Distributed Models in Plantwide Dynamic Simulators

W. S. Martinson and P. I. Barton
Dept. of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139

Modeling support for dynamic simulation of chemical-process flowsheets, which is of significant value for plantwide dynamic simulation using differential–algebraic model formulations, is to date very limited when one or more unit models include partial differential equations. Several new techniques that provide modeling support for such simulations are presented. These techniques are based on a generalized characteristic analysis and a differentiation index analysis of partial differential–algebraic models. They can be used to uncover systems that cannot be solved as part of a dynamic simulation, and to determine whether or not the initial and boundary conditions supplied by the modeler form a well-posed problem. In a network flow context, they can further be used to select, enforce, and adapt the boundary conditions as required to maintain automatically a mathematically well-posed problem. Each of these provides time-saving support to the system modeler.

Introduction

Significant research over the past twenty years has produced several highly developed software packages that are designed specifically for plantwide dynamic simulation. Examples include SpeedUp (Perkins and Sargent, 1982), DIVA (Marquardt et al., 1987), gPROMS (Barton and Pantelides, 1994), and ABACUSS (Allgor et al., 1996). These packages facilitate large-scale system simulation by isolating the engineer from numerical algorithms, code generation, and debugging, thereby leaving him or her free to concentrate on model formulation and application. The feasibility of dynamic simulation-based activities such as optimal batch policy synthesis, parametric sensitivity studies, safety interlock design verification, control system design, and start-up/changeover/shutdown studies in an industrial setting often depends on the productivity gains provided by this modeling support (Longwell, 1993).

These existing packages deal very effectively with the differential–algebraic equation (DAE) formulation of a plant model, which arises when spatial variations of dependent variables in the unit operations are ignored. This formulation admits a fairly general dynamic description of a chemical processing system. Given such a model, a modern simulation package typically advances the solution to the entire flowsheet using a single integration method [such as DASSL (Petzold, 1982), DASOLV (Jarvis, 1992), or the DASSL variant DSL48S (Feehery et al., 1997), which employ a Gear-type variable stepsize, variable-order BDF method)]. Modern software packages also automatically perform calculations to, for example, locate state events/implicit discontinuities (Park and Barton, 1996), reinitialize the system after a state event (Mayer et al., 1995), and integrate high-index systems (Feehery and Barton, 1996).

Sometimes a finer degree of detail is required than can be provided by the DAE, or lumped, formulation. For example, when the spatial variations of some dependent variables across a processing unit are important, that unit must be described by partial differential equations. Similarly, population balances or polymer chain-length distributions in an otherwise lumped system also give rise to partial differential equations. A model that includes partial differential equations, and possibly ordinary differential and algebraic equations as well, is referred to as a partial differential–algebraic equation (PDAE), or distributed model.

Efforts to provide modeling support for distributed models in network simulations have so far focused on generating and analyzing a discretization of the partial differential equations. Oh and Pantelides have addressed semidiscretization in the context of network flow simulations by developing an input language for generating discrete equations on rectangu-

Correspondence concerning this article should be addressed to P. I. Barton.
Present address of W. S. Martinson: Cargill Central Research, Cargill Inc., Minneapolis, MN 55440.

lar domains (Oh and Pantelides, 1996). This input language forms part of the larger gPROMS dynamic simulation software mentioned earlier. The language allows the user to define spatial coordinate axes and represent derivatives in each coordinate direction using either finite differences or collocation on a monospaced grid along each axis. gPROMS then uses the specified grid and method to substitute an approximating set of DAEs for the original PDEs, couples them to the rest of the flow-sheet equations, and solves the entire system in the same manner as a purely lumped model.

TRIFIT is a mesh-generation language (van der Wijngaart, 1994). It consists of a set of operators for several common manipulations of unstructured triangular two-dimensional grids. These operators can be used to refine or smooth such grids. Discrete approximations to partial differential equations can also be expressed compactly using the operators. The language, like the mesh-generation syntax in gPROMs, is designed to reduce the time required to generate a discrete scheme for solving systems of PDEs. However, it is not integrated into a large simulation environment.

The GRIDOP package (Liska and Shashkov, 1991; Liska et al., 1994) provides similar tools for generation of conservative finite difference schemes on logically rectangular domains in an arbitrary number of independent variables. The package takes as input a user-supplied definition of function spaces and associated scalar products, together with user-supplied definitions of grid operators as finite difference schemes. The user may then provide partial differential equations in terms of the defined grid operators or the adjoints of those operators, and the package returns the finite difference equations.

These tools are all designed to generate or evaluate a discretization scheme for solving a distributed model. The eventual goal is for these tools and others like them to advance to the point where an engineer simply provides a distributed model, and the simulator will generate a numerical solution along with rigorous guarantees of its accuracy. This would correspond to the current capabilities of process simulators for dealing with lumped models.

One step on the path toward such full support of distributed models by dynamic process simulators is automated screening for models that the simulator cannot solve. It is somewhat unreasonable to expect a simulator to solve a model that is mathematically ill-posed, for example. Similarly, DAEs that are high index are not amenable to numeric integration by standard integration codes. This article will focus on ways for a process simulator to examine models generated by an engineer and identify ones that are mathematically ill-posed or may be expected to lead to high-index DAEs in method-of-lines solution techniques.

Possibly the first step toward development of a tool of this nature was taken by Marquardt and coworkers, who have demonstrated PDEDIS, a software package for rapid construction and evaluation of method-of-lines semidiscretizations for one-dimensional PDE systems (Pfeiffer and Marquardt, 1993). The software accepts as input a system of at most second-order spatial derivatives and first-order time derivatives. It can symbolically discretize the spatial derivative terms using either a finite difference or a weighted residual method, retaining grid spacings or function weights as unknowns. This symbolic form can be easily evaluated, given a grid and values for any parameters and dependent variables required to calculate the coefficient matrices. This information is then output to a file and submitted to MATLAB, where temporal eigenvalues are calculated. Any positive real part of an eigenvalue indicates an unstable discretization.

As part of the preprocessing capabilities, PDEDIS is able to characterize the system as hyperbolic, parabolic, or elliptic; provide characteristic directions for purely hyperbolic systems; identify self-adjoint spatial operators; and perform several other classifications of the equations. This information allows consistency of the model to be evaluated, although only basic consistency checks (which are not detailed in the article) are implemented. However, this automated analysis of the model equations as provided by the engineer is precisely the type of technology that will be explored and developed in this article.

The question of well-posedness of systems of parabolic and hyperbolic type is very well understood (Courant and Hilbert, 1962). Recent work has extended some of the classic analysis to more general linear PDAEs of neither hyperbolic nor parabolic type, that may include purely algebraic equations (Martinson and Barton, 2001). The notion of the index of differential–algebraic systems has evolved steadily over the past decade, and is also well understood (Campbell, 1982; Brenan et al., 1989). The concept of the index of a system of partial differential equations has been the subject of a growing amount of research (Campbell and Marszalek, 1997; Martinson and Barton, 2000).

The article will begin with several motivating examples of difficulties with particular dynamic simulation problems, followed by a very brief review of some current work in the areas of index analysis and well-posedness of PDAEs. Calculations that can be performed by a simulator and the implications of the results will then be presented. These calculations will be applied to the examples in the following section, and the results examined. The article will conclude with a discussion of directions for future work.

Motivating Examples

The following examples illustrate some of the difficulties that may arise when trying to perform a dynamic simulation using models that involve partial differential equations. The flowsheet sections are simple and are chosen to illustrate specific problems; they are not intended to be large-scale case studies.

Pressure-swing adsorption

Consider greenhouse gas removal from a nitrogen gas stream by a two-column pressure-swing adsorption process. Part of the process flowsheet appears in Figure 1. A continuous high-pressure feed to the system is directed through one of the columns, where greenhouse gases are removed from the nitrogen stream by adsorption onto a zeolite packing. At the same time, a low-pressure nitrogen stream is blown through the other columns to remove the adsorbed species and carry them to another treatment unit. When the packing in the high-pressure column approaches saturation, the high-pressure feed is switched over to the second column,

and the low-pressure stream is switched to the first column. The process in repeated.

Here the engineer's overall task is to improve the operating policy for the process, using dynamic simulation for as much preliminary work as possible, because the system cannot be taken off-line without major expense. Laboratory experiments have provided good values for the parameters in the Kikkinides and Yang model of pressure-swing adsorption processes (Kikkinides and Yang, 1991), which describes column behavior under the assumptions of isothermal operation, negligible axial dispersion and pressure drop, plug flow, instantaneous solid–gas phase equilibrium, and perfect gas behavior, all of which are judged to be reasonable for this process.

Under this model, the adsorbate concentration on the solid $q_{i=1\ldots3}$, mole fractions in the gas phase of adsorbate $y_{i=1\ldots3}$ and inert y_4, and flow velocity u are related by the following system of equations over time t and axial position in the absorber z. Pressure P, pressurization rate P_t, temperature T, bed void fraction ϵ, bed density ρ_B, gas constant R, saturation loadings $q^{\text{sat}}_{i=1\ldots3}$, and load-relation correlation constants $n_{i=1\ldots3}$ and $B_{i=1\ldots3}$ are parameters. The values of these parameters have been experimentally validated for this process

$$\frac{\rho_B RT}{P}\sum_{i=1}^{3} q_{i_t} + \frac{\epsilon}{P}P_t + u_z = 0$$

$$\epsilon y_{i_t} + \frac{\rho_B RT}{P}q_{i_t} + \frac{\epsilon y_i}{P}P_t + (uy_i)_z = 0, \qquad i = 1\cdots3$$

$$\sum_{i=1}^{4} y_i = 1$$

$$q_i - \frac{q_i^{\text{sat}} B_i (y_i P)^{1/n_i}}{1 + \sum_{j=1}^{3} B_j (y_j P)^{1/n_j}} = 0, \qquad i = 1\cdots3. \quad (1)$$

The first equation is the total material balance. The second equation is the material balance for each adsorbed species. The third equation forces the mole fractions in the gas phase to sum to unity. The fourth equation is the loading ratio correlation that gives the equilibrium loading of each adsorbed component.

