulation of Coupled Processes

The numerical simulation of dynamic processes for large-scale plants in chemical industry requires a strongly modular division of modelling tasks with respect to the design of sufficiently encapsulated subprocess models, their mathematical formulation and numerical solution. In [3] - [6] we have shown how such a modular modelling can be used to solve homogeneous dynamic simulation problems for complex industrial distillation plants, achieving considerable speedup factors on parallel computers with shared memory. But the modular modelling concept is even more significant in the case of heterogeneous simulation problems, where different co-workers, possibly working with different simulation tools on different computer platforms, have developed well working codes for subprocesses, which have now to be coupled with other subprocesses to simulate plantwide dynamic process flowsheets. An example for such a process flowsheet, assembled from 3 subprocesses, is shown in Fig. 1.

A general approach of coupling p subprocesses can be derived in the case that each submodel i, which represents a specific subprocess of the entire plant, is determined by the submodel function which uniquely generates an output function $v_i(t)$ for each given input function $u_i(t)$, i.e.

$$v_i(t) = G_i(u_i(t)), \quad i = 1, \ldots, p \quad t \in [t_0, t_E]. \tag{1}$$

Here the submodel functions G_i are only given implicitly by applying a numerical solution procedure to the respective submodel equations. Frequently,

the internal model of subprocess i is described by a large system of differential algebraic equations (DAEs)

$$F_i(t, y_i(t), \dot{y}_i(t), u_i(t)) = 0, \qquad (2)$$
$$y_i(t_0) = y_i^0,$$

with $y_i = (x_i, v_i)^T$. These internal submodel equations (2), the dimension of the resulting discretized model as well as their numerical implementation are not known outside the subprocess simulation. But it is worth mentioning that the dimensions of the vectors of input and of output variables u_i and v_i are usually small compared with the large number of internal variables x_i.

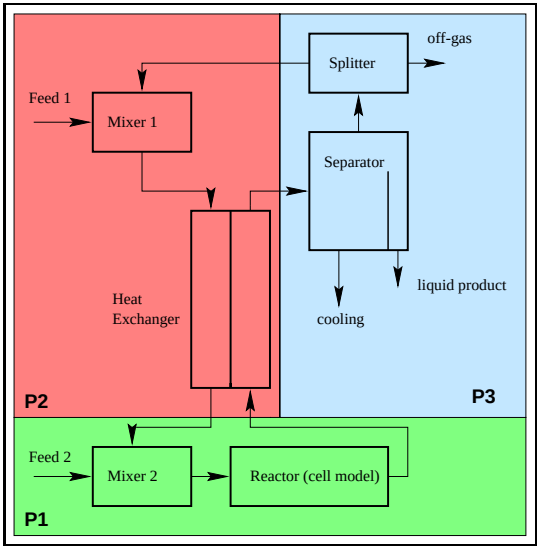


Fig. 1. Process flowsheet assembled from 3 subprocesses

In a global process flowsheet definition, each component of the overall input vector $u(t) = (u_1, ..., u_p)^T$ corresponds uniquely to one component of the overall output vector $v(t) = (v_1, ..., v_p)^T$. All feed and energy streams entering the plant from the outside as well as the products, byproducts, and energy streams leaving the plant are balanced within the appropriate submodels, so that they do not need to be considered here. Therefore, the system of equations describing the coupling between the submodels can be defined by

$$f\left(u(t)\right) \equiv u(t) - PG\left(u(t)\right) = 0, \quad t \in [t_0, t_E], \qquad (3)$$

where $P = ((p_{ij}))$, with $p_{ij} \in \{0, 1\}$, is a permutation matrix which allocates the individual input and output variables of all subprocesses, and G represents all submodel functions G_i, cf. (1).

A self-evident extension of the coupling equations (3) results from the assumption of a time-delay in submodel coupling which may be described by some simple residence-time model of material and heat flow within the interconnecting pipes like

$$\omega \frac{du(t)}{dt} = PG\left(u(t)\right) - u(t), \quad t \in [t_0, t_E] \tag{4}$$

or more sophisticated plug-flow assumptions. This approach is not exploited directly within this paper, because it is expected that such effects have to be modelled within the related subprocess models only. But the equation (4) can be used to derive a natural relaxation strategy with $\omega \to 0$ for the iterative solution of (3) as it will be shown later on.

