Preconditioning for the Dynamic Simulation of Reaction-Transport Systems

Yuan He,*,† Weizhang Huang,‡ Kyle V. Camarda,† and Kenneth A. Bishop†

Departments of Chemical and Petroleum Engineering and Mathematics, The University of Kansas, Lawrence, Kansas 66045

This paper presents an investigation of preconditioning techniques for the transient simulation of reaction−diffusion systems that are commonly encountered in chemical process design applications. Three model-based preconditioners were proposed. Their performance was compared with that of a standard incomplete LU decomposition based preconditioner (P1). Two of the model-based preconditioners, P2 and P3, were constructed using only the transport part of the reaction−diffusion model: One of them was calculated using numerical differentiation while the other was based on analytical formulas. Since the transport properties are commonly assumed to be constant in transient numerical simulations, the resulting preconditioners are time-independent and need to be calculated only once, at the beginning of the simulation, and thus the efficiency of the simulation is significantly improved. The preconditioner P4 was constructed by adding chemical reaction contributions, updated only when necessary during the entire transient simulation. The numerical results for several test cases show that P4, by adding reaction correction terms, is more effective in improving the speed of the reactor simulation than the other two while maintaining the advantages of easy construction and low computational cost. In addition, it was demonstrated that the preconditioner P4 can also be used as an approximated Jacobian matrix for Newton's iteration, which may improve the efficiency of the simulation even further.

1. Introduction

Efficient simulation of chemical reactor models requires efficient solution of the linear systems of algebraic equations that arise from the discretization of the governing partial differential equations (PDE). Such linear systems are often large and sparse and commonly are solved numerically with an iterative method and typically a Krylov subspace method[1,2] such as the conjugate residual method (CR).[1] This work is partially focused on searching for effective preconditioning techniques for accelerating Krylov subspace methods, based on physical insight to reaction-transport systems. Generally speaking, the finite element or finite difference discretization of the PDE leads to a system having condition number of order $\kappa = O(h^{-2})$, where h denotes the mesh size. In other words, the condition number depends on the spatial dimension. The number of iterations required by a Krylov subspace method to reach a given level of precision in the solution is often proportional to h^{-1}. This slow convergence is seldom affordable in numerical simulation of reactor models, especially when a fine mesh is used. To improve the efficiency of numerical simulation, the preconditioning technique has proved to be an effective tool in numerical solution of linear systems. The basic idea is simple and is illustrated as follows.

Consider the numerical solution of a linear system $Ax = b$. A nonsingular matrix P (called a preconditioner) is sought such that (i) the linear system $Py = c$ can be solved at low cost and (ii) P is close to A in some sense (such as $\kappa(P^{-1}A) \ll \kappa(A)$). Two extreme cases are $P=I$ (which satisfies requirement (i) but not (ii)) and $P = A$ (which satisfies (ii) but not (i)). If a preconditioner P satisfying both (i) and (ii) can be found, then the preconditioned system $P^{-1}Ax = P^{-1}b$ (which is equivalent to the original system) can be solved with a Krylov subspace method more efficiently than the original system.

Preconditioners developed by various researchers include both specialized and general-purpose types. Some of the specialized preconditioners are proposed according to the convergence properties of certain methods. For example, there are certain preconditioners that enhance the eigenvalue clustering of matrix A so that the conjugate gradient method (CG) converges much faster; there are also "split" preconditioners ($M = LL^T$, therefore $L^{-1}AL^{-T}u = L^{-1}b$, $x = L^{-T}u$) that preserve the symmetric positive-definite property of the original system. Other specialized preconditioners may be developed from simplification of the governing equations of physical problems, e.g., the reaction-transport-based preconditioner[3] that comes with the *DASPK* solver.

General-purpose preconditioners are used more often. Some are developed from the stationary iterative methods for linear systems, such as the Jacobi, Gauss-Seidel, SOR, and SSOR preconditioners.[4] In reactor simulation problems, it was found that a block diagonal preconditioner like block-SSOR or block-Jacobi is often effective for reaction-diffusion equations.[5] Another group of commonly used preconditioners are developed from incomplete LU factorization (ILU).

When performing the LU factorization for matrix $A = LU$, one almost always finds that L (lower triangular matrix) and U (upper triangular matrix) contain non-

* To whom correspondence should be addressed. Tel.: 217-244-6154. Fax: 217-333-5052. E-mail: yuanhe@uiuc.edu.
† Department of Chemical and Petroleum Engineering.
‡ Department of Mathematics.

zero values, so-called *fill-ins,* in many locations where the original matrix A contains zeroes. Even worse, L and U are significantly denser even though A is sparse. Thus, exact LU factorization is not realistic for large-scale problems because of its storage requirement and huge computational cost. A compromise is to restrict the number of locations in L and U where fill-ins are allowed during the LU factorization process. This reduces not only the space requirement for saving L and U but also the number of operations since it is much less expensive to compute the product LU when they have a small percentage of nonzero entries. An LU factorization with restriction of the nonzero entry pattern on L and U is commonly called the *incomplete LU factorization.* The preconditioner that is defined as the product of the approximated L and U, i.e., $P = LU$, is referred to as *an ILU preconditioner*. Level one or two fill-ins[2] are widely used in practical computations. In the former case, the nonzero entry pattern of L and U is taken as the same as the original matrix A, while for the latter case, the pattern of L and U is taken as the same as that of the product of the lower and upper triangular matrixes resulting from the level one fill-in ILU factorization. Another technique, called *row-sum equivalence*, adds the sum of the dropped entries during the LU factorization process to the diagonal entries. It has been shown[5] that such a modified ILU preconditioner reduces the condition number from $O(h^{-2})$ to $O(h^{-1})$ for some cases. The fill-in entries can even be multiplied by a relaxation factor w ($0 \le w \le 1$; $w = 0.95$ is often used) before being added to the main diagonal, which does not reduce the dependence of the condition number on the grid resolution of the differential equations, but should significantly accelerate the convergence rate of the iterative solvers.[4] However, in a particular problem one should run tests to determine a suitable choice of w to optimize the efficiency of iterative solvers.