The project requires dynamic simulation of the system from a cold start. Initial conditions for the six differential variables are

$$y_i(0, z) = 1.0\times10^{-6}, \qquad i = 1\cdots3$$
$$q_i(0, z) = 0, \qquad i = 1\cdots3, \quad (2)$$

while boundary conditions at startup are given by the feed compositions $y_{f,i=1\ldots3}$ and velocity $u_f = 0$

$$y_i(t, 0) = y_{f,i}, \qquad i = 1\cdots3$$
$$u(t, 0) = u_f. \quad (3)$$

Partial derivatives with respect to z are discretized using a first-order upwind finite difference scheme, and an implicit

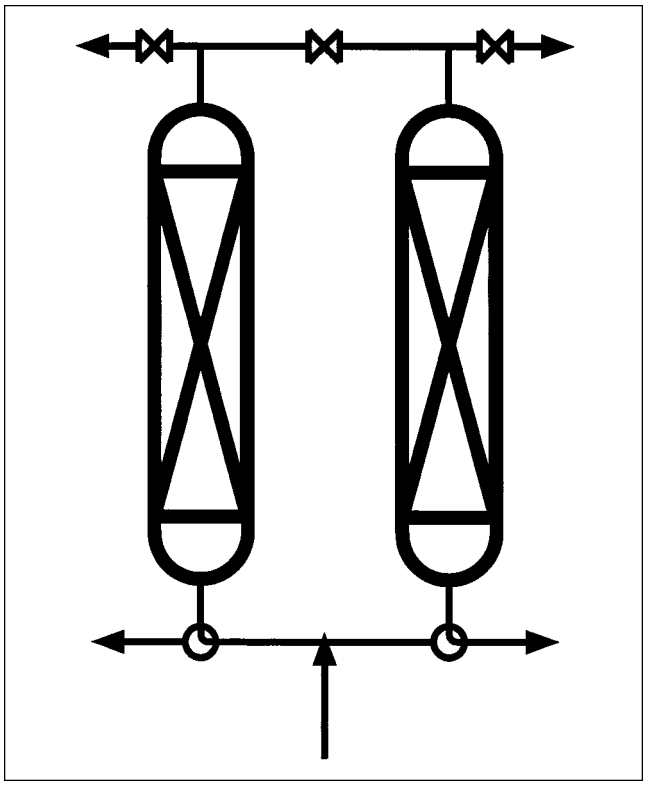

Figure 1. PSA flowsheet.

BDF integration method is used to advance the solution forward in t. The disappointing results appear in Figure 2. The simulation fails after a simulated time of 30 s, when the reini-

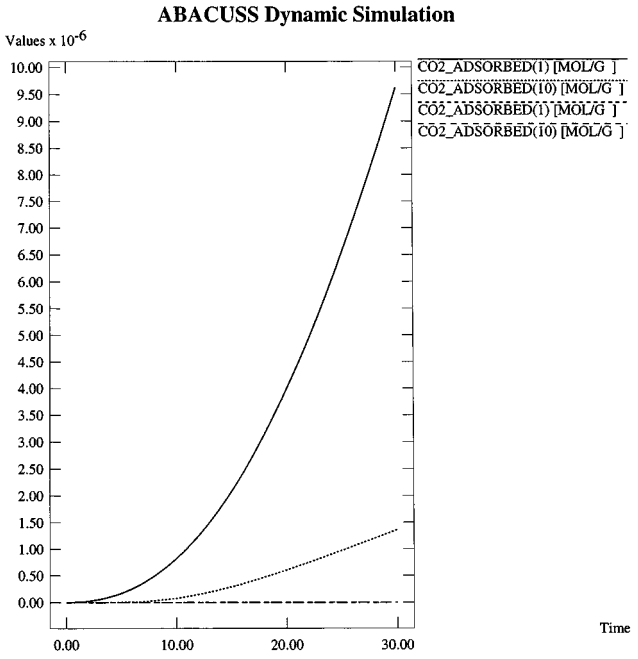

Figure 2. Simulation results for the PSA process.

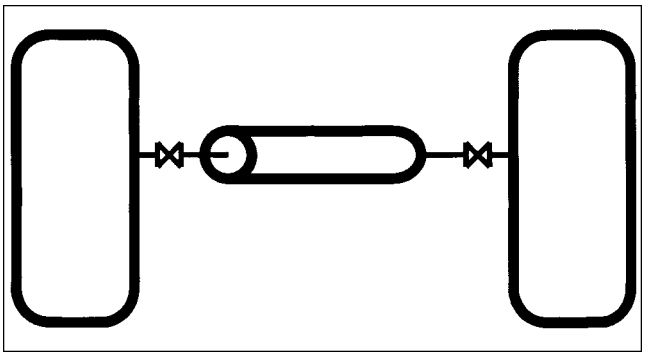

Figure 3. Vessel depressurization flowsheet.

tialization calculation required after the first valve position change does not converge.

What is wrong? The task facing the engineer is to figure out what is wrong, and do it as quickly as possible.

Compressible flow

The second example involves simulation of a vessel depressurization. The simplified flow sheet for this process consists of two pressure vessels, two valves, and the process piping, and appears in Figure 3. The gas is compressible, and if friction losses, gravity, and radial variations are ignored, and the gas is assumed ideal, flow is described by the *Euler equations* (Jeffrey, 1976; Roe, 1986)

$$\begin{aligned} \rho_t + (\rho u)_x &= 0 \\ (\rho u)_t + \left(p + \frac{1}{2}\rho u^2\right)_x &= 0 \\ (\rho h)_t + (up - \rho u h)_x &= 0 \\ p &= (\gamma - 1)\rho i \\ h &= i + \frac{1}{2}u^2. \end{aligned} \tag{4}$$

Here ρ is the fluid density, u is the flow velocity, p is pressure, h is the specific total energy, and i is the specific internal energy. The first three model equations are conservation of mass, momentum, and energy, respectively. The fourth is the ideal gas law, with a constant fluid heat capacity ratio of γ. The final equation relates total, internal, and kinetic energy.

The pipe segment under consideration is 10 m in length, so $0 \le x \le 10$, and also let $t \ge 0$. The initial and boundary conditions are

$$\begin{aligned} \rho(0, x) &= 79.6 \text{ kg/m}^3 \\ u(0, x) &= 0.0 \text{ m/s} \\ p(0, x) &= 2.76 \text{ MPa} \\ p(t, 0) &= f_{\text{valve1}}(t) \\ p(t, 10) &= f_{\text{valve2}}(t). \end{aligned} \tag{5}$$

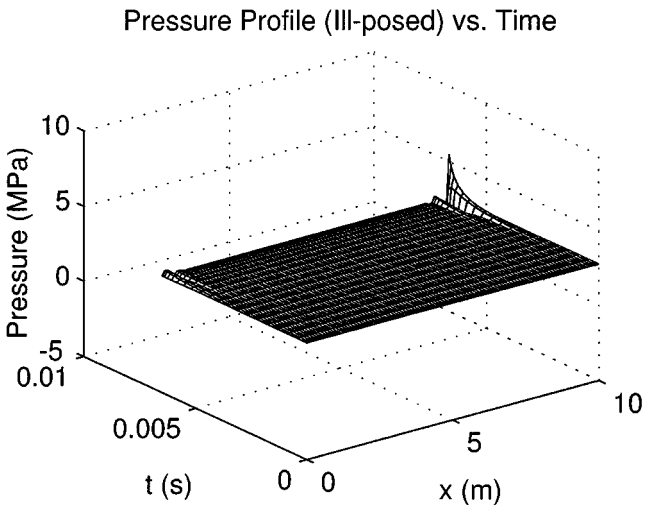

Figure 4. Pipe pressure profile.

The first scenario of interest is a case where the pressure in the pipe is initially slightly higher than the pressure in both vessels. The pressure in one vessel is significantly higher than the other.

Again, the problem will be solved using a first-order upwind finite difference scheme (Strikwerda, 1989). Initially, flow out of both ends of the pipe is expected, followed by establishment of a steady pressure gradient and flow from the high-pressure vessel to the low-pressure vessel.

Simulation results, specifically the pressure profile along the pipe, appear in Figure 4. Clearly, something is wrong. The calculated pressure profile blows up at the right endpoint. One would expect a rarefaction to enter the pipe from both ends, followed by establishment of a steady pressure gradient between the two ends. Instead, the calculated solution blows up after less than 0.3 simulated seconds.

Possible problems include improper boundary conditions, an improper discretization scheme, a time step or mesh spacing that is too large, and simple code bugs. The engineer again faces the task of uncovering the root of the problem and correcting it.

Electric power transmission

Next, consider simulations of 420-kV power transmission lines in an electric power distribution grid. Current flow I and voltage with respect to ground u over a transmission line are described by the following simple system of two equations, which are known as the *telegrapher's equations*

$$\begin{bmatrix} 0 & L \\ C & 0 \end{bmatrix} \begin{bmatrix} u \\ I \end{bmatrix}_t + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ I \end{bmatrix}_x + \begin{bmatrix} 0 & R \\ G & 0 \end{bmatrix} \begin{bmatrix} u \\ I \end{bmatrix} = 0. \tag{6}$$

Here L, C, R, and G are the inductance, capacitance, resistance, and conductance of the line per unit length.

The scenario of interest is a 1% increase in current demand occurring over 0.5 s, to be delivered over a 10-km line. For this particular line, $L = 0.0046\ \Omega \cdot \text{s/km}$, $C = 6.5$ nF/km, $G = 33.3\ 1/\Omega \cdot \text{km}$, and $R = 0.030\ \Omega/\text{km}$.

Measured values at the substation are 380 kV at 50 Hz, with a typical current demand of 3,160 A. These values will be used for boundary conditions. The current demand will be given as a sinusoid increase from 3,160 to 3,192 over 0.5 s

$$u(0,t) = 190{,}000 * \sin(50\pi t)$$
$$I(0,t) = \left(1.0 + 0.005\{1.0 + \sin[\pi(2t + 1.5)]\}\right)3{,}160. \quad (7)$$

The domain is a 10-Km line, and the simulation will cover the surge in demand, so $0 \le x \le 10$ and $0 \le t \le 0.5$.

Here, the engineer wants to build the complexity of the simulation slowly, and therefore begins with a simplified form (Massobrio and Antognetti, 1993) of the telegrapher's equations that neglects the line inductance, resistance, and conductance

$$\begin{bmatrix} 0 & 0 \\ C & 0 \end{bmatrix}\begin{bmatrix} u \\ I \end{bmatrix}_t + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} u \\ I \end{bmatrix}_x = 0. \quad (8)$$

While these assumptions behind this simplification are *not* valid for this system, experience with chemical process simulations has taught the engineer to start with simplified models, and move to simulations based on more rigorous models once the simulation based on a simplified model is working.