In order to investigate the dynamics of the coupling equations (3) in a strongly modular manner, one has to guarantee that within each submodel i there is no direct feedback of outputs $v_i(t)$ to its own inputs $u_i(t)$. The behaviour of the time-dependent system (3) is then mainly determined by the corresponding Jacobian

$$J = I - P\frac{\partial G}{\partial u}, \tag{5}$$

where $\partial G/\partial u$ denotes the overall block diagonal matrix of the dynamic input–output–sensitivities $\partial G_i/\partial u_i$ of the submodels. Generally, these dynamic sensitivities are not provided by todays commercial simulation tools. Even an initial estimation of the Jacobian at t_0 will be difficult to obtain with reasonable effort. Additionally, discontinuities of the solution and of parameter functions may arise in subprocesses at predefined time points or sporadically. Furthermore, a truly heterogeneous process simulation environment may require to handle different submodels on different computer platforms and to retain proven software for submodel solution with its own internal step size and accuracy control. Under these stringent modelling restrictions the subsequent algorithmic details for solving (3) have to be assessed.

2 Waveform Iteration Methods

It is obvious that for the distributed simulation of coupled subprocesses an adapted waveform iteration method, cf. [8] and [13], can be used. The basic idea of waveform iteration is to solve submodels of an overall model independently of each other on subsequent time intervals, so called windows. For that, on the current window, the time behaviour of the vector of input variables of the submodels is estimated, the corresponding time behaviour of the output variables is computed concurrently for all submodels, and the interconnecting variables of the flowsheet are matched iteratively. An overview of this approach with applications in chemical engineering and a discussion of preparatory literature has been given in [11].

2.1 The Discretized Problem

Due to the need to simplify communication and synchronization between subprocess simulations, the time variable $t \in [t_0, t_E]$ is discretized within each window $[t_0^s, t_m^s]$ equidistantly. With $t_0^0 = t_0$, one gets for the s'th window of length $T^s = t_m^s - t_0^s$

$$t_j^s = t_m^{s-1} + j * T^s/m, \qquad j = 0, 1, \ldots, m. \tag{6}$$

A sufficient degree of adaptivity is still preserved by a flexible window length T^s as well as by the possibility to reduce the window size, after some iteration, by shifting the left boundary of the window from t_0^s to some $t_l^s, 0 < l < m$, if convergence is achieved for all $t_j^s, j = 0(1)l$ already.

Keeping in use the same symbols for notational simplicity, now u_i, and v_i denote the vectors of all discretized input and output variables at all internal time points t_j^s, $j = 0, \ldots, m$ of the current window in an appropriate order, and u and v the related overall vectors $u = (u_1, ..., u_p)^T$ and $v = (v_1, ..., v_p)^T$, respectively.

2.2 Iterative solution of coupling equations

A Picard–type iteration, which solves (3) at the grid points $t_j^s, j = 0(1)m$, of the current window, is given by

$$v_i^{k+1} = G_i(u_i^k), \quad i = 1, ..., p \tag{7}$$

$$u^{k+1} = P v^{k+1}, \tag{8}$$

where k denotes the iteration index. Obviously, the evaluation of (7) can be done in parallel for all subprocesses, followed by a common step (8) of allocating the output variables to the related inputs. It is noteworthy to state that it is nowhere required at all that the elements of u and v have to be discrete function values, but instead they can be parameters of an appropriate continuous approximation of the trajectories of the interconnecting variables. In this paper we use a linear interpolation in order to get continuous piecewise linear approximations of the inputs u_i in (7). This is realized by calling an additional interpolation subroutine within each function evaluation of the associated systems (2). It avoids the permanent reinitialization of the integration procedure, as it has to be done in the case of piecewise constant inputs, and increases the approximation accuracy.