The objective of this paper is to investigate preconditioning techniques for improving efficiency of numerical simulation of transient reaction-transport systems for fixed-bed reactor models. In total, four ILU preconditioners based on the specific features of the underlying governing partial differential equations are developed. Their performances are examined for both dynamic and near-steady cases with small and large time step sizes. The cost of their construction is also addressed.

The organization of the paper is as follows. The physical problem is described in section 2, followed by a presentation of the numerical method for solving the governing PDEs in section 3. Four ILU-based preconditioners for reactive-transport systems are developed in section 4. Section 5 addresses the simulation cases for comparing the performance of these preconditioners. Numerical results and analysis are presented in section 6. Finally, conclusions are drawn in section 7.

2. Problem Description

A 2D fixed-bed reactor simulation code has been developed in the study to provide an efficient tool for performing parametric studies on different reaction systems and under different feed and radial cooling conditions. The pseudo-homogeneous model chosen here is a limiting case of a multiphase reactor model that assumes no transport barriers exist between fluid and solid phases. For the two-phase fixed-bed reactor, the model can be derived from a multiphase heterogeneous model by averaging physical properties of individual phases using porosity as the weighting factor and assuming that temperature and concentrations are equivalent between phases. It reads as

$$\frac{\partial C_k}{\partial t} = D_{er}\left(\frac{\partial^2 C_k}{\partial r^2} + \frac{1}{r}\frac{\partial C_k}{\partial r}\right) + D_{ez}\frac{\partial^2 C_k}{\partial z^2} - \frac{\partial (C_k \cdot u_z(z))}{\partial z} + M_k \quad k = 1, ..., N \qquad (1)$$

$$C_p\rho\frac{\partial T}{\partial t} = \lambda_{er}\left(\frac{\partial^2 T}{\partial r^2} + \frac{1}{r}\frac{\partial T}{\partial r}\right) + \lambda_{ez}\frac{\partial^2 T}{\partial z^2} - C_{pf}G\frac{\partial T}{\partial z} + H \qquad (2)$$

$$M_k = \rho_s(1-\epsilon)\sum_j^{N_k} w_{k,j} r_j \qquad (3)$$

$$H = \rho_s(1-\epsilon)\sum_j^{N}(-\Delta H_j) r_j \qquad (4)$$

subject to the boundary conditions

$$z = 0:\ -\frac{\partial C_k}{\partial z} = \frac{G}{D_{ez}\rho}(C_k - C_{k,0}), \quad k = 1, ..., N$$

$$-\frac{\partial T}{\partial z} = \frac{GC_{pf}}{\lambda_{ez}}(T - T_0)$$

$$z = L:\ \frac{\partial C_k}{\partial z} = 0, \quad \frac{\partial T}{\partial z} = 0$$

$$r = 0:\ \frac{\partial C_k}{\partial r} = 0, \frac{\partial T}{\partial r} = 0$$

$$r = R:\ \frac{\partial C_k}{\partial r} = 0, -\lambda_{ew}\left(\frac{\partial T}{\partial r}\right)_{r=R} = h_w(T - T_C) \qquad (5)$$

and the initial conditions

$$C_k = C_k^{\,0}, \quad T = T^0 \qquad (6)$$

Bulk concentrations C_k for the k species and the bulk temperature T are the unknown variables. It is noted that the governing eqs 1 and 2 are coupled nonlinear parabolic partial differential equations (PDE). The coupled parts are the reaction terms M_k and H, both of which contain the surface reaction rate r_j. The rate r_j has a general form of $r_j = S_j \exp(-A_j/RT_s)F(C_{k,s},C_{oxy,s})$, in which S is the pre-exponential factor, A is the activation energy, T_s is the catalyst surface temperature, and $C_{oxy,s}$ and $C_{k,s}$ are surface concentrations of oxygen and other species. In a pseudo-homogeneous model, surface values are regarded as the same as those in the bulk; therefore, F is indeed a nonlinear function of bulk concentrations.

The scales in temperature and concentration values are different, which often affects the accuracy and efficiency of numerical simulation. The following dimensionless variables are used in the simulation:

$$f_k = \frac{C_k}{C_{A0}}, \theta = \frac{T}{T_0}, z^* = \frac{z}{L}, r^* = \frac{r}{R} \qquad (7)$$

3. Numerical Methods

The initial-boundary value problem (1)−(6) is discretized and solved numerically on the computational

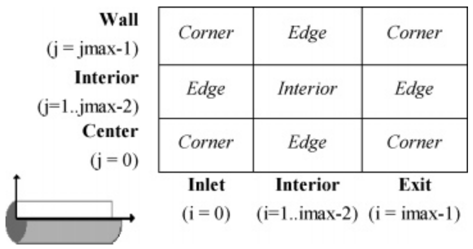

Figure 1. The layout of the computational domain.

domain $D = [0,1] \times [0,1]$ in the $z^* - r^*$ plane for a single reactor tube under the assumption of axial symmetry; see Figure 1. The domain is partitioned by a $i_{\max} \times j_{\max}$ rectangular grid with spacing $\Delta z^* = 1/(i_{\max} - 1)$, $\Delta r^* = 1/(j_{\max} - 1)$. The approximations of f (the dimensionless concentration) and θ (the dimensionless temperature) are denoted by $f_{i,j}$ and $\theta_{i,j}$, respectively, at the grid points $(z^*, r^*) = (z_i^*, r_j^*) = (i\Delta z^*, j\Delta r^*)$, $i = 0, ..., i_{\max} - 1$ and $j = 0, ..., j_{\max} - 1$. Since both the domain and the grid are rectangular, finite differences can be used in both the axial (from inlet to exit) and radial (from center to wall) directions for discretizing the PDEs.

The Method of Lines (MOL) is employed for discretizing the PDEs. In the MOL, the PDEs are first discretized in space (using finite differences); the resulting system of ordinary differential equations (ODE) or ordinary differential-algebraic equations (DAE) is then solved using an ODE or DAE solver. MOL allows the spatial and time derivatives to be handled separately. More importantly, the ODE or DAE system can be readily solved by many existing ODE/DAE codes.

