

# Steady States for Chemical Process Plants: A Legacy Code, Time-stepping Approach

**Andreas Kavouras**

Dept. of Chemical and Biological Engineering, Princeton University, Princeton, NJ 08544

**Christos Georgakis**

Dept. of Chemical and Biological Engineering, Tufts University, Science and Technology Center, Medford, MA 02155

**C. T. Kelley**

Dept. of Mathematics, Center for Research in Scientific Computation, North Carolina State University, Raleigh, NC 27695

**Constantinos Siettos**

School of Applied Mathematics and Physical Sciences, National Technical University of Athens, GR 15780 Greece

**Ioannis G. Kevrekidis**

Dept. of Chemical and Biological Engineering, Program in Applied and Computational Mathematics, and Dept. of Mathematics, Princeton University, Princeton, NJ 08544

DOI 10.1002/aic.14199

Published online August 5, 2013 in Wiley Online Library (wileyonlinelibrary.com)

*Given a legacy dynamic simulator of a chemical process plant, we construct a computational procedure that can be “wrapped around” the simulator to compute its steady states (both stable and unstable) and their dependence on input parameters. We apply this approach to the Tennessee Eastman (TE) challenge problem presented by Downs and Vogel, who also provided a FORTRAN process model. Using the FORTRAN simulator as a black-box input-output map, we enable it to systematically converge to isolated solutions and study their stability and parametric dependence within the equation-free framework. The presence of neutrally stable modes in TE problem (due to so-called inventories), their interplay with the problem formulation and the convergence of the solution procedure is explored and rationalized. Interestingly, our time-stepper formulation can automatically take advantage of separation of time scales, when present, to enhance computational convergence. The approach enables legacy dynamic simulators to calculate several dynamic problem characteristics useful for controller design and/or process optimization. © 2013 American Institute of Chemical Engineers AIChE J, 59: 3308–3321, 2013*

**Keywords:** equation-free computations, legacy dynamic simulator, unstable processes, inventories, time-steppers

## Introduction

From *Mathematical Methods in Chemical Engineering: Matrices and their Application*<sup>1</sup> to *Linear Operator Methods in Chemical Engineering with Applications to Transport and Chemical Reaction Systems*,<sup>2</sup> linear algebra and linear operator theory constitutes a fundamental research tool in the study, by Neal Amundson and his academic progeny, of chemical reaction and transport processes. In this article, we show how relatively recent developments in computational linear algebra (and, in particular, matrix-free iterative linear algebra algorithms) can naturally enable dynamic simulators of chemical process systems to perform a plethora of additional tasks: stability quantification, parametric sensitivity analysis, and potentially much more.

Correspondence concerning this article should be addressed to I. G. Kevrekidis at yannis@princeton.edu.

Frequently, chemical process industries develop simulators that provide a dynamic representation of single units or integrated processes. Good dynamic simulators lead to a better understanding of the process, allow the exploration of case studies with varying design parameters, and provide insight toward linking manipulated variables with key process measurements. When a dynamic process model is sufficiently accurate in its location of steady states, it can also be used as a tool for optimization, or for controller design. The source codes of such dynamic models are often not accessible, hence the term “black box” or “legacy code”; even if they are accessible, their size (and possibly lack of documentation) may be daunting when current users consider adapting them to new purposes. When alternative tasks such as optimization or controller design are considered, one may end up coding the model again from scratch for the new purpose, instead of reusing the available legacy dynamic simulator. Computational enabling technologies exist<sup>3</sup> that can fill this gap by wrapping a master code around the legacy dynamic simulator

(the “time-stepper approach,” as we will refer to it throughout this article). The master code uses the time-stepper as a computational experiment; repeated brief calls to the time-stepper with prescribed initial conditions and parameter values enable the master code to perform tasks such as steady-state computation, stability, and parametric analysis. The equations “hidden” in the black box code need not be known, nor are they ever extracted in closed form; important quantities (residuals, actions of Jacobians, derivatives with respect to parameters) are estimated from the time-stepper output on demand; for this reason we have termed the approach “equation-free”. In effect, we solve the equations coded in the black box dynamic simulator without ever obtaining them in closed form.\* A fundamental computational task in process modeling is the systematic determination of steady states. Computing the process steady states, as well as evaluating Jacobians and derivatives with respect to parameters at a steady state underpins both controller design and optimization computations. When a dynamic model is available, a simple, direct way to find stable steady states is to integrate in time until nothing changes. Alternatively, when the steady-state equations are available, fixed point algorithms like the Newton–Raphson method can be implemented using the Jacobian of the steady-state equations and computational linear algebra. The steady states of the Tennessee Eastman (TE) problem are, however, not so easy to find either way; the “reference” steady state<sup>6</sup> is dynamically unstable, so that transient integration of the open-loop system moves away from it. Furthermore, the “legacy code” does not produce Jacobians of the steady-state equations as an output. Thus, performing Newton–Raphson to find steady states is not immediately implementable. Finally (and more importantly), because of the existence of two neutrally stable (zero) eigenvalues in the linearized problem, physically associated, as we will discuss below, with the existence of two inventories,<sup>7,8</sup> the steady-state problem is singular, allowing for infinitely many solutions. The Newton–Raphson will, therefore, not converge—care is required in formulating the problem so as to have meaningful unique solutions.

In this article, we describe and implement a computational superstructure that is “wrapped” around the TE legacy code, and enables the systematic location and continuation of its steady states (stable as well as unstable). The additional complication of neutrally stable eigenvalues—the “inventories”—will require an appropriate number of so-called “pinning conditions”, so that well defined algebraic problems having isolated solutions can be formulated. We will discuss three different forms of such well posed augmented problems. Because of the legacy nature of the problem, which does not explicitly provide derivatives of the state equations, we use matrix-free numerical fixed point and continuation methods. Newton–GMRES (for generalized minimal residual) is our method of choice here, and as we will see, appropriate time-stepper-based formulation of the steady-state conditions can lead to a significant reduction of the number of overall iterations. Indeed, as we will demonstrate and rationalize below, the existence of a separation between fast and slow time scales in the TE problem combines with short time integration to what can be thought of as effective “online model reduction”. An early reference to such an exploitation of GMRES for effective model reduction can be found in Wigton et al.<sup>9</sup>

\*It is worth noting that for our Tennessee Eastman (TE) illustrative example the closed form equations have been extracted from the original source code<sup>4</sup> to be used for model predictive control purposes and dynamic optimization.<sup>3</sup>

The article is organized as follows: In the TE Process, its Simulation and its Steady States Section, we start with a brief presentation of the TE problem and a detailed discussion of current literature approaches to the computation of its steady states; the reader more interested in our approach may, in a first reading, bypass some of the TE-specific details. In Using Time-Steppers to Compute Steady States Section, we discuss our time-stepper-based, matrix-free Newton–GMRES approach to the computation of TE steady states. We then discuss, in Augmentations to Circumvent TE Jacobian Singularity Section, the interplay of neutrally stable directions with fixed point computation and continuation, and derive three different well posed augmentations of the steady-state equations. Augmentation Results for the TE Time-Stepper Section contains our computational results, and discusses the model-reduction-like features of certain time-stepper formulations. Finally, in Discussion and Conclusions Section, we summarize, and then discuss the applicability of our implementation toward control and optimization tasks that require the computation of steady states as well as their local dynamic system characteristics.

## The TE Process, its Simulation, and its Steady States

A flow sheet that gives an overview of the units that constitute the TE process and illustrates the strong interlinking between them is given in Figure 1. The model representation of the process<sup>4</sup> consists of 50 state variables  $\mathbf{x} \in \mathbb{R}^{50}$ , 12 manipulated variables  $\mathbf{u} \in \mathbb{R}^{12}$ , and 41 measurements  $\mathbf{y} \in \mathbb{R}^{41}$ . The TE code returns numerical values of the time derivatives of the states—the right-hand-sides of the transient differential Eq. 1,  $\mathbf{f}(\mathbf{x}, \mathbf{u})$ , at the current states  $\mathbf{x}$  and manipulated variables  $\mathbf{u}$ . In addition, the measurements  $\mathbf{y}$ , which are dependent on the current  $\mathbf{x}$  alone, are returned by the call of the TE code.

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \quad (1)$$

$$\mathbf{y} = \mathbf{g}(\mathbf{x}) \quad (2)$$

A random number generator that—if wanted—distorts the measurements with noise is also supplied with the routine; we did not use this feature. Base case values of a reference steady state  $\mathbf{x}_0$  and  $\mathbf{u}_0$  are also provided.<sup>4</sup>

There is a ready supply of well-established continuation software packages, such as AUTO<sup>10</sup> and MATCONT,<sup>11</sup> for the computation and continuation of branches of steady states and limit cycles; yet there are often nontrivial model-related issues associated with their use. First, they most often use fixed-point solvers relying on the explicit availability of the system’s Jacobian—the matrix with the partial derivatives of the “right-hand-sides”—of the governing sets of equations. This poses an obstacle in the analysis of large scale systems for which traditional direct linear algebra techniques are not practical. Second, for models with rank-deficient Jacobians as the TE one, standard Newton–Raphson iterations will not converge. In the chemical process literature, this problem has been circumvented by the use of feedback control techniques, which dynamically stabilize the open-loop unstable TE process.<sup>12–14</sup> Time integration of the closed-loop dynamic system will then computationally locate the open-loop steady states; and by changing setpoints, the