The partial derivative terms in x are discretized using centered finite differences, and the line voltage is initialized to 190 kV. Simulation results for the simplified model appear in Figure 5. The results look good, so the engineer proceeds to the full model.

The partial derivative of current with respect to time, while absent from the simplified model, is present in the full model. The engineer initializes the current in the line to its nominal demand of 3160 A. Results for the full current delivery model appear in Figure 6. The simulation blows up immediately. Once again, the task is to determine what is causing the simulation to fail.

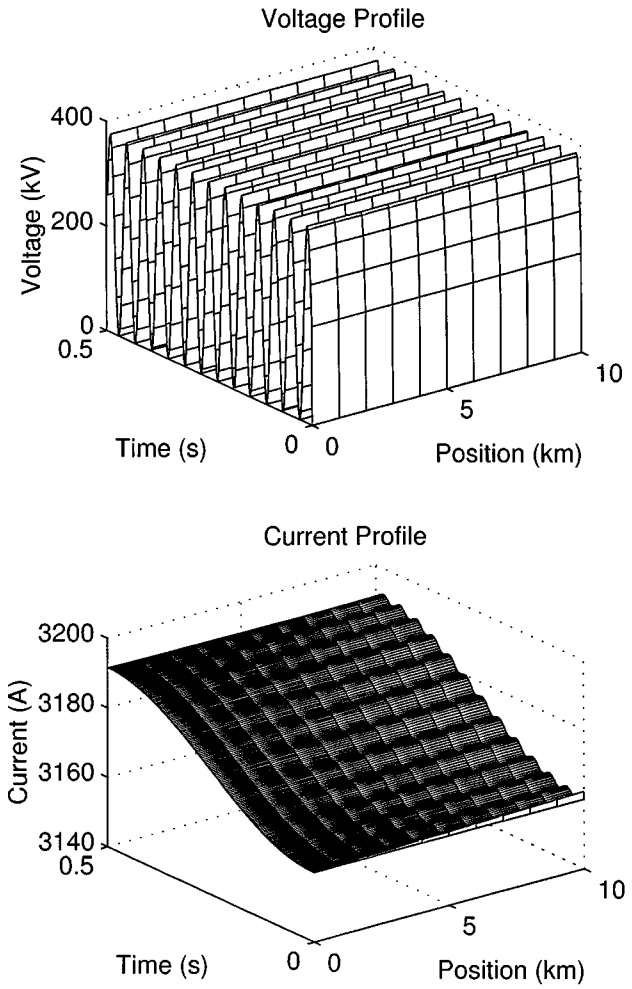

Figure 5. Simulation results for simplified electrical-current model.

Review

We consider a first-order PDAE system with u the dependent variables, and two independent variables x and t.

The *differentiation index with respect to t*, ν_t, or simply the *index with respect to t*, is defined (Martinson and Barton, 2000) as the minimum number of times some or all of the equations must be differentiated in order to determine u_t as a continuous function of u, x, and t. The index with respect to x is defined in an analogous manner.

This particular definition of the index [as opposed to perturbation, modal, or algebraic indices (Campbell and Marszalek, 1997)] is a natural generalization of the differentiation index of a DAE (Brenan et al., 1989). As such, the index with respect to t provides insight into the expected index of any DAE that is generated by a method of lines semidiscretization. It also allows algorithms based on the index of a DAE to be applied with only minor modification to PDEs (Martinson and Barton, 2001). The index is important because high index (2 or greater) DAEs cannot be solved accurately by standard numeric integration codes (Brenan et al., 1989); the calculated solution may fail or, worse, drift away from the true solution with no indication that specified error tolerances had been violated. Index analysis may also be used to assist with the task of proper Cauchy data formulation (Martinson and Barton, 2000). No classic analysis exists for this problem, because it is trivial in the case of a strictly hyperbolic or parabolic system; however, it can become significantly more complex for the general PDAE models considered in this article.

A PDE system is said to be *well-posed* if it has a unique solution that depends continuously on its data (Lieberstein, 1972). The model equations typically admit a family of solutions, and proper initial and boundary conditions must be provided in order to specify a unique member of that family. If no solution exists, or it is not possible to determine a unique solution, one cannot expect a standard numerical code to generate meaningful results. If the solution does not depend continuously on its data, tiny errors in initial or boundary conditions may govern the computed solution. Models that do not depend continuously on their data are therefore not suitable for dynamic simulation.

Consider a first-order linear PDE system over two independent variables t and x of the form

$$Au_t + Bu_x + Cu = f(t,x) \quad (9)$$

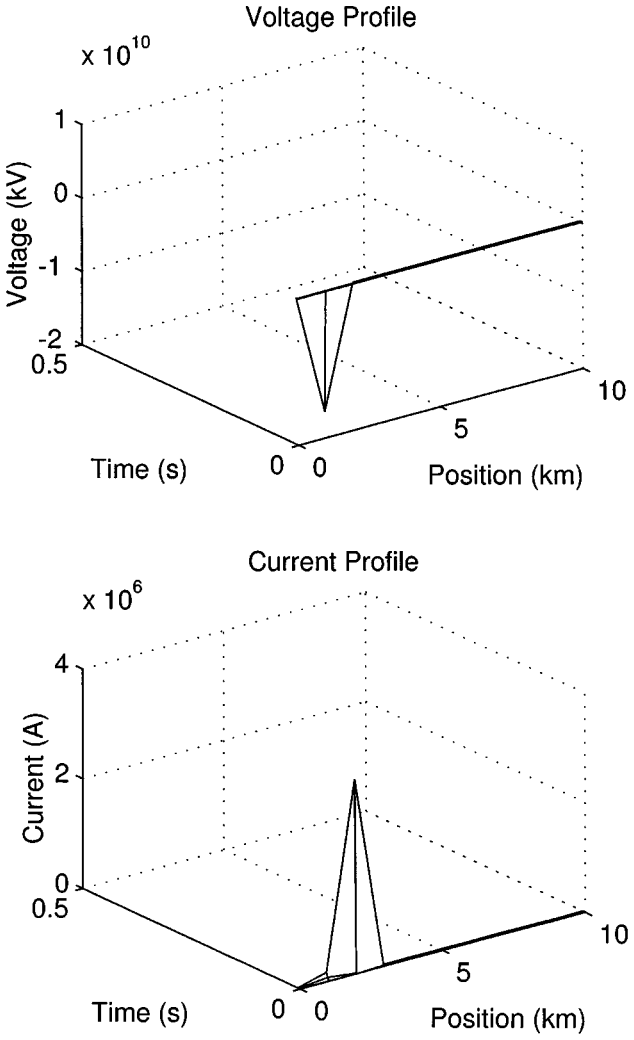

Figure 6. Simulation results for full electrical-current model.

on the semi-infinite domain $a \le x \le b, t \ge 0$, with A, B, $C \in \mathbb{R}^{n\times n}$, $u \in \mathbb{R}^n$, and $f:\mathbb{R}^2 \to \mathbb{R}^n$.

If A is invertible, multiplication on the left by A^{-1} produces

$$u_t + \bar{B}u_x + \bar{C}u = \bar{f}(t, x). \tag{10}$$

The solution to such a system depends continuously on its data [and is hyperbolic (Courant and Hilbert, 1962)] if $\bar{B}$ is diagonalizable with strictly real eigenvalues (Kreiss and Lorenz, 1989).

Let L be a matrix that consists of the left eigenvectors of $\bar{B}$, and let Λ be a diagonal matrix that contains the corresponding eigenvalues, so that $L\bar{B}L^{-1} = \Lambda$. If $\bar{C} = 0$, multiplication of the system on the left by L and introduction of new variables $v = Lu$ produces a system of decoupled (for $\bar{C} \neq 0$, the equations are coupled, but the implications for boundary condition placement are unchanged) equations of the form

$$v_{i_t} + \lambda_i v_{i_x} = \hat{f}_i(t, x), \tag{11}$$

which is equivalent to an ODE along $dx/dt = \lambda_i$. As such, an initial condition on v_i determines a unique solution. This is the *characteristic form* of the hyperbolic system.

Let there be n_1 eigenvalues of $\bar{B}$ greater than zero, and n_2 less than zero. The solution to a hyperbolic system exists and is unique if n independent initial conditions are provided at $t = 0$, n_1 independent boundary conditions are provided on $x = a$, and n_2 independent boundary conditions are specified on $x = b$.

This analysis can provide information on all three components of well-posedness, and furthermore because it relies only on calculation of eigenvalues and eigenvectors of a matrix, it is amenable to implementation in a chemical-process simulator. However, the analysis applies only to systems with A invertible and $\bar{B}$ diagonalizable. This precludes analysis of more general, nonhyperbolic systems. In particular, algebraic equations or equations that only involve partial derivatives with respect to x make a system nonhyperbolic.

More recent work (Martinson and Barton, 2001) has extended this analysis to a much broader class of nonhyperbolic systems. First, it has been proven that a system of the form given earlier (Eq. 9) depends continuously on its data if all generalized eigenvalues of the matrix pair (A, B) are strictly real and have geometric multiplicity 1. The proof also allows local consideration of quasi-linear systems by freezing the coefficient matrices. If some generalized eigenvalues of the matrix pair are strictly real, but infinite and of geometric multiplicity 2, the solution has also been shown to depend continuously on its data.

This work also developed a canonical form that may be thought of as a generalization of the characteristic form of a hyperbolic system. If the coefficient matrix pair forms a regular pencil, then the system is equivalent to one of the form

$$\begin{bmatrix} J & & \\ & N_1 & \\ & & I \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}_t + \begin{bmatrix} I & & \\ & I & \\ & & N_2 \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}_x + C^* u = \begin{bmatrix} f_1(t, x) \\ f_2(t, x) \\ f_3(t, x) \end{bmatrix}, \tag{12}$$

where J is a lower Jordan matrix, and N_1 and N_2 are lower Jordan matrices of nilpotencies ν_1 and ν_2, respectively.