In this study, central finite differences are used to discretize spatial derivatives to attain second-order accuracy in the truncation error. The discrete form of eqs 1 and 2 in dimensionless variables read as

$$F_{i,j,k} \equiv \frac{\mathrm{d}f_{i,j,k}}{\mathrm{d}t} - \left[\frac{D_{er}}{R^2 r_j^* \Delta r^*}\left(r^*_{j+1/2}\frac{f_{i,j+1,k} - f_{i,j,k}}{\Delta r^*} - r^*_{j-1/2}\frac{f_{i,j,k} - f_{i,j-1,k}}{\Delta r^*}\right) + \frac{D_{ez}}{L^2(\Delta z^*)^2}(f_{i+1,j,k} - 2f_{i,j,k} + f_{i-1,j,k}) - \frac{1}{2L\Delta z^*}(u_{i+1}f_{i+1,j,k} - u_{i-1}f_{i-1,j,k}) + \frac{M_{i,j,k}}{C_{A0}}\right] = 0$$
$$i = 0, ..., i_{\max} - 1, j = 0, ..., j_{\max} - 1, k = 1, ..., N \quad (8)$$

$$G_{i,j} \equiv C_p\rho\frac{\mathrm{d}\theta_{i,j}}{\mathrm{d}t} - \left[\frac{\lambda_{er}}{R^2 r_j^* \Delta r^*}\left(r^*_{j+1/2}\frac{\theta_{i,j+1} - \theta_{i,j}}{\Delta r^*} - r^*_{j-1/2}\frac{\theta_{i,j} - \theta_{i,j-1}}{\Delta r^*}\right) + \frac{\lambda_{ez}}{L^2(\Delta z^*)^2}(\theta_{i+1,j} - 2\theta_{i,j} + \theta_{i-1,j}) - \frac{C_{pf}G}{2L\Delta z^*}(\theta_{i+1,j} - \theta_{i-1,j}) + \frac{H_{i,j}}{T_0}\right] = 0$$
$$i = 0, ..., i_{\max} - 1, j = 0, ..., j_{\max} - 1 \quad (9)$$

where $M_{i,j,k}$ and $H_{i,j}$ are the coupled reaction/generation terms that depend on both temperature and concentration profiles. It is noted that the differencing of the PDEs is extended to the boundary points (with indices $i = 0$, $i = i_{\max} - 1$, $j = 0$, or $j = j_{\max} - 1$) for a second-order treatment of the Neumann boundary conditions (see the discussion in the following paragraph). The discrete mass and energy balance equations must be solved together with the Ergun equation,[6] a one-dimensional momentum balance equation predicting pressure drop along a packed bed. The kinetic model for the partial oxidation of *o*-xylene into phthalic anhydride used is proposed by Calderbank et al.;[7] see Figure 2.

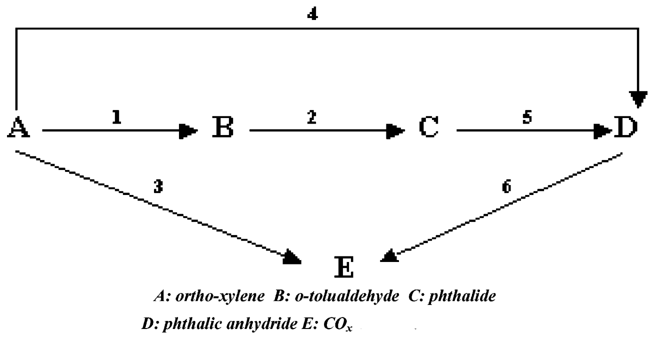
4
A
1
B
2
C
5
D
3
6
E
A: ortho-xylene B: o-tolualdehyde C: phthalide
D: phthalic anhydride E: CO$_x$

Figure 2. The reaction network for partial oxidation of *o*-xylene into phthalic anhydride.

The boundary conditions are treated similarly ($i = 0, ..., i_{\max} - 1$, $j = 0, ..., j_{\max} - 1$, $k = 1, ..., N$):

$$z^* = 0: \frac{f_{-1,j,k} - f_{1,j,k}}{2L\Delta z^*} = \frac{G}{D_{ez}\rho}(f_{0,j,k} - f_{feed,k}), \frac{\theta_{-1,j} - \theta_{1,j}}{2L\Delta z^*} = \frac{GC_{pf}}{\lambda_{ez}}(\theta_{0,j} - 1)$$

or

$$z^* = 0: f_{-1,j,k} = \frac{2L\Delta z^* G}{D_{ez}\rho}(f_{0,j,k} - f_{feed,k}) + f_{1,j,k}, \theta_{-1,j} = \frac{2L\Delta z^* GC_{pf}}{\lambda_{ez}}(\theta_{0,j} - 1) + \theta_{1,j} \quad (10)$$

$$z^* = 1: f_{i_{\max},j,k} = f_{i_{\max}-2,j,k}, \theta_{i_{\max},j} = \theta_{i_{\max}-2,j} \quad (11)$$

$$r^* = 0: f_{i,-1,k} = f_{i,1,k}, \theta_{i,-1} = \theta_{i,1} \quad (12)$$

$$r^* = 1: f_{i,j_{\max},k} = f_{i,j_{\max}-2,k}, \theta_{i,j_{\max}} = \theta_{i,j_{\max}-2} - \frac{2R\Delta r^* h_w}{\lambda_{er}}(\theta_{i,j_{\max}-1} - \theta_{Coolant}) \quad (13)$$

Note that fictitious variables (associated with subscripts $i = -1$, $i = i_{\max}$, $j = -1$, and $j = j_{\max}$) are used in eqs 8−13, which represent the variable values outside the simulation domain. They are introduced to warrant that the numerical discretization of the boundary conditions is consistent with internal equations. Since the PDEs are extended to the boundary points in discretization, the discrete system (8)−(13) is well posed in the sense that the number of unknown variables equals the number of equations. Using the kinetic model in Figure 2, one can see that five equations must be solved at each grid node, including four mass balance equations (8) for independent species and one energy balance equation (9).