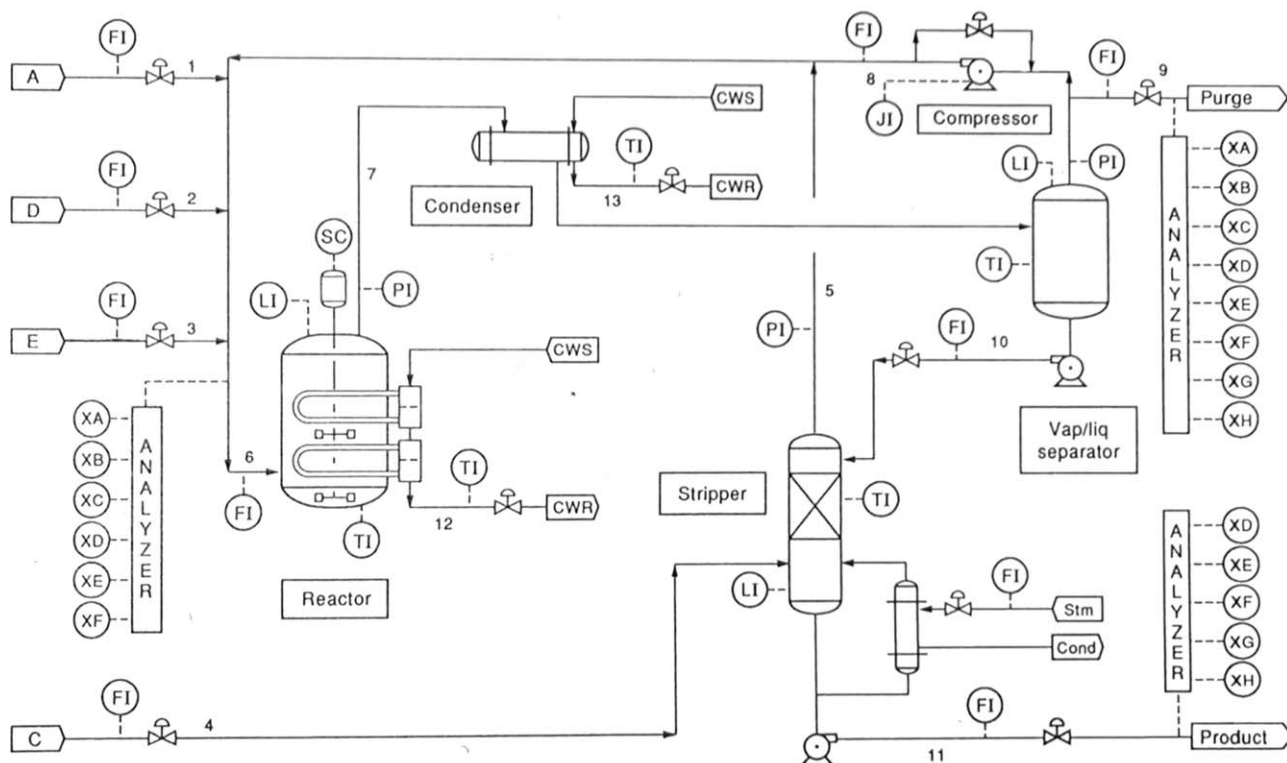


Figure 1. Flow sheet of the TE process.

open as well as closed-loop steady state dependence on operating parameters can be tracked.

At the beginning of our study, we linked the original Fortran TE code to Matlab with a mex file. We estimated the unavailable Jacobian  $\partial f/\partial x$  at the reference steady state using numerical derivatives and obtained its eigenvalues through Matlab. As it is well known, the eigenvalue analysis of the Jacobian at the reference state values yields several very large negative eigenvalues, two zero-eigenvalues (computationally  $\approx 10^{-6}$ ) and two pairs of complex conjugate eigenvalues with positive real parts. The large negative eigenvalues represent fast stable dynamics, whereas the positive real part eigenvalues indicate that the reference steady state is open-loop unstable. As a consequence, when time integration is performed to dynamically converge to the reference steady state, the process veers away from it and shuts down (violates operating constraints) within a short time. Figure 2 shows the eigenvalues of the reference steady-state Jacobian  $\partial f/\partial x$ . The eigenvalues with real parts larger than  $-10$  (of which there are 19) are given in the upper part of the plot; the lower part of the plot shows all eigenvalues.

For stable systems, it is common practice to determine steady states through time integration: when slightly perturbed from an isolated stable steady state, the simulation will asymptotically return to it. Depending on operating parameters, the TE process can also exhibit open-loop stable steady states and the transient open-loop response of the process after a perturbation from one such stable steady state (given in Table 1) is shown in Figures 3 and 4. This transient results from perturbing the steady state by lowering the feed stream of component A for a duration of 2.5-h process time (see Figure 3). Temporary decrease or loss of the A

feed stream was one scenario for disturbance rejection<sup>4</sup> and is considered one of the most challenging disturbances for control structures.<sup>8,15,16</sup> For this representative open-loop stable steady state, the applied perturbation is spontaneously damped, and the original steady state is asymptotically recovered after approximately 150-h process time. Time integration was here performed by a classical explicit Runge Kutta fourth-order scheme with individual time steps 1.5 s. At this representative stable steady state given in Table 1, two real part eigenvalues are again zero; but now the largest nonzero real part eigenvalue is  $-0.053$ , implying an overall neutrally stable (as opposed to asymptotically stable) system.

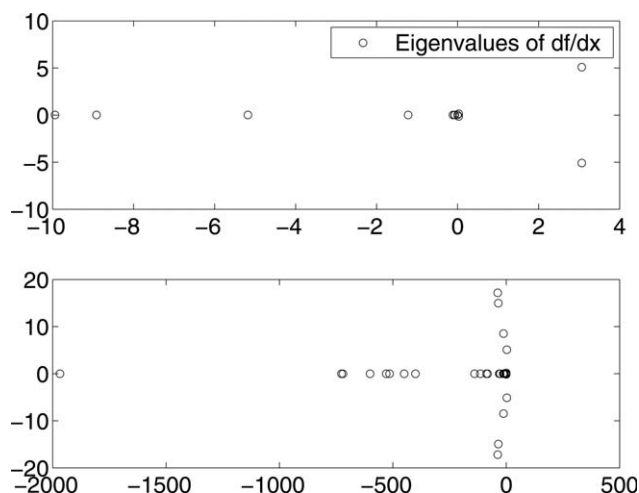


Figure 2. Plot of eigenvalues of  $\partial f/\partial x$  at base case.

**Table 1. Manipulated Variables and Key Process Measurements at an Open-Loop Stable Operating Point of the TE Process**

Manipulated Variables [%]	
D feed flow $u_1$	63.053
E feed flow $u_2$	53.980
A feed flow $u_3$	24.644
A+C feed flow $u_4$	61.302
Compressor recycle $u_5$	22.881
Purge valve $u_6$	42.322
Separator underflow $u_7$	38.256
Product flow $u_8$	46.421
Stripper steam $u_9$	47.446
Reactor cooling water $u_{10}$	35.620
Condenser cooling water $u_{11}$	19.620
Agitator speed $u_{12}$	50.000
Measurements	
Reactor pressure $y_7$ [mbar]	2678
Reactor level $y_8$ [%]	55.9
Reactor temperature $y_9$ [°C]	130.7
Stripper underflow $y_{17}$ [m <sup>3</sup> /h]	23.2

### Dynamically neutral directions, zero eigenvalues, and inventories

An interesting feature of the dynamic simulation is that the liquid levels of product separator and stripper do not regain their initial values after the other process measurements have equilibrated. Indeed, the levels in these vessels (the amount of material contained in them) do not affect the values of the other steady-state variables, these levels (alternatively, these vessels) are called “inventories,” and they are commonly present in chemical process plants. An infinity of steady states will, therefore, exist parameterized by the levels of the inventories, whereas all the other variables keep the same steady-state values. Infinities of steady solutions for a set of algebraic equations is associated with the Jacobian being rank-deficient (singular)—this is precisely associated with the two zero eigenvalues of the TE linearization. It is well known that such singularities (due to inventories) complicate closed-loop controller design, and the associated zero eigenvalues also interfere with steady-state approximation using singular-value-decomposition and approximate inverses.<sup>12,17</sup>

The second important feature is that these infinite solution families do not occur at every operating parameter setting—for most operating parameter settings the open-loop problem has no steady states. As the operating parameters vary, these entire families of steady states appear only for special parameter combinations. At a given set of operating parameters, all

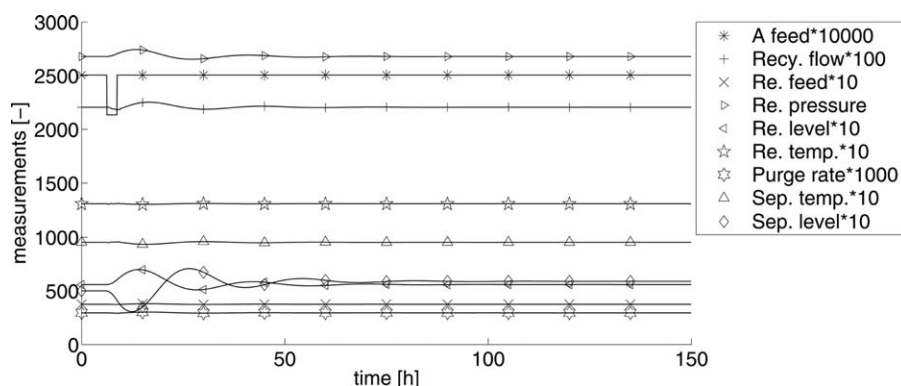
state values remain constant when the holdup in the inventories varies; the nonzero eigenvalues of the Jacobian  $\partial \mathbf{f} / \partial \mathbf{x}$ , however, do vary with the holdup, and the effect on the largest real part eigenvalue can be significant. For an increase of the levels of the product separator and the stripper at the TE process base case, the largest real part eigenvalue decreases, thus making the process less unstable. The reverse is observed when the levels of separator and stripper are decreased. The level of the product separator has a much stronger influence than that of the stripper—increasing the level of the product separator to 100% while keeping the stripper level at 50% decreases the leading eigenvalue real part to 2.528, whereas at the base case it is 3.065. These observations are easily rationalized, considering that the two vessels serve to dampen fluctuations of the concentrations of the liquid streams exiting from them. This effect of inventories on process dynamics can be taken advantage of in designing effective dynamic controllers for a process; for the nominal TE process in particular, dynamic operation is favorably affected by a large liquid holdup in the separator rather than the stripper.