Each subblock in one of the three block rows has been shown to be equivalent to an ODE system along a particular direction in the (t, x) plane. The direction is given by the generalized eigenvalue (τ_i, ρ_i) that corresponds to the subblock. For the first block row, called the *hyperbolic part*, the number of boundary conditions that are required on $x = a$ and $x = b$ are again equal to the number of positive and negative generalized eigenvalues, respectively, that are associated with the hyperbolic part. The number of initial conditions is equal to the dimension of the hyperbolic part. If any

associated generalized eigenvalues have nonzero degeneracy, which is defined as one less than the geometric multiplicity (the geometric multiplicity of a generalized eigenvalue is defined as the dimension of the associated Jordan block; also, the overall degeneracy of the system is defined as the maximum degeneracy of any generalized eigenvalue), the system does not depend continuously on its data.

No boundary conditions are required for the third block row, which is called the *differential part*. The number of initial conditions is equal to the dimension of the differential part. If any associated generalized eigenvalues have nonzero degeneracy, the system again fails to depend continuously on its data.

The total number of boundary conditions that are required for the second block row, which is the *parabolic part*, is equal to its dimension. The boundary condition associated with a generalized eigenvalue of degeneracy zero in the parabolic part can be enforced at either $x = a$ or $x = b$. No initial condition is required for a subblock associated with a generalized eigenvalue of degeneracy zero. For a generalized eigenvalue of degeneracy one in the parabolic part, if the index of the entire system with respect to t is less than two, the solution can depend continuously on its data and the problem can be well posed only if the two associated boundary conditions for the subblock are not enforced at the same point, and an initial condition is also specified.

A system for which the coefficient matrix pair (A, B) does not form a regular pencil (such as in the presence of algebraic equations), but that is equivalent to an algebraic system coupled to a PDE with a regular coefficient matrix pencil

$$\begin{bmatrix} A_{11} & \mathbf{0} \\ A_{21} & \mathbf{0} \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}_t + \begin{bmatrix} B_{11} & \mathbf{0} \\ B_{21} & \mathbf{0} \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}_x + \begin{bmatrix} C_{11} & \mathbf{0} \\ C_{21} & C_{22} \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}, \quad (13)$$

where $\dim(C_{22}) = n - r$, $r = \max_{\lambda \in \mathbb{R}}(\text{rank}(A + \lambda B))$, with (A_{11}, B_{11}) regular and C_{22} invertible, can be handled in the same manner as one with a regular pencil. Because the first block row involves only u_1, it can be considered independently of the second block row. Again assuming a dynamic simulation based on a time-evolution method, the first block row provides the same information regarding dependence on and location of data given by generalized characteristic analysis in the regular coefficient matrix pencil case. Once the first block row is solved for u_1, no additional data are required to uniquely determine u_2. The second block row is therefore called the *algebraic part* of the system.

A differential system that is equivalent to an algebraic system can also be coupled to a regular PDE and handled in the same way. Let N_1 and N_2 be two conforming nonzero nilpotent matrices, both either strictly upper triangular or strictly lower triangular

$$\begin{bmatrix} A_{11} & \mathbf{0} \\ A_{21} & N_1 \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}_t + \begin{bmatrix} B_{11} & \mathbf{0} \\ B_{21} & N_2 \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}_x + \begin{bmatrix} C_{11} & \mathbf{0} \\ C_{21} & I \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}. \quad (14)$$

An important special case is systems that contain one or more strictly algebraic equations. An algebraic equation constrains the dependent variables on *every* surface in the independent variable space, so a system that contains an algebraic equation can be viewed as one for which every surface is characteristic. This corresponds to $A_{21} = B_{21} = \mathbf{0}$ in the form just considered (Eq. 13).

If an algebraic equation is differentiated once with respect to time, it becomes an ordinary differential equation. This is an interior partial differential equation on surfaces of the form $x = constant$, such as the domain boundaries. In other words, differentiation with respect to t transforms an algebraic equation, which constrains the solution on all surfaces, to one that constrains the solution on domain boundaries of the form $x = constant$. If one is interested in the equations that partially determine the solution u on a domain boundary, the original and differentiated algebraic equations are equivalent. Because it may be difficult to identify what variables belong to the algebraic part, differentiating all algebraic equations once and analyzing the resulting system is a viable alternative if it produces a regular coefficient matrix pencil. It has been proven (Martinson and Barton, 2001) that if the differentiation index with respect to either t or x is zero, or equal to one subject to a mild rank condition, that differentiating the algebraic equations indeed regularizes the coefficient matrix pencil.

For semilinear and quasi-linear systems, boundary condition requirements, dependence on data, and the index are local properties that may change with different values of x, t, or u. The boundary-condition requirements can be determined by evaluating (or *freezing*) the coefficient matrices at some nominal value (u_0, x_0, t_0) of interest, and examining the canonical form of the resulting linear system. If the system has a parabolic part, additional assumptions are required, because the boundary-condition analysis is inherently nonlocal. It has been proven (Martinson and Barton, 2001) that the local dependence on data also can be determined by the generalized eigenvalues of the frozen coefficient system.

Implementation

The goal of this work is to automate the analyses of the previous sections as much as possible. In particular, determination of the index, degeneracy, characteristic directions, and variables associated with the subsystems of the canonical form will allow a simulator to verify initial and boundary conditions, identify systems of high index with respect to the evolution variable t, and detect some ill-posed systems.

Difficulties with direct calculation of the canonical form of a DAE (Bujakiewicz, 1994) and a desire to develop methods that can be used for nonlinear problems have led to the development of structural index algorithms (Kröner et al., 1992; Pantelides, 1988). These algorithms work with the occurrence information to determine the minimum number of differentiations required to produce a low-index (zero or one) system. It is well known that DAEs of high index due to numerical singularities may escape detection by structural algorithms. Recent work (Reisszig et al., 2000) has highlighted the fact that structural algorithms may also *overestimate* the number of differentiations required to produce a low-index system. However, the low computational cost of these algorithms and

their applicability to nonlinear and large, sparse systems allows them to be used with considerable success in practical applications (if new algorithms emerge that provably perform this analysis properly, then they can be applied directly and the answer will be unambiguous).

A second algorithm, called the method of dummy derivatives (Mattsson and Söderlind, 1993), has been used successfully in conjunction with Pantelides' algorithm to generate automatically a low index system that is mathematically equivalent (at least locally) to the original system and explicitly preserves all constraints. From this dummy reformulation of the original system, one can obtain the dynamic degrees of freedom, which is equal to the number of differential variables. Note that this number may be correct even in the case where the number of differentiations has been overstated by the structural algorithm.

Both algorithms can be applied in an extremely straightforward manner to PDEs. The index with respect to t, for example, is determined by considering all interior partial differential operators together with algebraic operators. The incidence matrix for t-algebraic occurrences of the dependent variables is formed by simply merging the incidence matrices for u_x and u. Once this has been done, the two algorithms will (in the absence of numerical singularities) produce an equivalent system of index 0 or 1 with respect to t that reflects the true number of t-differential variables. The number of initial conditions required in order to determine a unique solution is equal to the number of t-differential variables in the t-dummy reformulation. An analogous x-dummy reformation is evident.

The most basic necessary condition for well-posedness of a linear system is the regularity condition of Campbell and Marszalek (1997). A system that does not possess an output set assignment will not satisfy this necessary condition. Pantelides' algorithm also identifies systems that fail to meet this necessary condition as a result of structural singularity, as a preprocessing step that guarantees the algorithm has finite termination. If the system is linear and has the general form considered earlier (Eq. 9) with $C = \mathbf{0}$, and if any generalized eigenvalues of the coefficient matrix pair are given by 0/0, the system also fails the regularity condition.

Routines that calculate the generalized eigenvalues and their degeneracies for regular coefficient matrix pairs are readily available (Golub and van Loan, 1989; Demmel and Kågström, 1993). If any generalized eigenvalues are complex, the system is ill-posed. Otherwise, if the degeneracy of the system is zero, the solution depends continuously on its data. If the degeneracy of the system is nonzero but the forcing is simple, the system is weakly well-posed. For linear forcing and nonzero degeneracy, it is not in general possible at present to distinguish between weakly well-posed and strongly ill-posed systems.

Index analysis can be used to identify the total number of boundary conditions required to determine a unique solution. Just as index analysis with respect to t gives the number of dynamic degrees of freedom on surfaces of the form $t =$ const., index analysis with respect to x gives the number of dynamic degrees of freedom on surfaces of the form $x =$ const. In a dynamic simulation with t as the evolution variable, all such degrees of freedom on surfaces of the form $t =$ const. must be specified as initial conditions, while dynamic degrees of freedom on surfaces of the form $x =$ const. can be specified on either $x = a$ or $x = b$.

The distribution of these boundary conditions between the boundaries $x = a$ and $x = b$ can be ascertained from the generalized eigenvalues. Each block in the hyperbolic subsystem was shown to be equivalent to an ODE along a particular direction in the (x, t) plane, given by $dx/dt = \tau_i/\rho_i$. Because a dynamic simulation in t is assumed, data provided at t_2 may not be used to specify a unique solution at $t_1 < t_2$, so initial conditions for these ODEs must be provided as boundary conditions on $x = a$ for ODEs along $dx/dt > 0$, and as boundary conditions on $x = b$ for ODEs along $dx/dt < 0$.

Blocks in the parabolic subsystem are equivalent to ODEs in x, or along the direction $dt/dx = 0$. An initial condition for such an ODE can in general be given at either domain boundary in x. In particular, a parabolic block of dimension 1 requires a boundary condition at either $x = a$ or $x = b$.

If the only blocks with nonzero degeneracy are part of the parabolic subsystem and of dimension 2, and the index of the system with respect to t is 1, the solution to the parabolic blocks will not depend continuously on their data if those data are enforced at a single side of the domain in x. Such a problem may still be well-posed as an evolution problem in t if one boundary condition is enforced at each side of the domain in x for every parabolic block of degeneracy 1.

By the same approach but with the roles of t and x reversed, if the only blocks with nonzero degeneracy are part of the differential subsystem and the index of the system with respect to x is 1, the solution will not depend continuously on its data if those data are enforced at a single surface. As an evolution problem in t, the problem is therefore ill-posed.