Equations 8−13 form an index-one DAE system which may be written in a more general and compact form as

$$y' = f(t,y,u), g(t,y,u) = 0; \quad y(0) = y^0 \quad (14)$$

where y is a vector of state variables (including temperature and concentration values at interior and boundary grid points), t is time (the independent variable), and u is a vector of control variables (which are the fictitious variables). In principle, the system can be transformed into an ODE system by solving the boundary conditions (10)−(13) for the fictitious variables and substituting into those equations in (8) and (9) involving the fictitious variables. Thus, the algebraic equations and the control variable u (or the fictitious variables in our situation) are eliminated. Compared to ODE systems, DAEs are more convenient to use because they allow direct treatments of boundary conditions. But, it should be pointed out that DAEs are relatively harder to solve, often requiring consistent initial values for the control variables. The reader is referred to, e.g., Ascher and Petzold[8] for more extensive discussion on the numerical solution for DAEs.

For the simulation of fixed-bed reactors, a reactor model involves processes such as convection, diffusion, radiation, and chemical reactions. Each process has its own time scale, drastically different among each other. The resulting system (14) is often stiff in the sense that the eigenvalues of the Jacobian matrix, $\partial f/\partial y = (\partial f_i/\partial y_j)$, vary dramatically in magnitude.[9] Implicit schemes are commonly employed to integrate such stiff systems in order to avoid extremely small steps required by explicit schemes. The popular, general-purposed DAE solvers DASSL[10] and DASPK[3] employ the Backward Differentiation Formula (PDF), a multistep scheme. While here, the one-step multi-stage Runge−Kutta method is considered. An implicit Runge−Kutta scheme provides stability similar to a BDF scheme but does not need another one-step method to start. The Singly Diagonally Implicit Runge−Kutta (SDIRK) method is adopted in this work. SDIRK is an implicit method that can be implemented quite efficiently; e.g., see Alexander[11] and Cash.[12] Our implementation of SDIRK is written in C++, which integrates ODE or index-one DAE systems with the resulting nonlinear algebraic equations being solved using a damped Newton's method. The linear system (with the Jacobian matrix as its coefficient matrix) associated with Newton's iteration is normally large and sparse, which is solved using an iterative scheme with/without preconditioning. While the C++ code provides several choices of iterative schemes, the Conjugate Residual method (CR)[1,2] is used in the computations in this work. In these computations, the linear system iteration and the Newton iteration are stopped when the root-mean-square norm of the difference vector of two consecutive iterates is less than 10^{-7} and 10^{-6}, respectively.

4. Preconditioners

The efficient integration of the system (14) relies on efficient solution of the linear system of algebraic equations resulting from the Newton iteration used to solve the nonlinear equations in the implicit SDIRK discretization. Efficient solution of such a system can often be achieved by using effective preconditioning in conjunction with an iterative scheme. All the preconditioners studied here are constructed based on the incomplete LU (ILU) factorization of a matrix A^*, namely, $P = ILU(A^*)$, which in turn approximated the original Jacobian matrix $\partial f/\partial y = (\partial f_i/\partial y_j)$. The construction involved two levels of approximation: the first approximated the Jacobian matrix and the second approximately factored the approximate Jacobian matrix using ILU factorization. The ILU factorization $ILU(A^*)$ basically maintains the sparsity of matrix A^* and requires $O(n)$ operations to build, where n is the total number of unknown variables (or the length of vector y). To further save computer time, the preconditioner is computed only once at each time step during the transient solution of the reactor model. The same computed preconditioner is used for all the stages of SDIRK during each time step.

4.1. Full Finite Difference (FD) Jacobian Preconditioner (P_1). By applying numerical differentiation to the residual of governing equations (eqs 8−13) with respect to concentration and temperature variables, an approximated Jacobian A_1^* is obtained in a straightforward way. The computation is optimized to the discrete structure of the model equations so that only the nonzero entries in the sparse Jacobian matrix are evaluated and the number of residual evaluations is minimized. The preconditioner P_1 is then calculated by$P_1 = ILU(A_1^*)$. Since A_1^* is actually a numerical equivalent to the Jacobian matrix, one can expect that P_1 is most effective among all four preconditioners, but also the most computationally costly among all.

4.2. Linear Finite Difference (FD) Jacobian Preconditioner (P_2). The construction of the preconditioner P_1 involves calculations of the highly nonlinear reaction terms (eqs 3 and 4), and it has to be built at each time step of simulation. This may significantly increase the computational time of the transient numerical simulation. Here, the idea of approximating the Jacobian matrix using linear terms only is proposed in order to speed up the construction of the preconditioner. For the reaction-transport system (1) and (2), the linear terms involve the time-independent transport part. By numerical differentiation using finite differences, a constant approximate Jacobian matrix A_2^* is generated, which is then used for constructing a constant ILU preconditioner P_2. Note that P_2 needs to be computed only once at the beginning and can be used throughout the entire simulation. By eliminating the need of building the preconditioner at each time step, it is expected that the overall computational time required by transient simulations will be substantially reduced.

4.3. Linear Analytical Jacobian Preconditioner (P_3). Having suppressed the nonlinear reaction terms M and H, it was found that the linear part of the governing equations was simple enough to derive the analytical formula for its Jacobian matrix A_3^*, considered as an approximation to the Jacobian matrix of the full governing equations. We denote $P_3 = ILU(A_3^*)$. The explicit formula of A_3^* is given by He[13] for the fixed-bed reactor model (8)−(13). This reaction-independent analytical Jacobian matrix can be used for simulations of other problems sharing the same linear terms.