### Obtaining TE steady states through feedback control

In engineering practice, unstable steady states are stabilized through the addition of control loops. These give rise to closed-loop dynamic systems possessing the same steady states as the open-loop ones, but with different stability characteristics. Consider—as an illustration—the velocity formulation of a Single-Input, Single-Output (SISO) Proportional-Integral (PI) controller with gain  $G$ , time constant  $\tau$ , and setpoint of a measurement  $y_{\text{set}}$ , which can be viewed as a transient difference equation:  $u_{t+\Delta t} = u_t + G \cdot [(y_t - y_{t-\Delta t}) + (y_t - y_{\text{set}}) / \tau \cdot \Delta t]$ . In a control context, one aims at transforming an open-loop unstable system like (1, 2) into a stabilized system through addition of an appropriate set of such feedback controllers. If the closed-loop system dynamics converge to the desired setpoint  $y_{\text{set}}$ , for an equilibrium value  $u_{\text{set}}$  of  $u_t$ , we have also found an open-loop steady state for  $u = u_{\text{set}}$ . This approach has been used for the numerical computation of TE model steady states,<sup>18</sup> in the same way one would do it for physical experiments.

### Obtaining TE steady states by continuation methods

In general, the solution of square systems of algebraic equations of the form  $\mathbf{f}(\mathbf{x}, \mathbf{u}) = 0$  with  $\mathbf{f} \in \mathbb{R}^n$  for the states  $\mathbf{x} \in \mathbb{R}^n$  with prescribed parameters  $\mathbf{u} \in \mathbb{R}^m$  is performed using continuation methods and contraction mappings like the Newton–Raphson algorithm. When the Jacobian is



**Figure 3. Process measurements 1–stable steady-state.**



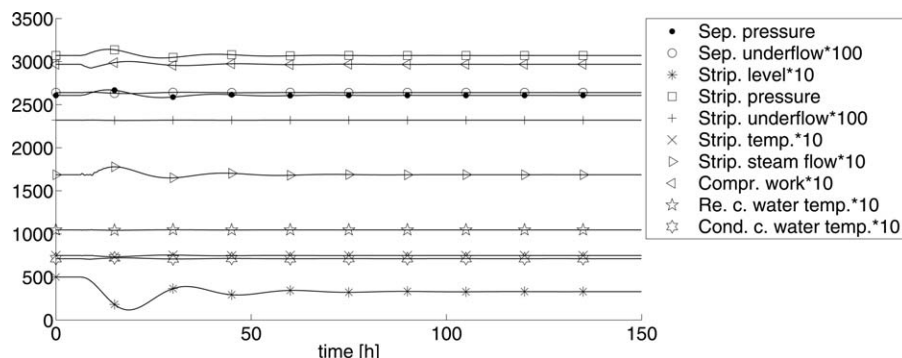


Figure 4. Process measurements 2-stable steady-state.

nonsingular, such algorithms trace both stable and unstable steady-state families as the parameters (for our purposes, manipulated variables)  $\mathbf{u}$  are varied; there is no need for closing control loops to stabilize the system so that its steady states can be found by integration.

One iteratively solves the linear set of equations

$$\mathbf{f}(\mathbf{x}, \mathbf{u}) + \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}} \cdot \delta \mathbf{x} = \mathbf{0} \quad (3)$$

for  $\delta \mathbf{x}$  and adding it to the current iterate  $\mathbf{x}$  until convergence. Inversion of the Jacobian  $\partial \mathbf{f} / \partial \mathbf{x}$  through the standard Gauss algorithm requires that the Jacobian has full rank, that is, is nonsingular. We already know (from eigenvalue analysis, but also from physical reasoning about inventory vessels) that the linearization of the TE equations is singular, and their steady states—if they exist—are not isolated. In Augmentations to Circumvent TE Jacobian Singularity and Augmentation Results for the TE Time-Stepper Sections below, we will discuss how to correctly formulate well-posed problems that account for the singularity issue.

### Using Time-Steppers to Compute Steady States

We want to find steady states of the problem (1), (2). In the normal “right-hand-side equals zero” formulation, when the equations are available, we solve for the vector of time derivatives of the system state variables to be zero. In the time-stepper formulation, we solve for (5) to be zero, where  $\Phi(\mathbf{x}_0, \mathbf{u}_0, \tau)$  is the result of integrating the system equations for a time horizon  $\tau$

$$\mathbf{x}_{t_0+\tau} = \mathbf{x}_{t_0} + \int_{t=t_0}^{t_0+\tau} \mathbf{f}(\mathbf{x}_{t_0}, \mathbf{u}_{t_0}) \cdot dt \equiv \Phi(\mathbf{x}_{t_0}, \mathbf{u}_{t_0}, \tau) \quad (4)$$

$$\hat{\mathbf{f}}(\mathbf{x}, \mathbf{u}, \tau) = \mathbf{x} - \Phi(\mathbf{x}, \mathbf{u}, \tau) = \mathbf{0}. \quad (5)$$

Clearly, a steady-state  $\mathbf{x}$  (a solution of  $\mathbf{f}(\mathbf{x}, \mathbf{u}) = \mathbf{0}$ ) also satisfies (5) for all  $\tau$ . In principle, steady states can be computed as fixed points of equations formulated with the time-stepper for any time-reporting horizon  $\tau$ . A number of articles in the recent numerical literature<sup>19–21</sup> as well as work in our group<sup>3,22–25</sup> discuss the enabling of temporal simulation codes to converge on steady-state solutions. The important element, however, in the time-stepper formulation, is the computation of the Jacobian for Newton–Raphson. In principle, if equations are available, integration of variational and sensitivity equations (with codes like ODESSA,<sup>26</sup>) will provide the derivatives with respect to initial conditions and parameter values that constitute Jacobian

elements. The solution of the linear equations that constitute a Newton step, however, need not be carried out through direct linear algebra (LU decomposition); they can be carried out iteratively (through methods like GMRES.<sup>27–30</sup> As GMRES computations actually require matrix-vector multiplications of the (Jacobian) matrix with a sequence of computed vectors, GMRES can be performed in a matrix-free fashion, using only calls to the integrator, as we will illustrate below. Partial derivative computations can thus be circumvented.

These matrix-vector multiplications can be performed at a rather low computational cost by calling the time-stepper from nearby initial conditions allowing the estimation of the action of the linearization of the map  $\Phi(\mathbf{x}, \mathbf{u}, \tau)$  on known vectors, say  $\mathbf{r}$ , as

$$\frac{\partial \Phi(\mathbf{x}, \mathbf{u}, \tau)}{\partial \mathbf{x}} \cdot \mathbf{r} = \frac{\partial \Phi(\mathbf{x} + \epsilon \mathbf{r}, \mathbf{u}, \tau) - \Phi(\mathbf{x}, \mathbf{u}, \tau)}{\epsilon} \quad (6)$$

where  $\epsilon$  is a small and appropriately chosen scalar. GMRES is a subspace method for the iterative solution of sets of linear equations  $\mathbf{A} \cdot \mathbf{x} + \mathbf{b} = \mathbf{0}$ . Starting from the initial residual vector  $\mathbf{r}_0 = \mathbf{A} \cdot \mathbf{x}_0 + \mathbf{b}$  a Krylov subspace (i.e., the span of a sequence of vectors arising from the matrix-vector products  $\langle \mathbf{r}_0, \mathbf{A} \cdot \mathbf{r}_0, \mathbf{A}^2 \cdot \mathbf{r}_0, \mathbf{A}^3 \cdot \mathbf{r}_0, \dots \rangle$ ) is generated. When  $\mathbf{A}$  is the Jacobian of a nonlinear function  $\mathbf{f}(\mathbf{x}, \mathbf{u})$ , the matrix vector products are retrieved from directional derivatives. Calculating a sequence of orthonormalized Krylov bases leads to the equality  $\mathbf{A} \cdot \mathbf{Z}_k = \mathbf{Z}_{k+1} \cdot \mathbf{H}_{k+1}$ , where  $\mathbf{Z}_k$  consists of the orthonormal basis vectors and  $\mathbf{H}_{k+1}$  is a Hessenberg matrix.

$$\mathbf{A} \cdot (\mathbf{x}_0 + \mathbf{Z}_k \cdot \mathbf{c}) + \mathbf{b} = \mathbf{Z}_{k+1} \cdot \mathbf{H}_{k+1} \cdot \mathbf{c} + \mathbf{r}_0 = \mathbf{r} \quad (7)$$

At the  $k+1$  step, the linear least squares problem (8) of order  $k+1$  is solved to find the solution that minimizes the residual over the subspace.

$$\|\mathbf{H}_{k+1} \cdot \mathbf{c} + \|\mathbf{r}_0\| \cdot \mathbf{e}_1\| \rightarrow \min \quad (8)$$

An extensive survey of this family of Newton–Krylov methods can be found in Knoll and Keyes.<sup>31</sup> For our calculations with GMRES, we used the code presented in Kelley.<sup>29</sup>

If the time-stepper (5), that is,  $\hat{\mathbf{f}}(\mathbf{x}, \mathbf{u}) = \mathbf{0}$  is used instead of  $\mathbf{f}(\mathbf{x}, \mathbf{u}) = \mathbf{0}$ , the iterative GMRES solver takes particular advantage of the eigenvalue structure of  $\partial \Phi(\mathbf{x}, \mathbf{u}, \tau) / \partial \mathbf{x}$ . Newton–GMRES solvers perform favorably with clustered eigenvalues of the Jacobian, which can arise when using time-steppers to solve  $\hat{\mathbf{f}}(\mathbf{x}, \mathbf{u}) = \mathbf{0}$ ; under such circumstances good solutions are often found after a relatively small number of iterations (in low-dimensional Krylov subspaces).

## Online effective model reduction

What makes this matrix-free computational approach an interesting one to combine with time-steppers, is that it can easily exploit time-scale separation when it is present in the problem of interest, and this is indeed the case in the TE process. Probably, the first archival observation of this effect can be found in Wigton et al.<sup>9</sup> It is obvious from the eigenvalue structure of the Jacobian  $\partial \mathbf{f} / \partial \mathbf{x}$  given in Figure 2 that the dynamics of the TE problem evolve on drastically different time scales. Integrating the differential equations with a time-stepper over large time horizons effectively reduces the problem size (in a linear problem it eliminates the components along eigenvectors associated with the fast stable dynamics from the output  $\mathbf{x}_{t_0+\tau}$ ). The advantage of using GMRES is that the explicit calculation and storage of the Jacobian is not required.