To accelerate the convergence of the Picard iteration, (8) can be substituted by a quasi-Newton approach

$$u^{k+1} = u^k - \left(B^k\right)^{-1} \left(u^k - PG(u^k)\right), \tag{9}$$

where B^k should be an approximation of the Jacobian, preferably

$$I - PG_u^k. \tag{10}$$

Here G_u^k represents a reasonable approximation of the overall block diagonal matrix of the dynamic input-output sensitivities of submodels at all internal time points of the current window. Generally, a trusted estimation of G_u^k, even at t_0, will be difficult to get. Therefore, we have been looking for an update procedure in the case that no initial estimation is available, i.e. the iteration starts with an iteration matrix $B^0 = I$, as it is discussed in [12], or in the case that only some of the submodels provide such information.

Broyden update: Numerous versions and modifications of Broyden update formulas have been derived in the past, mainly focussed on improving a given iteration matrix in (9) by calculating an adjacent matrix which still satisfies the quasi-Newton condition with the most recent function values, cf. [9], [12],

$$B^{k+1}\Delta u^k = \Delta f^k. \tag{11}$$

In this paper the classical Broyden method is adapted to the investigated waveform iteration technique and a possible extension to it is discussed.

With abbreviations

$$\Delta u^k \equiv u^{k+1} - u^k \tag{12}$$

$$\Delta v^k \equiv v^{k+1} - v^k \tag{13}$$

$$\Delta f^k \equiv f^{k+1} - f^k = \Delta u^k - P\Delta v^k \tag{14}$$

the conventional Broyden approach

$$B^{k+1} = B^k - \frac{(B^k\Delta u^k - \Delta f^k)(\Delta u^k)^T}{(\Delta u^k)^T\Delta u^k}, \quad (\Delta u^k)^T\Delta u^k \neq 0 \tag{15}$$

is applied even in the case that no initial estimate of the sensitivity matrix is available at all, i.e. $G_u^0 = 0$ or $B^0 = I$. Obviously, this update strategy (15) can be applied to G_u^k directly to get

$$G_u^{k+1} = G_u^k\left(I - \frac{\Delta u^k(\Delta u^k)^T}{(\Delta u^k)^T\Delta u^k}\right) + \frac{\Delta v^k(\Delta u^k)^T}{(\Delta u^k)^T\Delta u^k}, \quad (\Delta u^k)^T\Delta u^k \neq 0. \tag{16}$$

As demonstrated later on, this update strategy improves the iteration (9) significantly even in the case $G_u^0 = 0$.

Block-Broyden update: In the case that at least some submodels provide input-output sensitivity information, with

$$G_{u_i}^{k+1} = G_{u_i}^k\left(I - \frac{\Delta u_i^k(\Delta u_i^k)^T}{(\Delta u_i^k)^T\Delta u_i^k}\right) + \frac{\Delta v_i^k(\Delta u_i^k)^T}{(\Delta u_i^k)^T\Delta u_i^k}, \quad (\Delta u_i^k)^T\Delta u_i^k \neq 0, \tag{17}$$

a block-Broyden update of (10) is proposed which maintains the global block pattern, but is still in conflict with the sparsity pattern of sensitivity submatrices, cf. also [16]. Here $G_{u_i}^k$ denotes the i'th diagonal block of G_u^k and

$\Delta v_i^k = v_i^{k+1} - v_i^k$ the vector of output variations of the submodel calculated for an input change $\Delta u_i^k = u_i^{k+1} - u_i^k$, at all discrete time points of the current window.

The denominator $(\Delta u_i^k)^T \Delta u_i^k$ instead of $(\Delta u^k)^T \Delta u^k$ ensures that this update is approximating the dynamic sensitivity matrix of submodel i independently of the input changes Δu_j^k $(j \neq i)$ of the other submodels. Additionally, the local condition

$$G_{u_i}^{k+1} \Delta u_i^k = \Delta v_i^k \tag{18}$$

is fulfilled, which gives a finite difference approximation of the sensitivity matrix of the corresponding submodel in the case of an orthogonal sequence Δu_i^k, $(k = 1, 2, ...)$.