4.4. Linear Analytical Jacobian Preconditioner with Finite Difference Reaction Corrections (P_4). Although the preconditioners P_2 and P_3 are computationally cheap to build, they may not be as effective in preconditioning the Jacobian matrix as P_1 since they suppress the effects of the time-dependent, nonlinear chemical reaction terms. Generally speaking, reaction terms can play a significant role in transient reactions, and suppression of their effects may degrade the quality or effectiveness of the preconditioner and thus slow the convergence of iterative solution of linear systems within Newton's iteration. To overcome this problem while maintaining the low cost of constructing the preconditioner, it is possible to add a correction matrix R^* to the linear analytical Jacobian A_3^*, whenever needed, to acquire a better approximation $A_4^* = A_3^* + R^*$ to the Jacobian matrix. (It is noted that the linear FD Jacobian A_2^* can also be used in case A_3^* is unavailable or it is more convenient to use A_2^*.) Once A_4^* is obtained, the preconditioner is computed as $P_4 = ILU(A_4^*)$. For the system (10) and (15), the correction matrix R^* is computed by the numerical differentiation of the reaction terms M and H.

With this approach, economics can be gained in computing the approximate Jacobian matrix. Indeed, as for P_2 and P_3, the constant part (A_2^* or A_3^*) is computed only once and saved. For the correction part R^*, notice that the reaction terms $M_{i,j,k}$ and $H_{i,j}$ at the grid point $(z^*, r^*) = (z_i^*, r_j^*)$ involve only the unknown variables $f_{i,j,k}$ and $\theta_{i,j}$ at the same point. Consequently, R^* is block diagonal and its construction requires only a few evaluations of the reaction terms M and H.

It should be pointed out, though, that every time the correction matrix R^* is added, the ILU factorization of A_4^* should be re-built. The cost can be high if the correction term and the preconditioner are calculated every time step. To further improve the efficiency, they should be recalculated only when necessary. In our computations, the recalculation of R^* and P_4 is triggered by monitoring the iteration number of the iterative linear solver; that is, R^* and P_4 are recalculated if the number of iterations of the linear solver at the current step is 30% greater than that used in the last time step or if Newton's iteration fails. This is reasonable since for a transient reactor simulation the solution converges to a steady state and the number of iterations of the linear solver is supposed to decrease.

5. Case Studies

5.1. Dynamic Case and Near-Steady Case. Transient simulations can be categorized into two cases. The first one is the *dynamic case* where the time derivatives of simulation variables are large, and the variables change quickly with time. In the second case, when the simulation is approaching a steady state, the change of simulation variables is small along the time; it is called *the near-steady case*. The initial guess of Newton's iteration at each time step is chosen as the result from the last time step. In the dynamic case, the initial guess is often poor since the values of the variables may change significantly. Consequently, more computational time is needed since the convergence is slower. Moreover, the approximate Jacobian matrix calculated at the last time step may not be good enough for the current time step, and a recalculation is normally required. This in turn forces the recalculation of the preconditioner, and thus slows down the entire simulation. In contrast, the computation at the near-steady case is relatively easy since the result from the last time step is a good initial guess for Newton's iteration at the current time step and the same Jacobian matrix and preconditioner can be used again at the current (and future) time step without much degradation in the rate of convergence.

5.2. Two Extreme Stages of a Reactor Start-up Process. To evaluate the performance of the four preconditioners described in section 4, two transient simulation test cases are examined: the beginning and ending stages of the start-up of the reactor. At the beginning stage of the reactor start-up, both the concentration and temperature variables (y) and their changes with time (y') evolve quickly from one time step to another; therefore, it can be regarded as the *dynamic case*. The opposite situation is observed when the reactor is operated at the final stage of the start-up, which can be regarded as the *near-steady case*.

The importance of the reaction terms to the calculation of the Jacobian matrix are different in the above two cases. The difference is due to the different temperature and concentration profiles in the reactor at these two transient stages; see Figure 3. For the exothermic reaction system considered, at the beginning stage (time = 5 s.), the temperature is low and therefore the reactions are slow. At the final near-steady-state stage (time = 50 s.), the temperature is high and the reaction rates are also high. However, the reactant is also consumed faster along the reactor and the actual reaction rates are limited by the availability of the reactant.

The significance of the reaction terms to the Jacobian calculation in the aforementioned cases can be explained most clearly by using an example. Consider the reaction network shown in Figure 2, which can be reasonably simplified to a single reaction: $A \rightarrow B$,[14] where A is *o*-xylene and B is phthalic anhydride. With this single step, the generation terms M and H in governing equations (1) and (2) can be written as

$$M = \rho_s(1 - \epsilon)kP_{oxy}P_A$$
$$H = \rho_s(1 - \epsilon)(-\Delta H_j kP_{oxy}P_A) \tag{15}$$

where P_{oxy} is the partial pressure of oxygen, set at 0.21 atm as constant, $P_A = C_ART$, heat of reaction $\Delta H = -308269$ cal/mol, rate constant $k = \exp(11.597 - 13636/T)$ mol/g$_{cat}$-atm^2-s. Rewrite the governing equations (1) and (2) into the form $F = F_{diffusion+convection} + F_{reaction}$, in which $F_{reaction} = [M, H]$, and the transport part $F_{diffusion+convection}$ is given in (1) and (2). The Jacobian matrix used for the Newton's iteration of the SDIRK method is therefore

$$A_4^* = \frac{\partial F}{\partial y'\partial t} + \frac{\partial F}{\partial y} = \left[\frac{\partial F_{diffusion+convection}}{\partial y'\partial t} + \frac{\partial F_{diffusion+convection}}{\partial y}\right] + \left[\frac{\partial F_{reaction}}{\partial y'\partial t} + \frac{\partial F_{reaction}}{\partial y}\right] = A_3^* + R^* \tag{16}$$

in which $y = [C_A, T]$ and $y' = [C_A', T']$. Here the Jacobian matrix A_4^* consists of the summation of the transport contribution and reaction contribution. It is a block pentadiagonal matrix with each block of size 2 × 2. Again note the reaction terms appear only at the main block diagonal of the Jacobian matrix. For node (i,i) of A_4^*, the reaction contribution terms are given by