Now, consider the eigenvalues of the Jacobian  $\partial \Phi(\mathbf{x}, \mathbf{u}, \tau) / \partial \mathbf{x}$ . At steady state and for perfect integration these are  $\exp(\lambda \cdot \tau)$ , where  $\tau$  is the reporting horizon and  $\lambda$  denotes the respective eigenvalues of the Jacobian  $\partial \mathbf{f} / \partial \mathbf{x}$ . As the reporting horizon,  $\tau$ , of the time-stepper is increased, the fast eigenmodes of the time-stepper linearization move toward zero, whereas the slow (unstable) ones stay around (outside) the unit circle. For timestepping, we used both a classical Runge Kutta explicit method of fourth-order and an implicit Euler method. Due to the stiffness of the system, we used individual time steps of 1 s with the explicit method, while we covered the complete reporting horizon in one single time step with the implicit Euler method. When we evaluated the eigenvalues of  $\partial \Phi(\mathbf{x}, \mathbf{u}, \tau) / \partial \mathbf{x}$  with the Runge Kutta time-stepper at the base case, the number of eigenvalues close to zero (absolute value below  $10^{-4}$ ) increased from 0 at reporting horizon 10 s to 8 at 50 s, 14 at 100 s, 18 at 500 s, 22 at 1000 s, 28 at 2000 s. Similar results were found for time integration with the implicit Euler method.

Figure 5 shows the eigenvalues of the time-stepper evaluation for various reporting horizons at the base case operating point; once more, for perfect integration these ought to be  $\exp(\lambda \cdot \tau)$ , whereas for numerical integration they are approximations of  $\exp(\lambda \cdot \tau)$  that depend on the particular method. As we will see below, this feature will enhance our computational efficiency in locating TE steady states through the time-stepper approach.

## Augmentations to Circumvent TE Jacobian Singularity

As the Jacobian of the TE model is singular, standard Newton–Raphson fails; already McAvoy<sup>17</sup> applied singular value decomposition to the Jacobian  $\partial \mathbf{f} / \partial \mathbf{x}$ , and reported the existence of two singular values equal to zero at the base case. Their analysis was carried out in the framework of the Arkun and Downs<sup>12</sup> method to identify gain matrices for processes that contain integrating tanks. McAvoy<sup>17</sup> used a singular value decomposition (SVD)-based pseudo inverse to further converge the steady state of the base case<sup>4</sup> (as the steady state they reported is not well converged) and then derive the gain matrix of the process through numerical derivatives.

Although  $\mathbf{f}(\mathbf{x}, \mathbf{u})$  is of dimension 50,  $\partial \mathbf{f} / \partial \mathbf{x}$  is rank 48 (it has two null vectors). In general, a change in one manipulated variable  $u_i$  results in a change  $\partial \mathbf{f} / \partial u_i$  of  $\mathbf{f}$ , which is not in the range of the Jacobian matrix  $\partial \mathbf{f} / \partial \mathbf{x}$ . As we discussed, the full nonlinear problem has either no steady states or infinitely many. In order to obtain a well-defined problem, one augments the set of equations  $\mathbf{f}(\mathbf{x}, \mathbf{u}) = 0$  from (1) with two additional algebraic equations (the so-called “pinning conditions”); more generally, one would need as many pinning conditions as independent inventories in the problem. The pinning conditions select two out of the two-parameter

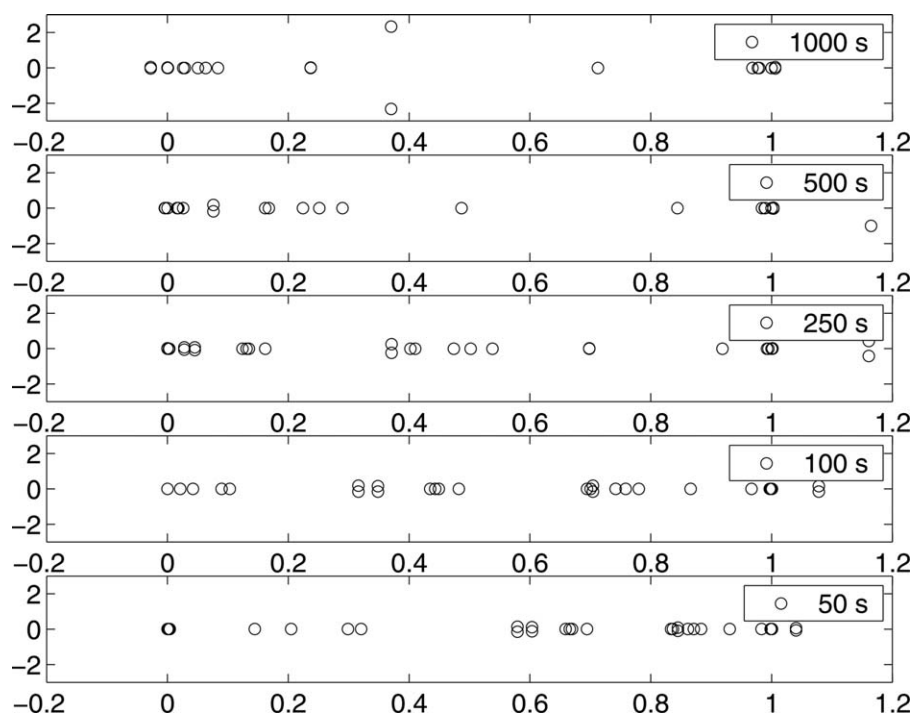


Figure 5. Plot of eigenvalues of the Jacobian  $\partial \Phi / \partial \mathbf{x}$  of the Runge Kutta time-stepper at the base case.

infinity of steady-state solutions. The augmented system has  $n+2$  equations, and  $n+2$  variables—the old  $n$  variables plus two more; typically the two additional unknowns will be two of the manipulated variables. In this approach, the fact that not all 12 manipulated variables can be selected at will is clearly seen: 10 manipulated variables are prescribed, and the values of the remaining two, for which steady states exist, result from the computation. Alternatively, one can prescribe ten process measurements; the values of all 12 manipulated variables as well as a single-steady state (out of the two-parameter infinity) will result from an appropriately augmented algebraic system.

Once a single-representative solution has been found, the entire two-parameter family can be routinely reconstructed using the null vectors of the steady-state Jacobian. Clearly, while the inventories are vital for dynamic plant modeling, they are effectively superfluous for the determination of steady states. Note that in conventional PI control schemes each inventory is stabilized by manipulating one process variable in a closed-loop. An analogy, therefore, exists between well-posed fixed point/continuation problems (two additional pinning conditions, used to find two of the manipulated variables) and the closed-loop control approach (two control loops linking two manipulated variables with the two inventory holdups).

### Augmented steady-state problems

As discussed, the TE process Jacobian has a rank deficiency of two, because of the two inventories. For parameter values for which steady states exist, they appear as a two-parameter infinity of solutions, parameterized by the inventory levels; the remaining steady-state variables remain constant over the family. We have implemented three different ways of obtaining nondegenerate formulations of the steady-state problem. We refer to the set of equations by  $\mathbf{F}$  and to their variables by  $\mathbf{X}$ .

Augmentation 1a. This formulation consists of the augmented set of equations

$$\mathbf{F}_{1a} = \begin{pmatrix} \mathbf{f}(\mathbf{x}, \mathbf{u}) \\ \mathbf{g}_l(\mathbf{x}) - \mathbf{y}_{l\text{set}} \end{pmatrix} = \mathbf{0} \quad (9)$$

where  $\mathbf{f}(\mathbf{x}, \mathbf{u})$  is the vector of 50 time derivatives of state variables of the system and  $\mathbf{g}_l(\mathbf{x})$  represents measurements affected by the inventory variables, which here are the levels of the product separator and stripper bottoms. The two additional “pinning” constraints, which in this case are  $y_{12} = g_{12}(\mathbf{x})$  and  $y_{15} = g_{15}(\mathbf{x})$ , select a single member of the two-parameter infinity of solutions; for the TE problem, these correspond to “pinning” measurements 12 and 15, that is, product separator and stripper levels. We solve for the steady state with 50% level in the two inventories (product separator and stripper); we have 52 equations, and can, therefore, solve for 52 unknowns, which we can arrange in the column vector  $\mathbf{X}_{1a}$ .

$$\mathbf{X}_{1a} = \begin{pmatrix} \mathbf{x} \\ \mathbf{u}_d \end{pmatrix} \quad (10)$$

The column subvector  $\mathbf{u}_d$  is of dimension 2 and contains two dependent manipulated variables (the ones that we will choose to solve for after prescribing the remaining ten). Our Newton step is then

$$\begin{pmatrix} \mathbf{f}(\mathbf{x}, \mathbf{u}) \\ \mathbf{g}_l(\mathbf{x}) - \mathbf{y}_{l\text{set}} \end{pmatrix} + \begin{bmatrix} \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}} & \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}_d} \\ \frac{\partial \mathbf{g}_l(\mathbf{x})}{\partial \mathbf{x}} & 0 \end{bmatrix} \cdot \begin{pmatrix} \delta \mathbf{x} \\ \delta \mathbf{u}_d \end{pmatrix} = \mathbf{0} \quad (11)$$

The Newton converges when the augmented system above has a nonsingular Jacobian; our choice of two dependent manipulated variables is guided by this requirement. For this specific process, any combination of two from the 12 manipulated variables of the TE problem would give a nonsingular Jacobian; certain choices, however, lead to better conditioned problems, or problems that do not easily hit saturation boundaries during parameter-space continuation. From the process side, one might select as the new dependent variables two process flows that are closest in physical proximity and largest in effect on the two measurements, whose values are specified.

Augmentation 1b. A simple variation of the first formulation pins two states, involved in the equation(s) describing the rate of change of the inventory variables, instead of two measured variables. This formulation picks a solution out of the two parameter infinity by solving the set of equations

$$\mathbf{F}_{1b} = \begin{pmatrix} \mathbf{f}(\mathbf{x}, \mathbf{u}) \\ \mathbf{x}_l - \mathbf{x}_{l\text{set}} \end{pmatrix} = \mathbf{0} \quad (12)$$

where  $\mathbf{x}_l$  contains two elements of the complete state variable vector  $\mathbf{x}$ . These two variables  $\mathbf{x}_l$  must have nonzero contributions to the null-eigenvectors of  $\partial \mathbf{f}(\mathbf{x}, \mathbf{u}) / \partial \mathbf{x}$ , so that the two parameter infinity can be pinned down to a unique solution (again, any combination of two manipulated variables as the additional unknowns in the augmented system will lead to well-posed problems).