It is possible to move beyond simply counting the number of required boundary conditions and to identify the information that those boundary conditions must provide. The matrices $\mathbf{P}$ and $\mathbf{Q}$ that transform the system to its generalized characteristic form can be computed stably only when the degeneracy of the system is zero; when the degeneracy is nonzero, stable similarity transforms exist that take both A and B to upper triangular matrices (Demmel and Kågström, 1993). While not the characteristic form of the system, this *generalized upper triangular form* can be used in the same manner as the characteristic form for a more detailed boundary-condition analysis.

Consider now a linear system in generalized upper triangular form (the generalized characteristic form can be used instead if available)

$$PAQ\boldsymbol{v}_t + PBQ\boldsymbol{v}_x = -PCQ\boldsymbol{v} + Pf(t, x). \qquad (15)$$

Let $\rho_i = (PAQ)_{ii}$ and $\tau_i = (PBQ)_{ii}$. Because the coefficient matrix pencil is assumed regular, it is not possible for $\rho_i = \tau_i = 0$, and thus an output set assignment of v_i to equation i is implied. Given this output set assignment, each dependent variable is given as the solution to a (possibly degenerate) one-way wave.

A dynamic simulation implies advancing a solution forward in t. The values of the dependent variables v_i, for which the associated characteristic direction ρ_i/τ_i is nonpositive, are determined at $x = a$ by the outward-directed characteristics. Similarly, values associated with characteristics that have

speeds greater than or equal to 0 are determined at $x = b$. Once the value of a dependent variable associated with an infinite-speed characteristic is specified at one domain boundary, it is determined at the other as well. Initial conditions on $t = 0$ determine the variables associated with characteristics of speed 0 on the boundaries at all later times.

Let $\boldsymbol{v}_p$, $\boldsymbol{v}_r$, $\boldsymbol{v}_l$, and $\boldsymbol{v}_d$ be the variables associated with infinite-, positive-, negative-, and zero-speed characteristics, respectively. The values of the dependent variables that are determined by characteristics at each boundary may be written as the solution to a system of the following form

$$\begin{bmatrix} I_p & 0 & 0 & 0 & I_p & 0 & 0 & 0 \\ 0 & I_l & 0 & 0 & & & & \\ 0 & 0 & 0 & I_d & & & & \\ & & & & 0 & 0 & I_r & 0 \\ & & & & 0 & 0 & 0 & I_d \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_p(a,t) \\ \boldsymbol{v}_l(a,t) \\ \boldsymbol{v}_r(a,t) \\ \boldsymbol{v}_d(a,t) \\ \boldsymbol{v}_p(b,t) \\ \boldsymbol{v}_l(b,t) \\ \boldsymbol{v}_r(b,t) \\ \boldsymbol{v}_d(b,t) \end{bmatrix} = g(t,x). \tag{16}$$

This system represents the parts of the solution that are fully determined at $x = a$ and $x = b$ by characteristics; it is not in general possible to give the righthand side analytically. It can, however, be used to evaluate the information contained in the boundary conditions. Each dependent variable v_i in the generalized upper triangular form is a linear combination of the original variables $\boldsymbol{u}$. Transforming back to these original variables, the system becomes

$$\begin{bmatrix} C_p & C_p \\ C_l & \\ C_d & \\ & C_r \\ & C_d \end{bmatrix} \begin{bmatrix} u(a,t) \\ u(b,t) \end{bmatrix} = f(t,x). \tag{17}$$

Suppose the boundary conditions to be enforced at $x = a$ are given by $G_a u = h_1(t)$, and at $x = b$ by $G_b u = h_2(t)$. The boundary conditions determine a unique solution if

$$\det \begin{bmatrix} C_p & C_p \\ C_l & \\ C_d & \\ & C_r \\ & C_d \\ G_a & \\ & G_b \end{bmatrix} \neq 0. \tag{18}$$

If the boundary conditions are Dirichlet conditions, then G_a and G_b are real matrices, and this determinant can be evaluated numerically. For Neumann and Robin conditions, the coefficient matrix for the boundary conditions is operator valued, which makes evaluation of the determinant a symbolic calculation.

Finally, consider systems with a singular coefficient matrix pencil. The cost of verifying the conditions under which differentiation of algebraic equations with respect to t is guaranteed to produce a regular coefficient matrix pencil is greater than the cost of simply performing the necessary differentiations. After differentiation, the generalized upper triangular form will reveal whether or not the differentiations produced a regular pencil.

This analysis and implementation can be summarized as follows:

1. Use Pantelides' algorithm to obtain an estimate of the index of the system with respect to both t and x. In the absence of numerical singularities of the relevant matrices, the algorithm will return the true indices.

2. Use the information returned by Pantelides' algorithm with the method of dummy derivatives to produce two reformulated systems that are by construction low index with respect to t and with respect to x (index 0 or 1).

3. Differentiate any algebraic equations once with respect to t. Calculate the generalized eigenvalues of this new coefficient matrix pair. Calculate the matrices P and Q that transform the coefficient matrix pair to either its canonical form or its generalized upper triangular form.

The results of the preceding three calculations provides a great deal of information regarding the index and well-posedness of a particular unit model. In the absence of numerical singularities, Pantelides' algorithm returns the index of the system with respect to t directly. If $\nu_i \geq 2$, any reasonable method of lines semidiscretization in t will produce a high-index DAE.

Well-posedness information based on the results of these three calculations can be summarized as follows:

1. If Pantelides' algorithm terminates because it is unable to generate a transversal with respect to all occurrences of the dependent variables, a unique solution does not exist and the problem is ill-posed.

2. If any generalized eigenvalues of the coefficient matrix pair are complex, the solution does not depend continuously on its data, and the problem is ill-posed.

3. If $C = \mathbf{0}$ and any generalized eigenvalues of the coefficient matrix pair are given by 0/0, the system fails the regularity condition, so the solution is not unique and the problem is ill-posed.

4. If the number of initial conditions is less than the number of t-differential variables in the t-dummy reformulation, the solution is not unique, and the problem is ill-posed. If the number of initial conditions is greater than the number of t-differential variables in the t-dummy reformulation, the problem is overdetermined. It may be redundant or inconsistent; in the latter case, no solution exists and the problem is ill-posed.

5. If the total number of boundary conditions is less than the number of x-differential variables in the x-dummy reformulation, the solution is not unique, and the problem is ill-posed. If the number of boundary conditions is greater than the number of x-differential variables in the x-dummy reformulation, the problem is overdetermined. It may be redundant or inconsistent; in the latter case, no solution exists and the problem is ill-posed.

6. If the number of boundary conditions at $x = a$ is less than the number of positive generalized eigenvalues, or the number of boundary conditions at $x = b$ is less than the number of negative generalized eigenvalues, the solution is not unique, and the problem is ill-posed.

7. If any eigenvalue given by τ/ρ, $\rho \neq 0$ has degeneracy 1, and $\nu_x < 2$, the solution does not depend continuously on its data, and the problem is ill-posed.

8. If any generalized eigenvalue given by τ/ρ, $\rho = 0$ has degeneracy 2, and $\nu_t < 2$, the solution may depend continuously on its data only if one boundary condition for the corresponding block is enforced at each domain endpoint, and an initial condition is also specified.

Again note that this analysis applies rigorously only to linear systems. Extensions based on local linearization can be made to semilinear and quasi-linear systems, but very few general statements can be made about truly nonlinear distributed unit models.

Demonstrations

Pressure-swing adsorption

Could the analyses outlined in the previous section enable a simulator to provide some insight into the cause of the difficulties with the pressure-swing adsorption simulations? The first step is estimation of the index. Pantelides' algorithm differentiates the isotherm once before terminating, indicating that the index of the system with respect to t is 2, and thus immediately pointing to the underlying cause of the simulation failure. The original system had a high index with respect to t, which was preserved by the method-of-lines semidiscretization, and thus produced a high-index DAE.

The simulator could provide an equivalent dummy reformulation of the original PDE that had index 1 with respect to t. There are two possible dummy reformulations; one is

$$\frac{\rho_B RT}{P}\sum_{i=1}^{3} q_i' + \frac{\epsilon}{P}P_t + u_z = 0$$

$$\epsilon y_{i_t} + \frac{\rho_B RT}{P}q_i' + \frac{\epsilon y_i}{P}P_t + (uy_i)_z = 0, \qquad i = 1\cdots 3$$

$$\sum_{i=1}^{4} y_i = 1$$

$$q_i - \frac{q_i^{\text{sat}} B_i (y_i P)^{1/n_i}}{1 + \sum_{j=1}^{3} B_j (y_j P)^{1/n_j}} = 0, \qquad i = 1\cdots 3 \quad (19)$$

$$\left(1 + \sum_{j=1}^{3} B_j (y_j P)^{1/n_j}\right) q_i' + \left(q_i \sum_{j=1}^{3} B_j P^{1/n_j} - q_i^{\text{sat}} B_i P^{1/n_i}\right) \times \left(\frac{1}{n_i}\right) y_i^{[1/(n_i-1)]} y_{i_t} = 0, \qquad i = 1\ldots 3.$$

By item 3 in the analysis of the equations, only three initial conditions should be enforced.

Discretizing this system using the same upwind finite difference scheme and employing the same BDF integrator in time produces a low-index DAE. Once the (redundant) initial conditions on $q_{i=1\ldots 3}$ are eliminated, the solution proceeds normally. Results for the first few operating cycles appear in Figure 7.

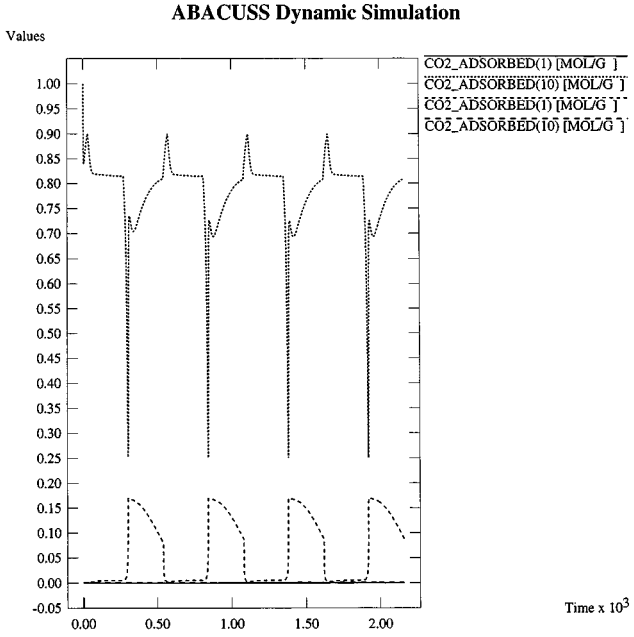

Figure 7. Simulation results for reformulated problem.