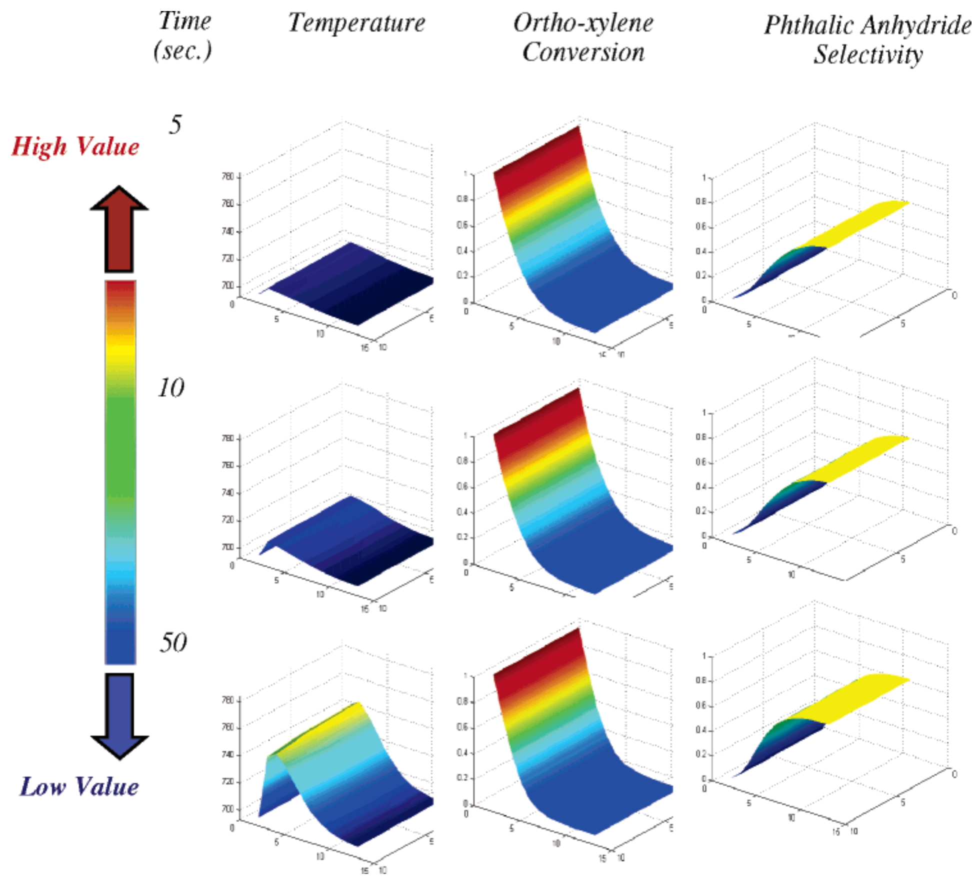

Figure 3. The transient reactor profiles for the first 50 s of the reactor start-up. Images shown are simulated results over a 103 × 2.54 cm (axial × radial distance) axial symmetric domain of a cylindrical reactor (the axial distance covers the first 1/3 length of the entire reactor) at different time stages. The images were rotated 37.5° clockwise. The left edge of each image is the reactor inlet, and the image edge toward the reader is the wall of the reactor.

$$R^*_{i,i} = \begin{bmatrix} R^*_{11} & R^*_{12} \\ R^*_{21} & R^*_{22} \end{bmatrix}_{(i,i)} = \tag{17}$$

$$\begin{bmatrix} \rho_s(1-\epsilon)\cdot e^{(11.597)-(13636/T)}P_{oxy}RT & \rho_s(1-\epsilon)P_{oxy}\cdot C_A R\left(1+\dfrac{13636}{T}\right)\cdot e^{(11.597-(13636/T))} \\ \rho_s(1-\epsilon)\cdot e^{(11.597-(13636/T))}\cdot(-\Delta H)P_{oxy}RT & \rho_s(1-\epsilon)(-\Delta H)\cdot P_{oxy}C_A R\left(1+\dfrac{13636}{T}\right)\cdot e^{(11.597-(13636/T))} \end{bmatrix}_{(i,i)}$$

Assume a constant inlet concentration that is given as $C_A = 0.176$ mol/m^3. Using the values given in the first paragraph of section 6, it can be calculated that at $T = 693$ K, $R^*_{11} = 3.43$, $R^*_{12} = 0.018$, $R^*_{21} = 1.06 \times 10^6$, and $R^*_{22} = 5.55 \times 10^3$, while at $T = 793$ K, $R^*_{11} = 46.93$, $R^*_{12} = 0.189$, $R^*_{21} = 1.45 \times 10^7$, and $R^*_{22} = 5.82 \times 10^4$. On the other hand, the diagonal transport contribution terms are at $O(10^7)$ at both temperatures (for a simulation with 33 × 11 nodes and $dt = 0.001$ s.). Therefore, the contribution to the Jacobian calculation from one of the reaction terms, R^*_{21}, is significant, and its significance even increases with the temperature. However, neglecting reaction terms only affects the approximation of the Jacobian matrix at about one-fourth of diagonal positions, and only when the reactor is operated at a high temperature. In addition, in the reality of the transient simulation, the concentration of the reactant (C_A) decreases quickly with time and along the reactor. From Figure 3, it is observed that, at 50 s, the concentration of the reactant *o*-xylene is nonzero only at the inlet part of the reactor, although the overall reactor temperature is high. Therefore, two other terms R^*_{12} and R^*_{22} are nonzero only at the inlet part of the reactor. In summary, the overall significance of the reaction terms to the Jacobian calculation varies with operating conditions, intrinsic characteristics of the reactor, simulation time, and spatial resolution and may not be as large as thought.

5.3. The Effect of Different Step Sizes. In transient simulations, the size of the time step also affects the computation speed. For a fixed time length ($t_{\text{final}} - t_{\text{start}}$) simulation, the use of a smaller time step (which requires more time steps to complete the simulation) is more likely to slow the simulation. This is because more computation is required overall, although Newton's iteration converges relatively faster. Nevertheless, a small time step size is still desired in certain cases, e.g., to observe a transition process closely, or to overcome the numerical problems caused by the stiffness of the model.

Before the reactor reaches a steady state in the simulation, the concentration and temperature values vary significantly over a large time step. Numerically, a large step size transient simulation is equivalent to the dynamic case while a small step size transient simulation is equivalent to the near-steady case.