The associated Newton–Raphson scheme for augmentation 1b is

$$\begin{pmatrix} \mathbf{f}(\mathbf{x}, \mathbf{u}) \\ \mathbf{x}_l - \mathbf{x}_{l\text{set}} \end{pmatrix} + \begin{bmatrix} \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u})}{\partial x_1} & \cdots & \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u})}{\partial x_{n-1}} & \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u})}{\partial x_n} & \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}_d} \\ 0 & \cdots & 1 & \cdots & 0 & 0 & 0 \\ 0 & \cdots & \cdots & 1 & \cdots & 0 & 0 \end{bmatrix} \cdot \begin{pmatrix} \delta \mathbf{x} \\ \delta \mathbf{u}_d \end{pmatrix} = \mathbf{0} \quad (13)$$

Augmentation 2. Here, we substitute the last two lines of (12), that is, the condition  $\mathbf{x}_l = \mathbf{x}_{l\text{set}}$  directly into  $\mathbf{f}(\mathbf{x}, \mathbf{u}) = \mathbf{0}$ , so that the equations for this augmentation are  $\mathbf{F}_2 \equiv \mathbf{f}(\mathbf{x}, \mathbf{u}) = \mathbf{0}$ . Instead of pinning at specified positions (like augmentations 1a and 1b) augmentation 2 removes two state variables, which represent inventories, as unknowns from the problem and replaces them (as unknowns) by an equal number of parameters (manipulated variables). The values of these manipulated variables will now be found as a function of the remaining parameters.

The vector of 50 unknowns now consists of the 48-long sub-vector  $\mathbf{x}_f$ , that contains all but two state variables, augmented by the two additional  $\mathbf{u}_d$  as above.

$$\mathbf{X}_2 = \begin{pmatrix} \mathbf{x}_f \\ \mathbf{u}_d \end{pmatrix} \quad (14)$$

This leads to the Newton–Raphson scheme

$$\mathbf{f}(\mathbf{x}, \mathbf{u}) + \left[ \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}_f} \quad \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}_d} \right] \cdot \begin{pmatrix} \delta \mathbf{x}_f \\ \delta \mathbf{u}_d \end{pmatrix} = \mathbf{0} \quad (15)$$

We refer to the sets of steady-state equations of formulations 1a, 1b, and 2, that use the time-stepper formulation  $\hat{\mathbf{f}}(\mathbf{x}, \mathbf{u}, \Delta t)$  instead of the time derivatives  $\mathbf{f}(\mathbf{x}, \mathbf{u})$  by  $\hat{\mathbf{F}}$ .

Clearly, the results of augmentation 2 will be the same as the results of augmentation 1b (they use the same pinning); augmentation 1a uses a different pinning, and so it will find different representatives of the family of solutions. Upon convergence, we obtain the values of the two dependent manipulated variables for which a two-parameter infinity of solutions exists, along with a single representative of the family (the one selected by our pinning). The rest of the steady-state family for the manipulated variable values in question (the ten, we prescribed and the two, we found) is routinely obtained by using the null vectors of the degenerate problem Jacobian.

This discussion is independent of whether the code is available in a legacy form or explicitly (i.e., independently of whether we can evaluate the Jacobians from the equations, or we need to estimate them from the code). As a sanity check, we used the three above augmentations for the right-hand-side of the TE equations, using direct linear algebra (LU decomposition) on augmented Jacobians whose elements were estimated through finite differences from the TE code. We reproduced the optimal operating points<sup>32</sup> (in “optimization modes” 1 and 4 of this article). At the stable steady state of Table 1, the process operating constraints stay well within the bounds for safe operation. As Ricker and Lee<sup>15</sup> described, a rise in the reactor temperature causes enhanced formation of byproduct F.

## Augmentation Results for the TE Time-Stepper

To highlight the advantages of the Newton-GMRES approach in the time-stepper formulation, we calculated a steady state of the TE process using all three augmentations 1a, 1b, and 2. The steady state to be determined was designated by a change of the ratio of the D and E feed mass streams off the base case (decrease of  $u_1$  by 0.25, increase of  $u_2$  by 0.17391). For all three augmentations,  $u_7$  and  $u_8$  were chosen as the dependent manipulated variables in  $\mathbf{u}_d$ . This corresponds to the typical inventory control scheme, where the underflows of the inventories are manipulated in order to control the vessel holdups. In augmentation 1b, the two state variables  $x_{13}$  and  $x_{22}$  were arranged in  $\mathbf{x}_f$  to be pinned by the additional equations. In augmentation 2, these two state variables were kept at the values of the TE base case, so that  $\mathbf{x}_f$  contained all state variables but  $x_{13}$  and  $x_{22}$ . Convergence of the outer Newton–Raphson iteration was declared, when the quantity  $\hat{\mathbf{F}}^T \cdot \hat{\mathbf{F}}$  fell below  $10^{-10}$ .

The GMRES convergence is controlled by two tolerance parameters. The first (a so-called “forcing term,”  $\eta$ ) is the ratio, by which the residual of the linear set of equations

must be reduced before a sequence of inner GMRES iterations is declared successful, and the approximate solution of the linear set of Eqs. 11, 13, or 15 is accepted. The second parameter limits the maximum number of inner iterations performed in one outer (Newton) iteration. An  $\eta$  of 0.1 performed satisfactorily for all computations reported here. We did not limit the maximum number of inner iterations to a smaller number than the problem size (i.e., we allowed for a full solution of the set of linear equations by the GMRES method if necessary; this never occurred in the time-stepper formulation). Restarted GMRES<sup>29</sup> is also an option for keeping the size of the system solved small; we did not use it in this article.

Convergence results of the three augmentations for time-stepper-based Newton-GMRES are given in Tables 2 and 3 for both a Runge Kutta and an implicit Euler time-stepper. In particular, Table 2 gives the total number of calls to time-stepper  $\Phi(\mathbf{x}, \mathbf{u}, \Delta t)$  (for one reporting horizon) during all outer iterations until convergence. Note that each directional derivative, by which a direction is added to the Krylov subspace, involves one function call, that is, a call to the time-stepper over the reporting horizon. Furthermore, the number of outer Newton–Raphson iterations until convergence is displayed in Table 2. Integration over a time horizon requires repeated calls to the TE code. The total number of such calls of the TE code [right-hand-side evaluations of (1)] incurred for a full steady-state computation and the total CPU time for it is given in Table 3. Our Matlab programs and the TE code, that was connected to Matlab as a mex file, were run on Matlab Release 13 on a Pentium IV, 3-GHz computer under RedHat Linux.

The convergence results are strikingly different for the three problem augmentations. Augmentation 1a performed overall best with both time integration methods. Calling the time-stepper over increasing reporting horizons reduces the required number of time-stepper calls significantly. The same tendency is observed with augmentation 1b, but not with augmentation 2.

Figure 6 shows the full convergence history of augmentation 1a with the explicit Runge Kutta time-stepper and different reporting horizons. On the ordinate is the residual  $\hat{\mathbf{F}}^T \cdot \hat{\mathbf{F}}$ , and on the abscissa is the cumulative number of

**Table 2. Number of Time-Stepper Calls and Number of Newton–Raphson Iterations for the Continuation Step with Formulations 1a, 1b, and 2 as a Function of the Reporting Horizon  $\Delta t$**

$\Delta t$	1a	1b	2	1a	1b	2
Runge Kutta	TS Calls			NR Iters.		
10 s	201	315	406	8	14	11
50 s	236	252	408	10	10	10
100 s	224	208	367	10	9	9
250 s	132	185	365	8	10	9
500 s	130	164	361	9	11	9
1000 s	107	96	384	10	8	12
2000 s	96	102	399	12	14	13
3000 s	96	99	416	12	12	15
Implicit Euler						
50 s	245	251	321	10	10	8
100 s	226	247	407	11	10	10
500 s	111	164	374	9	11	10
1000 s	98	113	338	9	8	10
5000 s	62	85		7	8	



**Table 3. Total Number of Calls of the Right-Hand-Side of (1) and CPU Time for the Continuation Step with Formulations 1a, 1b, and 2 as a Function of the Reporting Horizon  $\Delta t$**

$\Delta t$	1a	1b	2	1a	1b	2
	$10^3 \times \text{RHS Calls}$			CPU Time		
Runge Kutta						
10 s	8.7	13.7	17.1	0.6	1.0	1.2
50 s	51.2	54.4	85.6	2.4	2.6	4.0
100 s	97.6	90.4	154	4.3	4.0	6.7
250 s	148	205	383	6.2	8.7	16.0
500 s	296	372	758	12.3	15.4	31.3
1000 s	508	448	1632	20.9	18.5	67.1
2000 s	960	1040	3400	39.3	42.5	139.6
3000 s	1440	1476	5352	58.9	60.8	218.1
Implicit Euler						
50 s	44.5	44.7	61.8	3.4	3.4	4.8
100 s	42.0	49.4	77.5	3.2	3.9	6.0
500 s	23.8	34.2	84.8	1.9	2.7	6.8
1000 s	22.5	24.2	75.6	1.8	1.9	6.8
5000 s	14.8	19.3		1.2	1.6	

time-stepper or function calls, which is equal to the cumulative number of inner GMRES iterations. Each marker in this plot represents the residual at an outer iteration, which was evaluated after completion of a sequence of inner GMRES iterations. Due to the inexact solution of the linear set of equations in the Newton–Raphson augmentation (11) by GMRES, the contraction of the residual with the Newton–Raphson iterations is slower than quadratic, even in the last few iterations. With increasing reporting horizon of the time-stepper, the decline of the residual vs. the number of function calls becomes steeper. In fact, with augmentations 1a and 1b, it takes about the same number of outer Newton–Raphson iterations to converge, but the dimension of the problem (the number of inner GMRES iterations for one approximate linear solve) reduces. The reason for this is that time integration, by eliminating the fast dynamic modes of the system, acts effectively as a preconditioner; the approximate linear solutions are obtained in an effectively smaller subspace than the original problem space. For augmentation 1a and the Runge Kutta time-stepper with a reporting horizon of 2000 s approximate, linear solves can be obtained in, roughly, a ten-dimensional (10-D) subspace (down from the

original full space size of 52). The corresponding problem becomes a compact perturbation of the identity, the eigenvalues of its linearization cluster, and better performance is expected from GMRES.<sup>24</sup>