Relaxation Strategies: In the case of smooth trajectories, a relaxation strategy can be introduced by formally discretizing the time delayed coupling equations (4) with an implicit Euler formula simultaneously for all internal grid points of the current window. This gives a quite natural relaxation method

$$u^{k+1}(t_j^s) = (1 - \lambda_j) u(t_0^s) + \lambda_j PG(u^k), \quad \lambda_j = \frac{1}{1 + \frac{m\omega}{jT^s}}, \quad j = 1(1)m \tag{19}$$

where the desired solution is obtained for $\omega \to 0$, i.e., $\lambda_j \to 1$. This embedding technique is especially recommended if a submodel fails to initialize the integration procedure at a given inlet stream variation. Additionally, a component specific relaxation parameter ω can be introduced, if an experienced user knows internal dependencies of the process. With

$$u^{k+1} = u^k - \left(I - \Lambda PG_u(u^k)\right)^{-1} \left[u^k - \Lambda PG(u^k) - (I - \Lambda)u^0\right], \tag{20}$$

a damped version of the iteration (9) results with a diagonal relaxation matrix $\Lambda = \Lambda(\omega)$, with $\lim_{\omega \to 0} \Lambda(\omega) = I$.

3 Implementation and Results

This project is primarily intended to develop the algorithmic details of an improved waveform iteration method for large-scale dynamic process simulation. The implementation of a communication software for distributed simulation tasks are beyond the scope of this project, because our cooperation partners at Bayer AG, Leverkusen, have already developed such a software tool called *Simulation Manager* [7]. This tool is mainly used for distributed simulation tasks of stationary process models, but it also allows the distributed dynamic simulation based on a simple Picard–type iteration (7,8) with a piecewise constant approximation of the interconnecting variables. A graphical user interface is provided for deriving the set of coupling equations by drawing the

process flowsheet and for communication and control of subprocesses which can run on different computer platforms using different commercial simulation tools. We have successfully implemented dynamic processes within this environment by using the commercial simulator SPEEDUP™[2] to formulate and solve the subprocesses. Additionally, an own software environment has been implemented which is still limited to the case of spreading the subprocesses within a network of coupled workstations with a common NFS file system. This allows a more rapid testing and an immediate access to the model equations of subprocesses. The following examples have been tackled within both environments by using our block–oriented process simulator BOP [5] to solve the submodels, or the simulator SPEEDUP™ within the *Simulation Manager* tool, respectively.

Details of a large-scale model of three interconnected reactive distillation columns have been reported previously [10]. Here, each subprocess model consists of 600 cell models with different parameter settings, which results into an DAE system with an overall number of 45 600 differential algebraic equations. Each submodel consists of 15 200 DAEs while the total number of connecting variables is only 40. This is not an untypical ratio in industrial applications. Although the process flowsheet of this model has non–sequential reflux streams, the iteration of the Picard–type method converges quite fast, and the Broyden update does not give a significant additional progress. Obviously, the number of iterations per window and the inlet stream variations are too small, so as to alter the initial iteration matrix significantly.

A different behaviour has been observed for the process flowsheet in chapter 1 (Fig. 1), where a reactor model (Subprocess P1, 507 DAEs) is interconnected with a heat-exchanger (P2, 779 DAEs) and a simplified separation unit (P3, 48 DAEs). The plant serves for the production of 1,2-dicloroethane. Reactants are pure chlorine and ethene diluted by ethane. The exothermal reaction is performed in a tubular plug-flow reactor. To get a sufficient reaction rate, the fresh reaction mixture is preheated by the hot reactor output stream in a counter-current manner. Afterwards the reactor output stream