In this case automated model analysis is able to immediately identify the root cause of the simulation failure. Furthermore, a simulator would be able to correct the underlying problem automatically, with no intervention on the engineer's part.

Compressible flow

What about the difficulties with the compressible flow simulation? Can a process simulator use these tools to help get this simulation working?

In quasi-linear form, the model equations are

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ u & \rho & 0 & 0 & 0 \\ h & 0 & 0 & \rho & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \rho \\ u \\ p \\ h \\ i \end{bmatrix}_t + \begin{bmatrix} u & \rho & 0 & 0 & 0 \\ u^2 & 2\rho u & 1 & 0 & 0 \\ -uh & p-\rho h & u & -\rho u & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \rho \\ u \\ p \\ h \\ i \end{bmatrix}_x = \begin{bmatrix} 0 \\ 0 \\ 0 \\ p-(\gamma-1)\rho i \\ i - h + \frac{1}{2}u^2 \end{bmatrix}. \quad (20)$$

Pantelides' algorithm, applied to determine the index with respect to t, locates no structurally singular subsets of equations. The index with respect to t is in fact 1. No dummy reformulation is necessary.

Differentiating the algebraic equations with respect to t produces

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ u & \rho & 0 & 0 & 0 \\ h & 0 & 0 & \rho & 0 \\ (1-\gamma)i & 0 & 1 & 0 & (1-\gamma)\rho \\ 0 & u & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} \rho \\ u \\ p \\ h \\ i \end{bmatrix}_t + \begin{bmatrix} u & \rho & 0 & 0 & 0 \\ u^2 & 2\rho u & 1 & 0 & 0 \\ -uh & p-\rho h & u & -\rho u & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \rho \\ u \\ p \\ h \\ i \end{bmatrix}_x = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{21}$$

The system is quasi-linear, so the coefficient matrices must be frozen at a point of interest. Consider the domain boundary at $x = 10$, and let conditions at $x = 10$ be $\rho = 79.6$ kg/m^3, $u = 0.00$ m/s, $p = 2.76$ MPa, $h = 86.6$ kJ, and $i = 86.6$ kJ. The frozen-coefficient matrices are submitted to an eigensolver, such as the LAPACK routine dgegv. The result is three characteristic directions parallel to the t coordinate axis and two complex characteristic directions.

The system is thus ill-posed in a neighborhood of these nominal values, and cannot be solved by a simulator as part of a dynamic simulation. A process simulator could then advise the engineer that the equations, as entered, are ill-posed, at least in the vicinity of the initial conditions. On review of the input, the sign error made in the energy balance (Eq. 4) should be corrected

$$(\rho h)_t + (\rho uh + up)_x = 0. \tag{22}$$

The analysis can then be repeated for the corrected system. Now, all generalized eigenvalues are strictly real. Three are zero: the other two are ± 220.3. The corrected problem is well-posed.

Simulation results for the corrected problem appear in Figure 8. As expected, a rarefaction enters the pipe from both ends. This time, the simulation failure was the result of a simple sign error on the engineer's part. This sign error produced a strongly ill-posed system, which can be detected by a process simulator through the use of the analyses developed in this article.

Electric power transmission

Could the automatable analyses presented in this article help uncover the cause of the electric power-line simulation failure? The index of the system with respect to both t and x is zero; Pantelides' algorithm would correctly return no differentiations. Therefore, no reformulation is necessary. The coefficient matrices are linear and have two generalized eigenvalues, $\pm 182{,}879$, each of geometric multiplicity 1. The

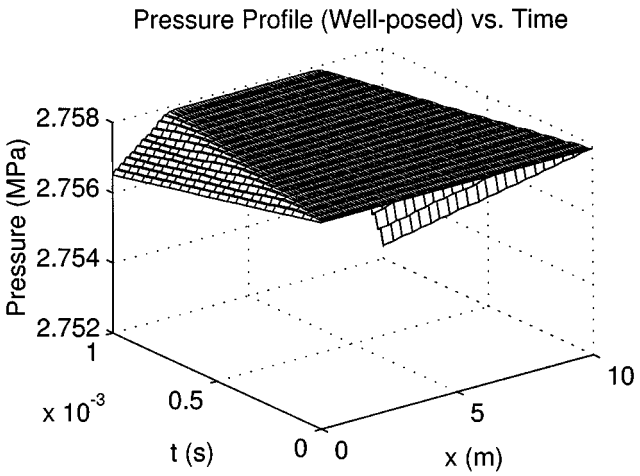

Figure 8. Corrected pipe pressure profile.

problem, as the engineer has defined it, is thus ill-posed, because the two boundary conditions enforced at the substation do not determine a unique solution. In this case, it means that the engineer must obtain data from another substation at the other end of the line, in order to provide the required boundary condition at that end of the domain.

Also, once these measurements have been taken, the characteristic speeds give a time-step size restriction. For a finite difference scheme, the time step must be limited by a CFL condition (Strikwerda, 1989). Here, that restriction is $\Delta t \leq \Delta x/182{,}879$.

Why, then, did the simplified model work so well? Analysis of the simplified model shows that the index with respect to t is 2. No initial conditions may be arbitrarily specified. Initializing u at an inconsistent value caused the small initial jump in current shown in the simulation results. So, there was in fact a problem with the simplified model, but it was less serious than the outright failure that befell the simulation based on the full model. Also, the canonical form of the simplified system consists of a single degenerate parabolic block with simple forcing. Two boundary conditions at the same domain endpoint therefore do determine a unique solution of the simplified model. Finally, there is no CFL condition limiting the time step.

The generalized eigenvectors form the transformation matrices P and Q

$$P = \begin{bmatrix} -1.19E-3 & 1.00 \\ 1.19E-3 & 1.00 \end{bmatrix}$$

$$Q = \begin{bmatrix} -4.21E+2 & 4.21E+2 \\ 5.00E-1 & 5.00E-1 \end{bmatrix} \tag{23}$$

that take the system to its canonical form

$$\begin{bmatrix} -5.47E-6 & \\ & 5.47E-6 \end{bmatrix} \boldsymbol{v}_t + \begin{bmatrix} 1 & \\ & 1 \end{bmatrix} \boldsymbol{v}_x + \begin{bmatrix} -1.40E+4 & 1.40E+4 \\ -1.40E+4 & 1.40E+4 \end{bmatrix} \boldsymbol{v} = \boldsymbol{0}. \tag{24}$$

Several things are apparent from the canonical form. As noted from the eigenvalues, one boundary condition must be enforced at each end of the domain. Furthermore, the mathematical properties of the simplified model are very different from those of the full model. The simplified model is parabolic and equivalent to an ODE in x, while the full model is hyperbolic. The analyses uncover these differences, and can be used to provide very understandable feedback to the engineer; specifically, that one boundary condition at the left domain endpoint must be removed, and one boundary condition must be enforced at the right endpoint. This means going out into the field and obtaining a new set of measurements at a new location, or inferring new information from existing data.

Compressible flow revisited: Adaptive boundary conditions

The boundary-condition evaluation method described earlier (Eq. 18) can be modified slightly to create a method by which a simulator could automatically adapt boundary conditions as required to form a well-posed problem.

The Courant-Isaacson-Rees (CIR) scheme (Courant et al., 1952) solves hyperbolic partial differential equations using a linear finite difference approximation to the characteristic form of the model equations. Consider a quasi-linear hyperbolic system in t and x over the domain $0 \le x \le 1,\ t \ge 0$:

$$u_t + B(u, t, x)u_x = f(u, t, x). \tag{25}$$

Let the domain be discretized into a set X of equispaced points, and let $x_i \in X$ be a particular point in that set. Initial data give the values of the dependent variables $u(x_i, 0)$.

This scheme evaluates the coefficient matrix B at each node. For example, consider the ith node in Figure 9. The frozen coefficient system is

$$u_t + B[u(0, x_i), 0, x_i]u_x = f[u(0, x_i), 0, x_i]. \tag{26}$$

Now, let L and Λ contain the left eigenvectors and the eigenvalues of $B[u(0, x_i), 0, x_i]$, respectively, so the characteristic form of the frozen coefficient system is

$$L\frac{du}{dt} = Lf[u(0, x_i), 0, x_i] \quad \text{along} \quad \text{diag}(Idx) = \text{diag}(\Lambda\, dt). \tag{27}$$

This system (Eq. 27) is then used as an approximation to the system after a small increment h in time t. Using the explicit Euler finite difference approximation to the directional derivative along each characteristic given equations of the form

$$l_i\left(\frac{u(h, x_i) - u_i^*}{h}\right) = l_i f[u(0, x_i), 0, x_i], \tag{28}$$

where u_i^* is the vector of values of u at the foot of the ith characteristics of the frozen coefficient system, calculated by interpolation between values at grid points on $t = 0$. For example, in Figure 9, $u_a^* = u(x_a, 0)$ is the value at the foot of characteristic a.

Let $v_i = l_i u_i^*$ and $g_i = l_i f_i(u_i^*, 0, x_i)$. Then the equations that give the value of $u(x_i, h)$ are

$$Lu(x_i, h) = v + hg. \tag{29}$$

This is the CIR scheme. For linear systems with simple or linear forcing, the coefficients on the left- and righthand sides are constant, so calculating new values after a time step at each node only requires solving the same system with multiple righthand sides.

Performing the same approximation at a boundary node, but retaining only the outward-directed characteristics, produces the system that partially determines the solution at that boundary (Eq. 18). If the characteristics associated with each line in that system are traced back from the next time $t + h$ to the current time t, and interpolation is used to determine the values at the feet of those characteristics, the righthand side is given in the same manner as in the CIR scheme, and is depicted graphically in Figure 10.