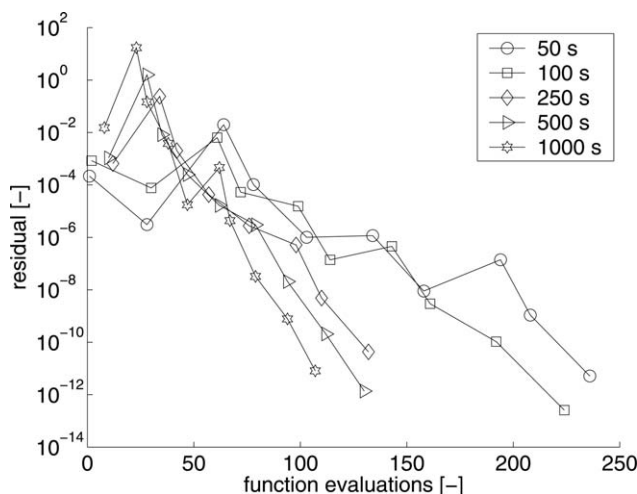
For a stable process, of course, a call with a relatively long time horizon would require no iterations; yet it would take a large computational effort. We try to exploit separation of time scales, so that the effective preconditioning benefit from a relatively short integration—combined with the Newton-GMRES algorithm—outperforms both direct integration and Newton-GMRES on a right-hand-side formulation. Of course in many problems (e.g., split step dynamic simulators, or in cases where the inner time-stepper involves a stochastic or microscopic code,<sup>22,23</sup>) the right-hand-side formulation is not an option. Due to the unstable nature of the TE process, the reporting horizon cannot be increased at will. Even if the input  $\mathbf{x}_t, \mathbf{u}_t$  to the time-stepper is the steady state itself (to computer roundoff), it will veer away from the steady state if the open-loop unstable process is integrated over too long reporting horizons. A rule of thumb for the selection of an appropriate time horizon would put its inverse in the gap between the strong stable eigenvalues of the system linearization and the absolute values corresponding to the slow (both stable and unstable) eigendirections; so, fast components die (get slaved to slow ones), yet we do not integrate longer than necessary, and the unstable modes do not “explode” away from the steady state of interest.

Augmentation 1b shows the same qualitative behavior as in Figure 6 with both time integration methods; in augmentation 2, however, an increase in the time-reporting horizon of the time-stepper does not reduce the number of function evaluations required for convergence (not shown). This suggests that the linearization of augmentation 2 does not exhibit eigenvalue clustering as 1a and 1b do. Plots of the eigenvalues of the Jacobian  $d\hat{\mathbf{F}}/d\mathbf{X}$  of augmentations 1a, 1b, and 2 with the Runge Kutta time integration at the base case values are given in Figures 7–9. Augmentation 2 with the Runge Kutta time-stepper produces one negative real eigenvalue with large absolute value at reporting horizons 500 and 1000, not shown in Figure 9.

An important factor for the GMRES method to perform well on a linear set of equations is the clustering of the eigenvalues of the problem matrix.<sup>29</sup> We clearly see that the eigenvalues of augmentations 1a and 1b indeed form clusters as the reporting horizon of the time-stepper is increased and the GMRES-time-stepper approach effectively reduces the problem size. It appears that these advantageous clustering characteristics do not arise in augmentation 2, and we believe that this underlies the observation that the problem size does not effectively reduce with increasing time-reporting horizon.

It might be argued, that the convergence criterion used for comparison between the three formulations is not fair, because the norm  $\hat{\mathbf{F}}^T \cdot \hat{\mathbf{F}}$  of the vector function scales differently with different problem augmentations and reporting horizons of the time-stepper. We thus also used—as a convergence criterion—the residual  $\mathbf{f}^T \cdot \mathbf{f}$  of the right-hand-side of (1), which corresponds to the set of equations that shall ultimately be solved; this criterion is the same for all problem augmentations and reporting horizons. Yet, the trends presented above did not change.

In earlier work,<sup>24</sup> GMRES was used to perform pseudoclength continuation for steady states of dynamical systems



**Figure 6. Convergence history of formulation 1a with the time-stepper.**

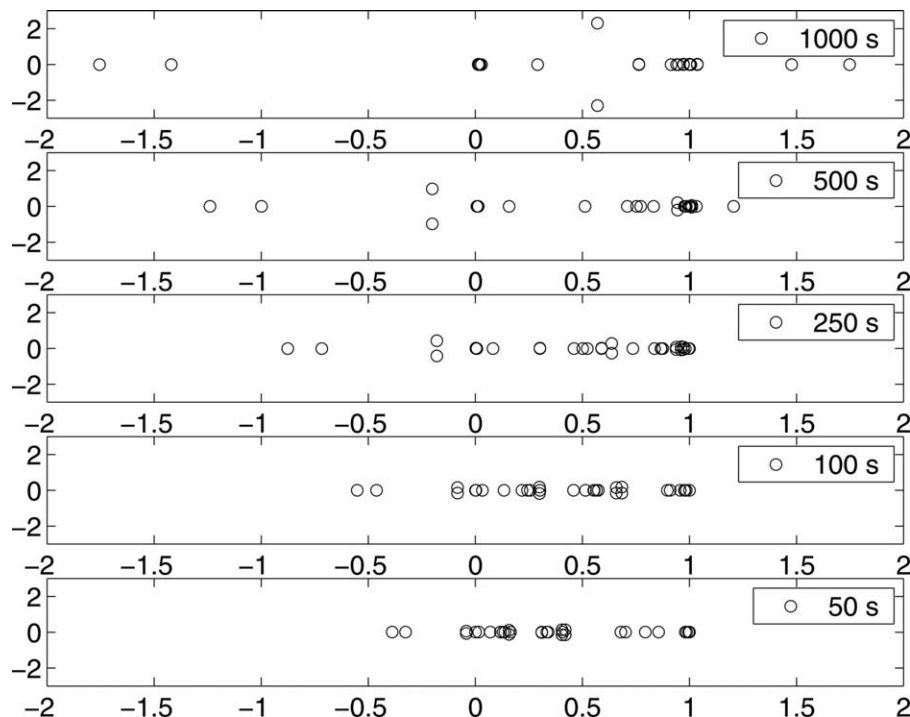


Figure 7. Plot of eigenvalues of Jacobian  $d\hat{F}_{1a}/dX_{1a}$  at base case (formulation 1a with the Runge Kutta time-step) at various reporting horizons  $\Delta t$ .

using time-steppers. They found, that the augmentation of the system equations with one additional equation (the pseudo-arclength equation for continuation) does not destroy the eigenvalue clustering of the augmented system.

The crucial question for the viability of the legacy time-stepper-based Newton-GMRES approach to steady-state computation is, of course, under which circumstances it performs better than direct time integration. Clearly, if one needs to

compute unstable branches of steady states or marginally stable solutions, integration is simply not an option. As we saw for the TE process, direct open-loop integration also does not constitute an option for problems with inventories, whether they be stable or unstable. If we do not know the right combination of manipulated variables, steady states simply do not exist. For problems with asymptotically stable steady states, performance depends drastically on the distribution of the

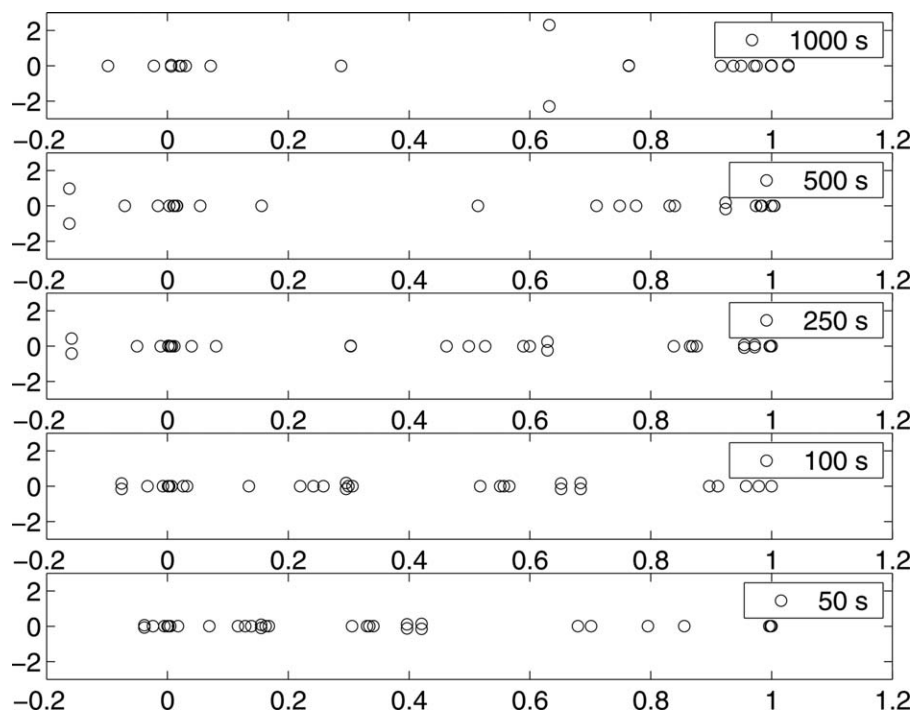
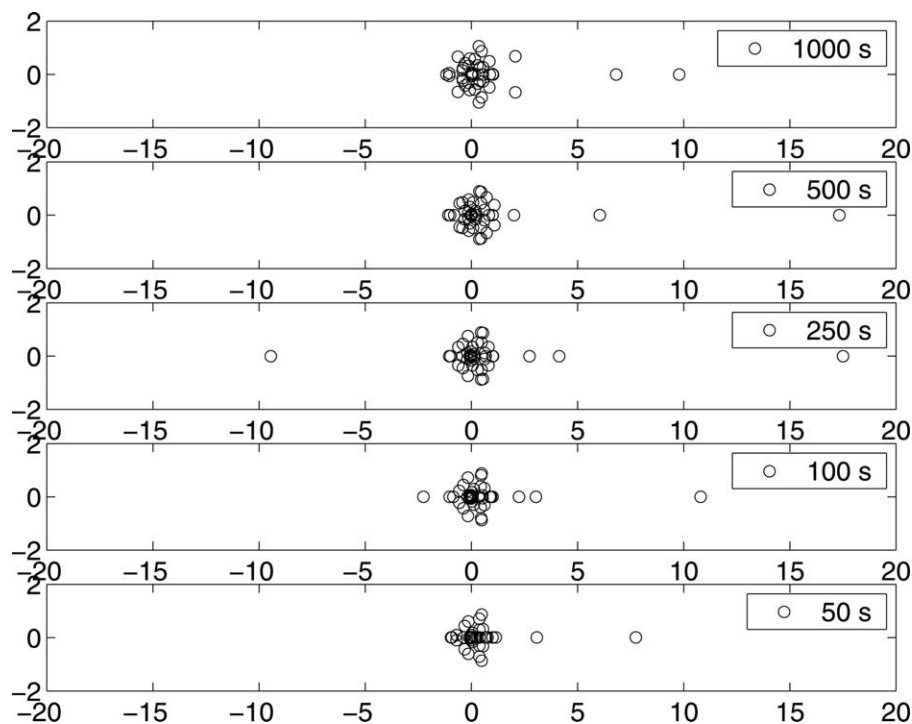