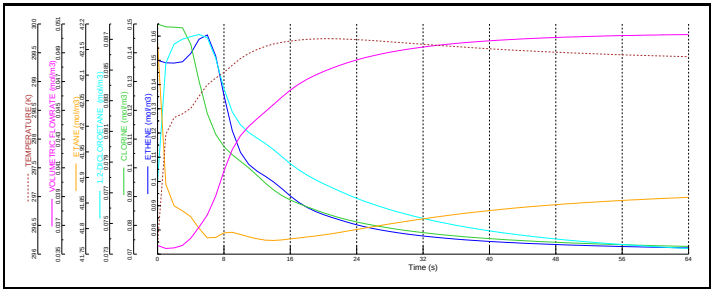


Fig. 2. Trajectories of stream P3 → P2

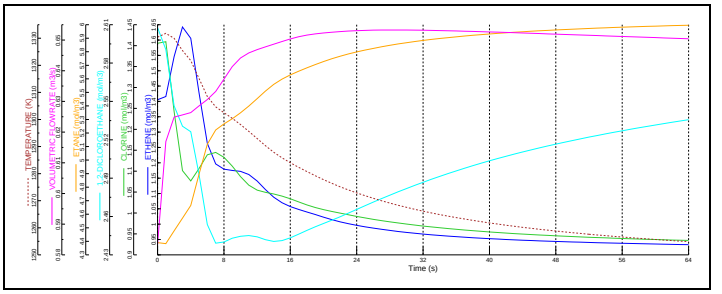


Fig. 3. Trajectories of stream P1 → P2

is further cooled down in the separator (P3) to condense the desired product which leaves the condenser as liquid. The remaining gas stream is than partly returned to the input of the heat exchanger (P2). By varying input concentrations, flow rates or temperatures of both feed streams as well as by changing internal parameters, the model allows to perform different dynamic scenarios like ignition or extinguishing of the reaction as well as very rapid transitions between steady states. The exemplary curves in Fig. 2, 3 have been achieved by switching the recycle ratio within the splitter in P3. Fig. 2 represents the resulting connecting stream variables between P3 and P2, while Fig. 3 shows the dynamics of the corresponding product stream leaving the rector P1 towards the heat exchanger P2. The presented scenario shows rapid changes in the very beginning and a smooth tail towards the new steady state, and is therefore well suited for testing purposes. In both figures a window length of 8 is indicated while the inner grid size of the windows is equal to 1 for the linear approximation of all trajectories.

Table 1 shows the comparison of the Picard–type iteration (7)–(8) versus its modification with Broyden update (16) at a constant window length of 16 seconds. In fact, the number of iterations, i.e. the number of subsequent subprocess integration steps, is significantly reduced by Broyden update in the dynamic part of the trajectories (window 1) compared with the smoother tail afterwards. Furthermore, comparative calculations have been

Window number	1	2	3	4	1–4
Time interval (sec.)	0–16	16–32	32–48	48–64	0–64
Number of iterations:					
without Broyden update	39	29	23	18	109
with Broyden update	23	20	14	13	70

Table 1. Convergence acceleration of Picard–type iteration by Broyden update

done within the simulation interval $t \in [0, 64]$ by choosing different window lengths of 1, 2, 4, 8 16, and of 32 seconds to demonstrate the even more dramatic influence of the length of the time interval on the total number of iterations. Fig. 4 shows the cumulative number of iterations needed with Broyden update (16) and applying this windowing strategy on the same subgrid, in order to achieve a comparable approximation accuracy. Apparently, the number of iterations is reduced remarkably to less than 10% by increasing the length of the time window from 1 to 32 seconds. This is mainly caused by the fact that the number of coupling systems to be solved iteratively as well as their dimensions are changing significantly with the window length.

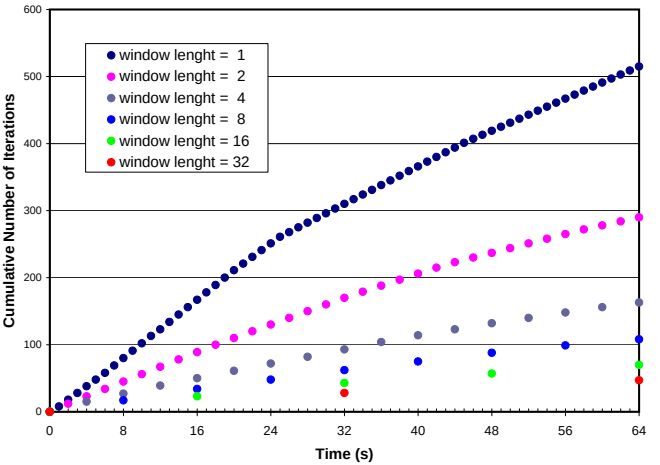


Fig. 4. Influence of the window length on the number of iterations

The preceding results reveal the inherent potential of a heterogeneous modelling approach based on an appropriate waveform iteration technique. Basic requirements are a correct modelling of the input-output behaviour of each subprocess over small time intervals, as well as a continuous at least piecewise linear approximation of all trajectories of the connecting variables in order to avoid a permanent re-initialization during submodel integration. A well adapted Broyden update strategy may accelerate the solution procedure. Furthermore, the presented approach allows the modular treatment of subprocess model formulation as well as their parallel numerical solution.