Performing Gauss elimination with row and column pivoting on this (possibly underdetermined) system gives a number of pivot variables that are determined by the characteristic information. The simulator could take this information, to-

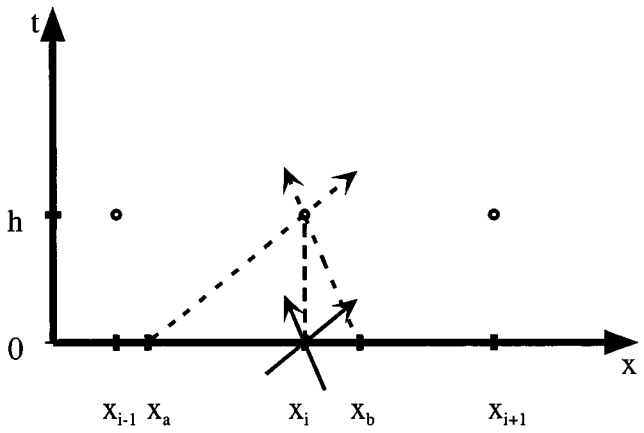

Figure 9. Stencil for CIR scheme.

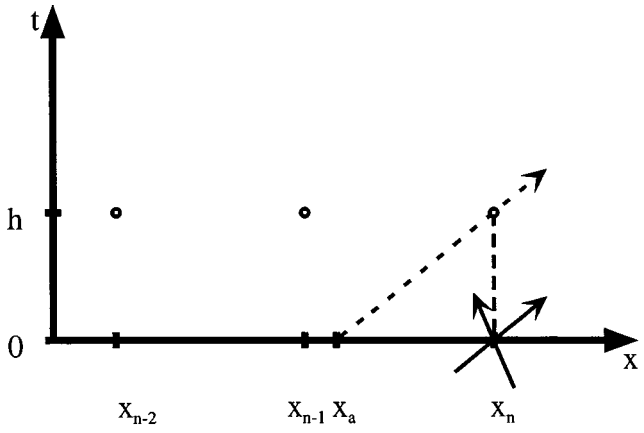

Figure 10. Modified CIR scheme for boundary point.

gether with the flowsheet topology and a specification of what variables refer to the same quantities in different unit models (for example, ρ in the pipe model refers to the same quantity as ρ_A in the model for valve 1), and attempt to set Dirichlet conditions on the remaining variables by equating values at the boundary to those in the adjacent unit, in order to form a fully determined system.

For this problem, consider use of this adaptive boundary-condition scheme at the pipe ends, together with a Godunov scheme (Godunov et al., 1962) using Roe's Riemann solver (Roe, 1986) on the domain interior. Using the LAPACK routine rgg to solve the generalized eigenvalue problem, and allowing the quantities that appear in both the pipe and the valve models to be u, ρ, p, and i, the method described is able to adapt the boundary conditions as needed to maintain a well-posed problem.

Possible characteristic directions at the domain endpoints and corresponding boundary-condition regimes appear in Figure 11. Three characteristics directed into the domain correspond to supersonic flow into the pipe at that end, and three boundary conditions are required. Two characteristics directed inward and one outward occurs when flow enters the pipe at subsonic conditions, and two boundary conditions are required. One characteristic directed inward corresponds to subsonic flow out of the pipe, which requires one boundary condition. Finally, no inward characteristics represents supersonic (or choked) flow out of the pipe, and no boundary conditions are required. The conditions at the two ends of the pipe may occur independently in any combination. Because it is based on the characteristics, the modified CIR scheme at the boundary together with the boundary condition selection method can correctly adapt to any combination of these flow regimes.

The pressure profile appears in Figure 12. The dual rarefaction shown earlier in the short-time profile is replaced quickly by the evolving quasi-steady pressure gradient.

The boundary condition changes at the left end ($x = 0$) appear in Figure 13. The short-time results appear in the bottom frame, and results for the entire simulation appear in the upper frame. The method correctly adapts from one (ρ) to two (ρ and i) boundary conditions after the flow reversal. It correctly adjusts again when a sonic transition occurs, and enforces a third (p) boundary condition.

Boundary-condition changes enforced by the method at the right end ($x = 10$) appear in Figure 14. No flow reversal occurs, and the method correctly enforces a single boundary condition on ρ until the sonic transition at approximately 0.1 s. The method removes this boundary condition when it is no longer needed, and obtains the solution at the boundary entirely from characteristic information after the sonic transition.

Without any intervention from the engineer, or even any knowledge of the mathematical changes in the boundary-condition requirements for well-posedness that occur at flow reversals and sonic transitions, a simulator employing this method could successfully adapt the boundary conditions. The engineer need only provide information regarding what variables refer to the same physical quantities in the different unit models.

Boundary-condition placement, stability, and continuous dependence on data

Boundary-condition placement for partial differential equation models is typically motivated by the need to formulate a well-posed problem. Split boundary conditions for ordinary differential and differential–algebraic models are often motivated by stability considerations. These two analyses are very different, even when they produce the same results.

For example, consider a material balance for a single reactive species in a PFR in the presence of dispersion. For incompressible flow with a constant superficial velocity,

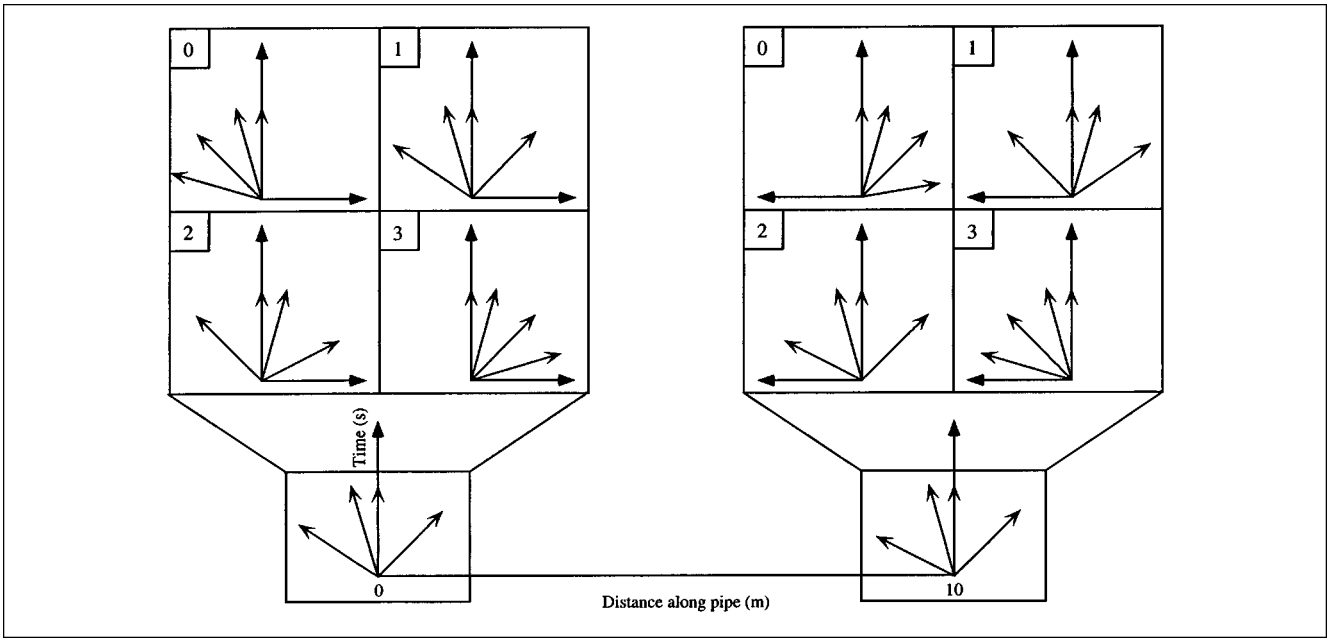

Figure 11. Characteristics and boundary-condition requirements for Euler equations of compressible flow.

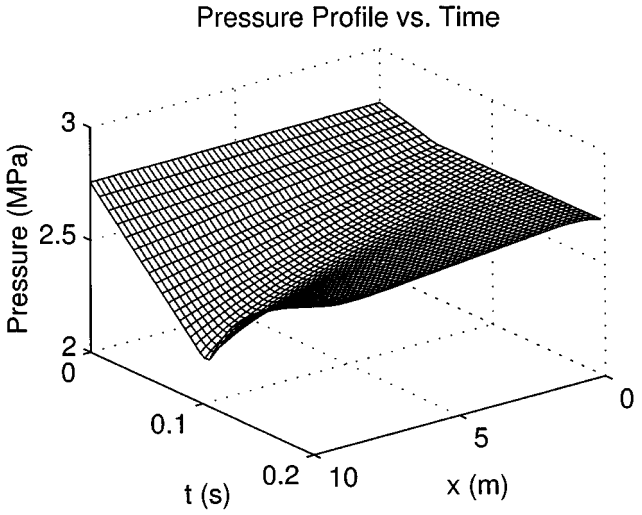

Figure 12. Pressure profile.

isothermal operation, irreversible reaction, and a first-order rate law, the material balance (written as a first-order system) is simply

$$C_t = D_a V_z - UV - kC$$

$$0 = C_z - V, \tag{30}$$

where C is the concentration of the species of interest, V is the first partial derivative of C with respect to position z along the length of the reactor, U is the superficial velocity, D_a is the diffusivity, and k is the reaction rate constant.

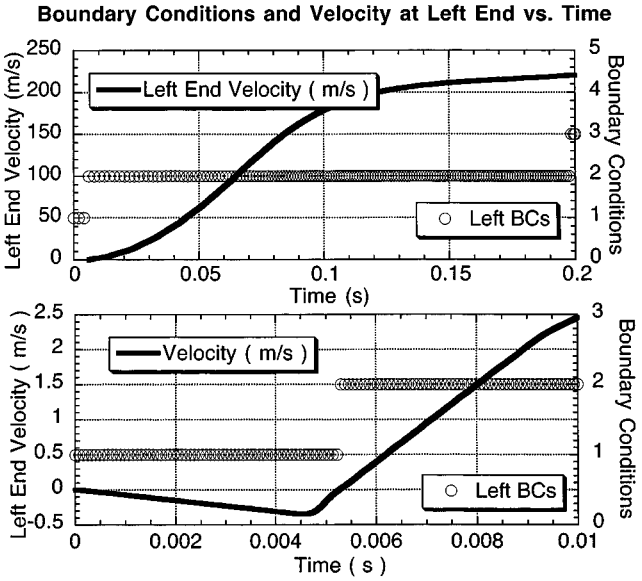

Figure 13. Results at left end of pipe.