Figure 8. Plot of eigenvalues of Jacobian  $d\hat{F}_{1b}/dX_{1b}$  at base case (formulation 1b with the Runge Kutta time-step) at various reporting horizons  $\Delta t$ .



**Figure 9. Plot of eigenvalues of Jacobian  $d\hat{F}_2/dX_2$  at base case (formulation 2 with the Runge Kutta time-stepper) at various reporting horizons  $\Delta t$ .**

eigenvalues of the Jacobian  $\partial f/\partial x$  of the system differential Eq. 1. Time-stepper Newton-GMRES is in principle best suited for large systems, characterized by a few slow (possibly unstable) eigenmodes and a large majority of fast, stable eigenmodes.

In our discussion of time-scale separation and the corresponding gap between the system eigenvalues, we assumed for simplicity that the short numerical integration was perfectly accurate; when numerical integrators are used, the eigenvalues of the linearized time-stepper are not any more exponentials of the system eigenvalues multiplied by the reporting horizon  $\tau$ . What integration method is used will not change the fixed point, but may drastically change the eigenvalues of the time-stepper linearization (large explicit steps may even render a stable steady state numerically unstable). These linearized numerical time-stepper eigenvalues affect GMRES performance, as can be seen in Table 3. In our example, we used both an explicit and an implicit time-stepper and found—for our computational protocols—different trends of the CPU time for steady-state determination with GMRES, when the reporting horizon was increased. For the implicit Euler time-stepper, for example, we chose a single time-step equal to the entire reporting horizon. With reporting horizons larger than 100 s, the time step size was unreasonable in terms of accuracy, yet the resulting dampening was beneficial for steady-state determination. In fact, the reduction of the CPU time and number of right-hand-side (1) evaluations for increasing reporting horizons with the implicit Euler and augmentations 1a and 1b (as shown in Table 3) is entirely due to the GMRES “effective model reduction” and the resulting reduction of required time-stepper calls. The savings in the GMRES matrix inversion due to a smaller number of basis directions is not significant here. In conclusion, an increasing reporting horizon increases the computational expense for time integration, while possibly leading to improved GMRES

performance; optimal performance will result from balancing these two trends.

#### Further system augmentation

During continuation with chemical process models, operating limits of the process need to be observed and failure to do so results in process shutdown. In terms of the TE process, these shutdown limits are the pressure and temperature high limits in the reactor and the high or low limit of the reactor liquid level.

In the augmentations discussed above, we used the minimum number of pinning conditions that would give a well-posed problem—we prescribed 10 manipulated variables and allowed two to be computed from the model. In principle, we can now perform pseudo-arclength continuation along any one-parameter path in 10-D manipulated variable space. To make sure, however, that we do not frequently encounter operating limits, we may want to explore this 10-D space by adding further constraints that keep key system measurements at fixed levels away from operability boundaries. Additional such pinnings result in reduction of the number of independent manipulated variables (and corresponding growth of the number of dependent variables)—one additional dependent variable for each additional pinned measurement.

The constrained measurements  $y_c$  given in (16) are augmented to 1a formulation and thus the equations for constrained steady-state determination (17) are derived.

$$g_c(x) - y_c = 0 \quad (16)$$

$$\begin{pmatrix} f(x, u) \\ g_f(x) - y_{fset} \\ g_c(x) - y_c \end{pmatrix} = 0 \quad (17)$$

The associated Newton–Raphson scheme is (18), the variables are the states  $\mathbf{x}$ , the dependent manipulated variables  $\mathbf{u}_d$  and the manipulated variables  $\mathbf{u}_c$ .

$$\begin{pmatrix} \mathbf{f}(\mathbf{x}, \mathbf{u}) \\ \mathbf{g}_l(\mathbf{x}) - \mathbf{y}_{l,\text{set}} \\ \mathbf{g}_c(\mathbf{x}) - \mathbf{y}_c \end{pmatrix} + \begin{bmatrix} \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}} & \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}_d} & \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}_c} \\ \frac{\partial \mathbf{g}_l(\mathbf{x})}{\partial \mathbf{x}} & 0 & 0 \\ \frac{\partial \mathbf{g}_c(\mathbf{x})}{\partial \mathbf{x}} & 0 & 0 \end{bmatrix} \begin{pmatrix} \delta \mathbf{x} \\ \delta \mathbf{u}_d \\ \delta \mathbf{u}_c \end{pmatrix} = \mathbf{0} \quad (18)$$

In the same manner, formulations 1b and 2 can be augmented to solve for constrained steady states.

Such additional augmentations find their counterpart in the closed-loop approach to steady-state stabilization/computation—additional setpoints require further control loops and “slave” additional manipulated variables. Consider the set of control and output variables in control structure 2<sup>8</sup>: there, single SISO loops tie all manipulated variables (except for the agitator speed) to controlled variables in the process. The controlled variables are the reactor temperature, the recycle flow rate and the stripper underflow (measurements 9, 5, 17), the levels in the stripper, the separator, and the reactor (measurements 8, 12, 15), and besides concentrations in the reactor feed, the purge, and the product stream (measurements 23, 26, 27, 30, 38). Note that reactor pressure is not controlled by this control scheme.

In our steady-state augmentation formulation, if we impose the same change of reactor temperature from 120.4 to 121.6 and select the same sets of additional prescribed measurements and dependent manipulated variables as in Lyman and Georgakis,<sup>8</sup> the values of the manipulated variables we find are given in Table 4. Also given in Table 4 is an example, where the product flow rate is increased by 3.5% with the remaining 10 pinned measurements remaining at the base case TE process values.

Both cases of setpoint change show, that with the control structure proposed,<sup>8</sup> the manipulated variable  $u_9$  (stripper steam) undergoes extreme changes ensuing from the generic setpoint changes of reactor temperature and production rate; the control loop linking  $u_9$  with the concentration of compo-

nent E in the product stream may easily saturate. Ricker and Lee<sup>4</sup> also report that the stripper steam rate has negligible effect on the process behavior. Removing the stripper steam rate (manipulated variable  $u_9$ ) and the concentration of component E in the product stream (controlled variable  $y_{38}$ ) from the set of control and output variables of the control structure yields a set of 62 equations with two independent manipulated variables. Keeping now  $u_9$  at its base case value and continuing in reactor temperature—without attempting to pin  $y_{38}$ , which floats—we find that the reactor temperature can be increased to  $y_9 = 128.2$  [°C] before one of the dependent manipulated variables hits its bound. In this particular case, the compressor recycle valve position  $u_5$  hits its lower bound. Continuation in the product flow rate shows that it can be increased by 13.1% ( $y_{17} = 25.95 \text{ m}^3/\text{h}$ ) before the reactor pressure hits the high limit for normal operation of 2895 kPa.

Having an algorithm that easily computes and parametrically continues steady states subject to various sets of constraints (keeping a given set of measurements constant through varying a given set of manipulated variables) allows a systematic exploration of the system operability. In particular, one can explore whether generic setpoint changes can be in principle performed using prescribed sets of manipulated variables to keep all controlled measurements at their setpoints without control saturation. One can even envision pinning conditions that ask for a manipulated variable or state variable to marginally attain saturation as a computational approach for tracing operability boundaries.

## Discussion and Conclusions

Steady states of a dynamic process simulator were determined through matrix-free techniques (Newton-GMRES) that use sequences of input-output calls to the legacy code. Two formulations (a differential one, where the legacy code returns numerical values of the time-derivatives of the state variables, and an integral one, where it returns the result of integration over a time-reporting horizon) can be used for this purpose. This computational approach does not require prior stabilization of unstable processes through closing control loops.

The adaptation of this approach to problems with dynamically neutral directions (due to physical storage vessels, inventories) was explored. Inventory vessels lead to singularities in the system linearization, causing the steady-state problem to have with either zero or (families of) infinite solutions. Their steady-state levels have no influence on the other process states such as streams, concentrations, pressures, and temperatures nor on the inflow or outflow of the inventory itself. Because of the associated singular Jacobians,<sup>12</sup> the steady-state equations must be appropriately augmented to yield well-posed problems. The number of freely adjustable manipulated variables among the total of manipulated variables reduces by one for each inventory. For a steady state to exist, dependent manipulated variables need to be defined, whose values are determined by the solution procedure.