6. Numerical Results and Discussion

The transient pseudo-homogeneous fixed-bed reactor model is discretized using the MOL approach over the spatial domain (eqs 8−13), and the resulting DAE system is integrated using the SDIRK scheme and the Conjugate Residual (CR) iterative linear solver. Kinetic parameters include the activation energies and pre-exponential terms that were estimated using a quasi-Newton's method with the steady-state experimental data of the air oxidation of *o*-xylene to phthalic anhydride by Calderbank et al.[7] Unless otherwise stated, numerical experiments are conducted under the following conditions: *reactor tube radius* = 2.54 cm, *reactor length* = 3 m, $P_{\text{reactor}} = 1$ atm, $\epsilon = 0.56$, $C_{p,f} = 1.086$ J/m/g/K, $C_{p,s} = 0.95$ J/m/g/K, and $\rho_s = 2097$ kg/m^3. For a reactor start-up process, the preheated reactor and the coolant are at a temperature of 693 K. The feed conditions are as follows: temperature 693 K, 1% *o*-xylene (molar ratio) or $C_{o-\text{xylene}} = 0.176$ mol/m^3 and $G = 4746$ kg/m^2/h.

The following discussion is based on the results of reactor start-up simulations at three different mesh resolutions of the two-dimensional spatial domain (axial and radial directions): 33 × 9 (totally 33 × 9 × 5 = 1485 equations); 66 × 18 (totally 66 × 18 × 5 = 5940 equations); and 132 × 36 (totally 132 × 36 × 5 = 23760 equations). Two fixed time step sizes are used, $dt = 0.02$ and $dt = 0.2$ s. In the figures, the complete FD Jacobian preconditioner (P_1) is called "FD Jacobian", the linear FD Jacobian preconditioner (P_2) is called "FD Linear", the linear analytical Jacobian preconditioner (P_3) is called "Analytical Linear", and the linear analytical Jacobian preconditioner (P_4) with FD reaction correction is called "Analytical Linear w/FD Reaction Correction". The case of transient simulation for the first 10 s of reactor start-up is regarded as the initial stage of the transient simulation and is denoted as "Initial" while the case of transient simulation for the last 10 s of reactor start-up, when the reactor is approaching its steady state, is denoted by "S.S."

6.1. Effects of Fill-in Levels on Performance of ILU Preconditioners. The comparison between the computational performances of preconditioning using different levels of fill-in is shown in Figure 4. To demonstrate the overall improvement in the efficiency of the computation using the ILU preconditioning technique, Figure 4a also contains the results for the same simulation without using any preconditioning. It is evident that the improvement is dramatic.

By construction, a higher fill-in level of an ILU preconditioner leads to a better ILU factorization to the approximate Jacobian matrix because fewer elements are discarded. At the initial stage of a transient simulation with a large time step, the reactor concentration and temperature profiles vary significantly over a single time step. In this case, Newton's iteration requires less computational time with a higher fill-in level ILU preconditioner since the corresponding linear system can be solved faster with a better preconditioner. However, a level two fill-in preconditioner is more expensive to compute than a level one fill-in preconditioner. When an accurate preconditioner is not necessary, using a level 2 fill-in preconditioner can actually slow the computation. For example, for the case of the transient simulation with small time steps ($dt = 0.02$), which is the numerically equivalent near-steady case, the Newton's iteration converges very fast since the values of the variables do not change significantly over each time step. In this case, the savings in solving the corresponding linear system with a high-quality preconditioner only accounts for a very small percentage of the total computational time. Indeed, the level one fill-in ILU preconditioner outperforms the level two fill-in ILU preconditioner (Figure 4, $dt = 0.02$). On the other hand, it can be seen from Figure 4b that when a larger step size is used ($dt = 0.2$, the numerically dynamic case), the better quality, level two fill-in ILU preconditioner outperforms the level one preconditioner. In this case, Newton's method takes more iteration to converge and the efficiency in solving the underlying linear system is important. Generally speaking, level one fill-in ILU preconditioners are recommended for small step size transient simulations while level two (and higher) fill-in ones are suggested for large time step size simulations. But how big should a time step be regarded as large? For the fixed-bed reactor start-up test cases in this study, our numerical experiments show that, at the initial stage of the transient simulation, a step size greater than 0.05 can be regarded as "large". However, when the reactor is approaching steady state, a "large" step size must be any value greater than 0.15. A quantitative definition of small and large step size for general cases is always difficult, and it is more realistic to make the decision with test simulations.

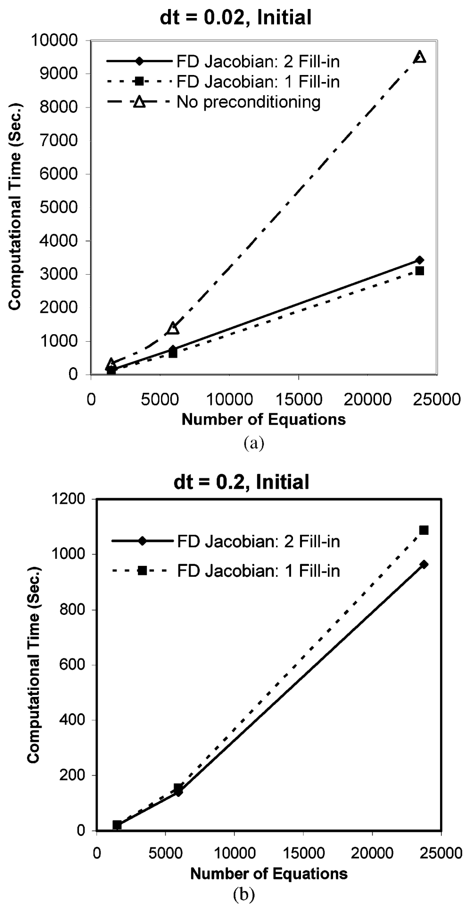

Figure 4. The computational time for simulating the initial 10 s of the reactor start-up; the complete finite difference Jacobian preconditioner (P_1) is used with level 1 and 2 fill-in ILU.