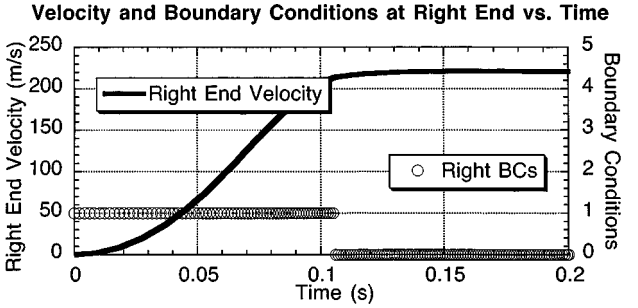

Figure 14. Results at right end of Pipe.

In canonical form, the system is

$$\begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} C^* \\ V \end{bmatrix}_t + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} C^* \\ V \end{bmatrix}_z + \begin{bmatrix} 0 & -1 \\ k & -\dfrac{U}{D_a} \end{bmatrix}\begin{bmatrix} C^* \\ V \end{bmatrix} = \mathbf{0}, \tag{31}$$

where $C^* = -C/D_a$. It consists of a single degenerate parabolic block, much like the simplified telegrapher's equations. However, here $\nu_t = 1$, so if two boundary conditions are enforced at the same side of the domain, the problem will be ill-posed, because it does not depend continuously on its data. One boundary condition must therefore be enforced at each side of the domain.

At steady state, the system is

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} C \\ V \end{bmatrix}_z + \begin{bmatrix} 0 & -1 \\ -\dfrac{k}{Da} & -\dfrac{U}{D_a} \end{bmatrix}\begin{bmatrix} C \\ V \end{bmatrix} = \mathbf{0}. \tag{32}$$

The stability of this system as an evolution problem in z is determined by the eigenvalues of the coefficient matrix, which are

$$\lambda = -\frac{U}{2D_a} \pm \frac{1}{2}\sqrt{\frac{U^2}{D_a^2} + \frac{4k}{D_a}}\,. \tag{33}$$

Because $D_a, k > 0$, $\sqrt{(U^2/D_a^2) + (4k/D_a)} > (U/D_a)$, and thus there is one positive and one negative eigenvalue. This means that the system is unstable as an evolution problem in both the forward and backward z-directions, and should instead be formulated as a split boundary-value problem.

Given a reaction zone of length L, and assuming that the reaction zone is fed by and flows into well-mixed vessels, the well-known closed-closed Danckwerts boundary conditions

$$C_{\text{vessel}} = -\frac{D_a}{U}C_z + C$$

$$C_z = 0 \tag{34}$$

produce both a well-posed initial-boundary value problem and a stable steady-state problem.

Conclusions

Modeling support and automated analysis tools have proven crucial for flow-sheet-scale dynamic simulation. Most existing tools are limited to models that consist only of ordinary differential and algebraic equations. This article outlines new model analysis tools that can be applied to models that include partial differential equations. In particular, they allow a simulator to identify models that are ill-posed due to model inconsistency or incorrect initial or boundary conditions. These tools also identify some models that do not depend continuously on their data, possibly as a result of a simple sign error on the part of the engineer. Finally, they can identify some models that are high index with respect to t. In each case, numerical method-of-lines solution methods cannot be expected to generate meaningful results. The tools therefore allow a simulator to screen models and identify the true cause of a simulation failure that results from the fundamental properties of the model equations themselves.

Literature Cited

Allgor, R., M. Berrera, L. Evans, and P. I. Barton, "Optimal Batch Process Development," *Comput. Chem. Eng.*, **20**, 885 (1996).

Barton, P. I., and C. Pantelides, "Modeling of Combined Discrete/Continuous Processes," *AIChE J.*, **40**, 966 (1994).

Brenan, K. E., S. L. Campbell, and L. R. Petzold, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, North-Holland, New York (1989).

Bujakiewicz, P., *Maximum Weighted Matching for High Index Differential Algebraic Equations*, PhD Thesis, Delft Univ. of Technology, Delft, The Netherlands (1994).

Campbell, S. L., *Singular Systems of Differential Equations*, Pitman, San Francisco (1982).

Campbell, S. L., and W. Marszalek, "The Index of an Infinite Dimensional Implicit System," *Math. Modelling Syst.* (1997).

Courant, R., and D. Hilbert, *Methods of Mathematical Physics*, Vol. 2, Interscience, New York (1962).

Courant, R., E. Isaacson, and M. Rees, "On the Solution of Nonlinear Hyperbolic Differential Equations by Finite Differences," *Commun. Pure Appl. Math.*, **V**, 243 (1952).

Demmel, J., and B. Kågström, "The Generalized Schur Decomposition of an Arbitrary Pencil $A-\lambda B$: Robust Software with Error Bounds and Applications. Part I: Theory and Algorithms," *ACM Trans. Math. Softw.*, **19**, 160 (1993).

Feehery, W. F., and P. I. Barton, "A Differentiation-Based Approach to Dynamic Simulation and Optimization with High-Index Differential-Algebraic Equations," *Computational Differentiation*, M. Berz, C. Bischof, G. Corliss, and A. Griewank, eds., SIAM, Philadelphia, PA (1996).

Feehery, W. F., J. Tolsma, and P. I. Barton, "Efficient Sensitivity Analysis of Large-Scale Differential-Algebraic Systems," *Appl. Numer. Math.*, **25**, 41 (1997).

Godunov, S. K., A. V. Zabrodin, and G. P. Prokopov, "A Computational Scheme for Two-Dimensional Non Stationary Problems of Gas Dynamics and Calculation of the Flow from a Shock Wave Approaching a Stationary State," *USSR Comput. Math. Math. Phys.*, **1**, 1187 (1962).

Golub, G., and C. van Loan, *Matrix Computations*, 2nd ed., The Johns Hopkins Univ. Press, Baltimore, MD (1989).

Jarvis, R., "DASOLV: A Differential-Algebraic Equation Solver," Tech. Rep., Imperial College, London (1992).

Jeffrey, A., *Quasilinear Hyperbolic Systems and Waves*, Pitman, London (1976).

Kikkinides, E. S., and R. T. Yang, "Simultaneous SO_2/NO_x Removal and SO_2 Recovery from Flue Gas by Pressure Swing Adsorption," *Ind. Eng. Chem. Res.*, **30**, 1981 (1991).

Kreiss, H.-O., and J. Lorenz, *Initial-Boundary Value Problems and the Navier-Stokes Equations*, Academic Press, New York (1989).

Kröner, A., W. Marquardt, and E. D. Gilles, "Computing Consistent Initial Conditions for Differential-Algebraic Equations," *Comput. Chem. Eng.*, **16**, 131 (1992).

Lieberstein, H. M., *Theory of Partial Differential Equations*, Academic Press, New York (1972).

Liska, R., and M. Y. Shashkov, "Algorithms for Difference Schemes Construction on Non-Orthogonal Logically Rectangular Meshes," *Proc. Int. Symp. on Symbolic and Algebraic Computation ISSAC'91*, New York (1991).

Liska, R., M. Y. Shashkov, and A. V. Solovjov, "Support-Operators Method for PDE Discretization: Symbolic Algorithms and Realization," *Math. Comput. in Simulation*, **35**, 173 (1994).

Longwell, E. J., "Dynamic Modeling for Process Control and Operability," *Advances in Instrumentation and Control*, Vol. 48, Chicago, p. 1323 (1993).

Marquardt, W., P. Holl, D. Butz, and E. D. Gilles, "DIVA—A Flowsheet Oriented Dynamic Process Simulator," *Chem. Eng. Technol.*, **10**, 64 (1987).

Martinson, W. S., and P. I. Barton, "A Differentiation Index for Partial Differential-Algebraic Equations," *SIAM J. Sci. Comput.*, **21**(6), 2295 (2000).

Martinson, W. S., and P. I. Barton, "Index and Characteristic Analysis of Nonhyperbolic Systems," *SIAM J. Sci. Comput.*, in press (2001).

Massobrio, G., and P. Antognetti, *Semiconductor Device Modeling with SPICE*, 2nd ed., McGraw-Hill, New York (1993).

Mattsson, S. E., and G. Söderlind, "Index Reduction in Differential-Algebraic Equations Using Dummy Derivatives," *SIAM J. Sci. Stat. Comput.*, **14**, 677 (1993).

Mayer, C., W. Marquardt, and E. D. Gilles, "Reinitialization of DAEs After Discontinuities," *Comput. Chem Eng.*, **19**, S507 (1995).

Oh, M., and C. C. Pantelides, "Process Modelling Tools and Their Application to Particulate Processes," *Powder Technol.*, **87**, 13 (1996).

Pantelides, C. C., "The Consistent Initialization of Differential-Algebraic Systems," *SIAM J. Sci. Sta. Comput.*, **9**, 213 (1988).

Park, T., and P. I. Barton, "State Event Location in Differential-Algebraic Models," *ACM Trans. Modelling Comp. Simulation*, **6**, 137 (1996).

Perkins, J. D., and R. W. H. Sargent, "SPEEDUP: A Computer Program for Steady-State and Dynamic Simulation and Design of Chemical Processes," *AIChE Symp. Ser.*, 78 (1982).

Petzold, L. R., "A Description of DASSL: A Differential-Algebraic Equation Solver," *Proc. IMACS World Congr.*, Montreal, P. Q., Canada (1982).

Pfeiffer, B., and W. Marquardt, "Symbolic Semi-Discretization of Partial Differential Equation Systems," *IMACS Symp., SC*-93, Lille, France (1993).

Reisszig, G., W. S. Martinson, and P. I. Barton, "Differential-Algebraic Equations of Index 1 May Have an Arbitrarily High Structural Index," *SIAM J. Sci. Stat. Comput.*, **21**(6), 1987 (2000).

Roe, P. L., "Characteristic-Based Schemes for the Euler Equations," *Annu. Rev. Fluid Mech.*, **18**, 337 (1986).

Strikwerda, J. C., *Finite Difference Schemes for Partial Differential Equations*, Wadsworth & Brooks/Cole, Pacific Grove, CA (1989).

van der Wijngaart, R. F., "Concepts of TRIFIT, A Flexible Software Environment for Problems in Two Space Dimensions," *Math. Comput. Simulation*, **37**, 505 (1994).

Manuscript received Nov., 15, 1999, and revision received Dec. 19, 2000.