Although using a time-stepper-based Newton-GMRES to solve these appropriately augmented problems, we explored the effect of time scale separation and the time-stepper reporting horizon on GMRES performance. Increasing the time-stepper reporting horizon could lead under appropriate conditions to significant reduction of the GMRES iterates

**Table 4. Manipulated Variables of the TE Process at Base Case, at Reactor Temperature  $y_9 = 121.6^\circ\text{C}$  and at a Product Flow Rate that is Increased by 3.5% ( $y_{17} = 23.752 \text{ m}^3/\text{h}$ ) with the Set of Control and Output Variables Used in Control Structure 2<sup>8</sup>**

MV	Base Case	$y_9 = 121.6^\circ\text{C}$	$y_{17} = 23.752 \text{ m}^3/\text{h}$
1	63.0526	63.6584	64.1466
2	53.9797	54.2347	56.1574
3	24.6436	25.5426	25.0296
4	61.3019	61.6400	63.0798
5	22.2100	18.9699	22.7734
6	40.0637	41.6996	40.2187
7	38.1003	37.6623	39.8558
8	46.5342	46.8157	47.8766
9	47.4457	4.4434	95.6305
10	41.1058	40.8412	42.4343
11	18.1135	16.1119	20.8812



(time-stepper calls) required for the linear solve at each Newton iteration. We presented results pertaining to the effects of different augmentations as well as different numerical integration methods for the time-stepper.

A systematic steady-state computation/continuation routine can be a valuable tool for various purposes like stability analysis, controller design, and optimization. Determination of steady-state gains is valuable toward controller design; yet for problems with inventories and/or unstable setpoints these gains may be difficult to obtain (for the TE problem see McAvoy and Ye<sup>7</sup>). Systematic constrained continuation (implemented through further augmentations, as just discussed) can provide useful diagnostics in strongly nonlinear problems for gain sign change and control saturation.

For optimization, where the steady-state equations appear as constraints, a feasible approach can take advantage of a fast, systematic steady-state solver to turn the problem into an unconstrained one of reduced dimension. If the steady-state solver underlying a feasible optimization code is based on control loops and integration,<sup>7,18</sup> the closed-loop stability requirement will limit the steady states one can find, and thus possibly affect the optimization. Note that, thanks to the open structure of the original TE Fortran source code, steady-state optimization has been accomplished by linking the TE code to a standard optimization routine.<sup>32</sup>

It is interesting, especially for legacy numerical approaches (including time-stepper-based ones) where derivatives are not available, to explore the applicability of more direct search techniques (such as Nelder–Mead or Hooke–Jeeves algorithms) which have been experiencing a revival in both literature and practice.<sup>33</sup> “Black box” steady-state solvers, turning the problem into a feasible one, may be an asset in such computations. We can provide some supporting evidence for this statement: locating an open-loop stable steady state for the TE process (the one we presented in Table 1) was the result of a Nelder–Mead optimization, where our objective was to optimize steady-state stability. In particular, we tried to minimize the largest real part of the eigenvalues of the TE linearization at steady state. We solved this as a feasible problem; our steady-state solver underlying the objective function evaluation was a Newton-GMRES time-stepper-based one, and the Nelder–Mead direct search algorithm.<sup>34</sup> The optimization was performed over four manipulated variables that were chosen as “design parameters” (numbers 5, 6, 10, and 11); the two dependent manipulated variables in our augmentation were 7 and 8, and the remaining ones were kept at the base case values. Clearly, this methodology also has the potential to be useful for the optimization of the TE process, as well as other legacy-code-based optimization tasks.

## Acknowledgments

This work was supported by a research scholarship of the Max Kade foundation for Andreas Kavouras, which is gratefully acknowledged. I. G. K. also acknowledges the partial support of AFOSR (Dynamics and Control) and an NSF ITR grant.

## Notation

**A** = coefficient/mass matrix of a linear set of equations  
**b** = coefficient vector of a linear set of equations  
**c** = state vector **x** projected on the Krylow subspace  
**f** = function of the derivatives of the states toward time in the TE process

**f** = time-stepper-based set of equations for steady state determination in the TE process  
**F** = nondegenerate replacement for steady-state equations **f**, formulations 1a, 1b and 2  
**g** = correlation between state vector and measurements  
**H** = Hessenberg matrix in GMRES method  
**t** = time  
**r** = residual vector of a linear set of equations  
**u** = vector of manipulated variables  
**u<sub>d</sub>** = vector of dependent manipulated variables  
**x** = state vector of the TE process, solution vector of a set of linear equations  
**x<sub>r</sub>** = reduced state vector of the TE process  
**X** = variables of the nondegenerate steady-state equations **F**=0, formulations 1a, 1b, and 2  
**y** = measurements  
**Z<sub>k</sub>** = orthonormalized Krylow subspace with *k* basis vectors of the Jacobian in a Newton-GMRES solver

## Greek letters

**Λ** = eigenvalue  
**Φ** = right-hand-side of a time-stepper formulation of the differential evolution equation

## Literature Cited

- Amundson NR. *Mathematical Methods in Chemical Engineering: Matrices and Their Application*. Englewood Cliffs, NJ: Prentice-Hall, 1966.
- Ramkrishna D, Amundson NR. *Linear Operator Methods in Chemical Engineering with Applications to Transport and Chemical Reaction Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1985.
- Siettos CI, Pantelides CC, Kevrekidis IG. Enabling dynamic process simulators to perform alternative tasks: a time-stepper-based toolkit for computer-aided analysis. *Ind Eng Chem Res*. 2003;42(26):6795–6801.
- Ricker NL, Lee JH. Nonlinear modeling and state estimation for the Tennessee Eastman challenge process. *Comput Chem Eng*. 1995;19(9):983–1005.
- Albuquerque J, Gopal V, Staus G, Biegler LT, Ydstie BE. Interior point SQP strategies for large-scale, structured process optimization problems. *Comput Chem Eng*. 1999;23:543–554.
- Downs JJ, Vogel EF. A plant wide industrial process control problem. *Comput Chem Eng*. 1993;17(3):245–255.
- McAvoy TJ, Ye N. Base control for the Tennessee Eastman problem. *Comput Chem Eng*. 1994;18:383–413.
- Lyman PR, Georgakis C. Plant-wide control of the Tennessee Eastman problem. *Comput Chem Eng*. 1995;19(3):321–331.
- Wigton LB, Yu NJ, Young DP. GMRES acceleration of computational fluid dynamics, Vol. A85–1494. AIAA Conference, Washington DC, 1985:67–74.
- Doedel E, Tuckerman LC, Editors. Numerical methods for bifurcation problems and large-scale dynamical systems. *IMA Volumes in Mathematics and its Applications*, Vol. 119. New York: Springer-Verlag, 2000.
- Dhooge A, Govaerts W, Kuznetsov YA. MATCONT: a MATLAB package for numerical bifurcation analysis of ODEs. *ACM Trans Math Softw*. 2003;29:141–164.
- Arkun Y, Downs J. A general method to calculate input-output gains and the relative gain array for integrating processes. *Comput Chem Eng*. 1990;14(10):1101–1110.
- Alstad V, Skogestad S. Null space method for selecting optimal measurement combinations as controlled variables. *Ind Eng Chem Res*. 2007;46(3):846–853.
- Molina GD, Zumoffen DAR, Basualdo MS. Plant-wide control strategy applied to the Tennessee Eastman process at two operating points. *Comput Chem Eng*. 2011;35(10):2081–2097.
- Ricker NL, Lee JH. Nonlinear model predictive control of the Tennessee Eastman challenge process. *Comput Chem Eng*. 1995b;19(9):961–981.
- Price RM, Lyman PR, Georgakis C. Throughput manipulation in plantwide control structures. *Ind Eng Chem Res*. 1994;33:1197–1207.
- McAvoy TJ. A methodology for screening level control structures in plantwide control systems. *Comput Chem Eng*. 1998;22(11):1543–1552.
- Subramanian S, Georgakis C. A methodology for the steady-state operability analysis of plantwide systems. *Ind Eng Chem Res*. 2005;44(20):7770–7786.

19. Jarausch H, Mackens W. Solving large nonlinear systems of equations by an adaptive condensation process. *Numer Math.* 1987;50:633–653.
20. Shroff GM, Keller HB. Stabilization of unstable procedures: the recursive projection method. *SIAM J Numer Anal.* 1993;30(4):1099–1120.
21. Lust K, Roose D, Spence A, Champneys AR. An adaptive Newton Picard algorithm with subspace iteration for computing periodic solutions. *SIAM J Sci Comput.* 1998;19:1188–1209.
22. Gear CW, Hyman JM, Kevrekidis PG, Runborg O, Theodoropoulos K. Equation-free coarse-grained multiscale computation: enabling microscopic simulators to perform system-level tasks. *Commun Math Sci.* 2003;1(4):715–762.
23. Kevrekidis IG, Gear CW, Hummer G. Equation-free: the computer-assisted analysis of complex, multiscale systems. *AIChE J.* 2004;50(7):1346–1354.
24. Kelley CT, Kevrekidis YG, Qiao L. Newton-Krylov solvers for time-steppers. 2004. arXiv:math/0404374.
25. Qiao L, Erban R, Kelley CT, Kevrekidis IG. Spatially distributed stochastic systems: equation-free and equation-assisted bifurcation analysis. *J Chem Phys.* 2006;125:204108.
26. Leis JR, Kramer MA. The simultaneous solution and sensitivity analysis of systems described by ordinary differential equations. *ACM Trans Math Softw.* 1988;14(1):45–60.
27. Saad Y. Krylov subspace methods for solving large unsymmetric linear systems. *Math Comput.* 1981;37:105–126.
28. Saad Y. Practical use of some Krylov subspace methods for solving indefinite and nonsymmetric linear systems. *SIAM J Sci Stat Comput.* 1984;5(1):203–228.
29. Kelley CT. Iterative methods for linear and nonlinear equations. *SIAM Ser Frontiers Appl Math.* 1995:16.
30. Brown PN, Hindmarsh AC, Petzold LR. Using Krylov methods in the solution of large-scale differential-algebraic systems. *SIAM J Sci Comput.* 1994;15:1467–1488.
31. Knoll DA, Keyes DE. Jacobian-free Newton-Krylov methods: a survey of approaches and applications. *J Comput Phys.* 2004;193:357–397.
32. Ricker NL. Optimal steady-state operation of the Tennessee Eastman challenge process. *Comput Chem Eng.* 1995;19(9):949–959.
33. Kolda TG, Lewis RM, Torczon V. Optimization by direct search: new perspectives on some classical and modern methods. *SIAM Rev.* 2003;45(3):385–482.
34. Kelley CT. Iterative methods for optimization. *SIAM Ser Frontiers Appl Math.* 1999;19.

*Manuscript received Dec. 21, 2012, and revision received Jun. 9, 2013.*