Acknowledgements We thank Bayer AG, Leverkusen, for the valuable support and Aspen Technology, Inc. for providing us a free SPEEDUP™ license for academic use. The assistance of P. Čapek, Institute of Chemical Technology Prague, for test model formulation is kindly acknowledged.

References

1. Arnold, M., Günther, M.: Preconditioned dynamic iteration for coupled differential-algebraic systems. DLR German Aerospace Center, Institute of Aeroelasticity, Technical Report IB 532-99-01 (1999) 1-21
2. AspenTech: SPEEDUP™ Release 5.5-6, User Manual, Library Manual. Aspen Technology, Inc., Cambridge, Massachusetts, USA (1997)
3. Borchardt, J.: Parallel Numerical Methods for Large-Scale DAE Systems. In: Proceedings of AspenWorld 2000, Orlando, USA (2000) 1-19
4. Borchardt, J., Grund, F., Horn, D.: Parallelized Methods for Large Nonlinear and Linear Systems in the Dynamic Simulation of Industrial Applications. Surveys on Mathematics for Industry **8** (1999) 201–211
5. Borchardt, J., Ehrhardt, K. Grund, F., Horn, D.: Parallel Modular Dynamic Process Simulation. In: Keil, F. et al. (eds.), Scientific Computing in Chemical Engineering II, Springer, Berlin (1999) 152–159
6. Borchardt, J.: Parallelized Block–Structured Newton–Type Methods in Dynamic Process Simulation. In: B. Kågström et al. (eds.), Applied Parallel Computing, Lecture Notes in Computer Science 1541, Springer, Berlin (1998) 38–42
7. Brüll, L.: Simulation manager: A software tool for process simulation in a heterogeneous hard- and software environment. SPEEDUP Journal **10** (1996) 51–53
8. Burrage, K.: Parallel and sequential methods for ordinary differential equations. Clarendon Press, Oxford (1995)
9. Dennis, J.E. and Schnabel, R.B.: Numerical Methods for Unconstrained Optimization and Nonlinear Equations. SIAM Series: Classics in Applied Mathematics **16**, Philadelphia (1996)
10. Ehrhardt, K., Borchardt, J., Grund, F., Horn, D.: Divide and Conquer Strategies in Large Scale Dynamic Process Simulation. Comput. Chem. Engrg. **23**, Supplement (1999) S335–S338
11. Helget, A.: Modulare Simulation verfahrenstechnischer Anlagen. Fortschritt-Berichte VDI Reihe 20, Nr. 251, Düsseldorf VDI Verlag (1997)
12. Kelley, C. T.: Iterative Methods for Linear and Nonlinear Equations. SIAM Series: Frontiers in Applied Mathematics **16**, Philadelphia (1995)
13. Lelarasmee, E., Ruehli, A.E., Sangiovanni–Vincentelli, A.L.: The Waveform Relaxation Method for Time–Domain Analysis of Large Scale Integrated Circuits. IEEE Trans. on CAD of Integrated Circuits and Systems **1** (1982) 131–145
14. Paloschi, J.R., Zitney, S.E.: Parallel Dynamic Simulation of Industrial Chemical Processes on Distributed-Memory Comput. Chem. Engrg. **23**, Supplement (1999) S395–S398
15. Secchi, A.R., Morari, M., Biscaia Jr., E.C.: The Waveform Relaxation Method in the Concurrent Dynamic Process Simulation. Comput. Chem. Engrg. **17** (1993) 683–704
16. Yang, G., Dutto, L. C., and Fortin, M.: Inexact block Jacobi-Broyden methods for solving nonlinear systems of equations. SIAM J. Sci. Comput. **18** (1997) 1367–1392