6.2. Performance of the Preconditioners. The performance of the four preconditioners described in section 4 is compared for various simulations. For comparison purposes, all of the preconditioners are constructed using level 2 fill-ins. For all cases (Figures

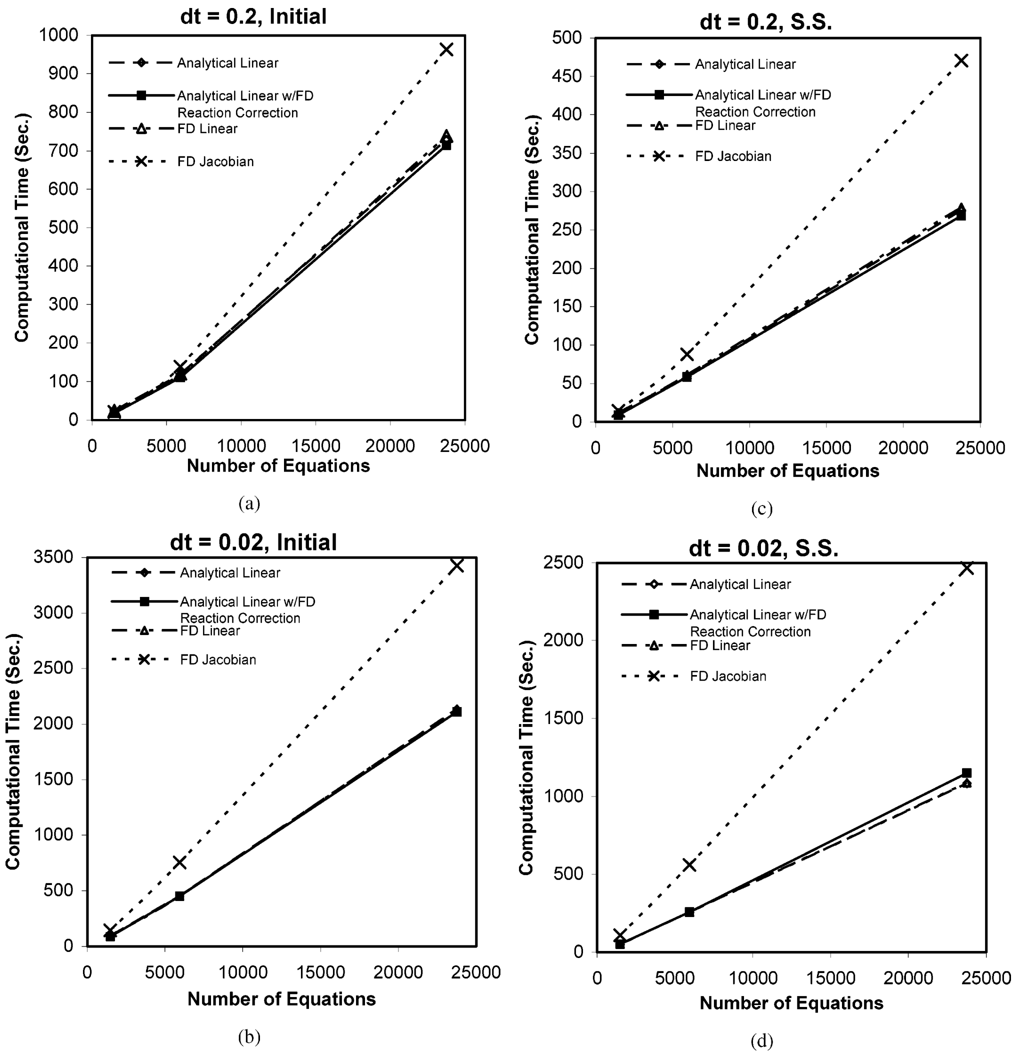

Figure 5. The computational time for simulating 10 s at the initial and steady-state stages of the reactor start-up with $dt = 0.2$ or 0.02 s. All preconditioners use level 2 fill-in.

5a–d), it is found that the simulations using the complete FD Jacobian preconditioner (P_1) are consistently the slowest, and therefore P_1 is excluded from the following discussion.

The first set of results is acquired using $dt = 0.2$ at the initial stage of the reactor start-up example (Figure 5a). This is a "dynamic + dynamic" case (large time step size + large variation of the variables over each time step), where the quality of the preconditioners is important to the efficiency of the transient simulation. It was expected that the performance of the Analytical Linear w/FD Reaction Correction preconditioner (P_4) should be better than the other less accurate ones. Interestingly, the results show that it performs only marginally better than P_2 and P_3, which are calculated based on the linear, time-independent part of the governing equations and have the advantage of extremely low construction cost.

The second set of results is obtained by using a smaller time step size ($dt = 0.02$) at the initial stage (Figure 5b), which is a "near-steady + dynamic" case (small time step size + large variation of the variables over each time step). The third set of results is computed by using a larger time step size ($dt = 0.2$) at the nearly steady-state stage (Figure 5c), which is a "dynamic + near-steady" case. In these two cases, the quality of the preconditioners and the cost of calculating the Jacobian matrix and the preconditioners are both important to the overall efficiency of the computation. Recall that, for our transient simulation test cases, the reaction terms are not significant to the approximation of the Jacobian matrix. It is evident that the three preconditioners (P_2, P_3, and P_4) perform equally well. The Analytical Linear w/FD Reaction Correction preconditioner (P_4) again performs slightly better than P_2 and P_3, while the performance of the latter two is barely distinguishable.

The last set of results is acquired using a small step size ($dt = 0.02$) at the nearly-steady-state stage of the reactor start-up example (Figure 5d). This is a "near-steady + near-steady" case, where Newton's method converges fast at each time step. The cost of calculating the Jacobian matrix and the preconditioner dominates the overall computational time of the transient simulation. Recall that the cost of constructing the preconditioners is on the order of Analytical Linear (P_3) < FD Linear (P_2) < Analytical Linear w/FD Reaction Correction (P_4) < FD Jacobian (P_1). It can be seen that the computational time required for the simulations using these preconditioners basically is in the same order.

The above results show that the preconditioners P_2 and P_3 based on the linear part of the governing equations are easy to construct and have extremely low construction cost while being effective in improving the

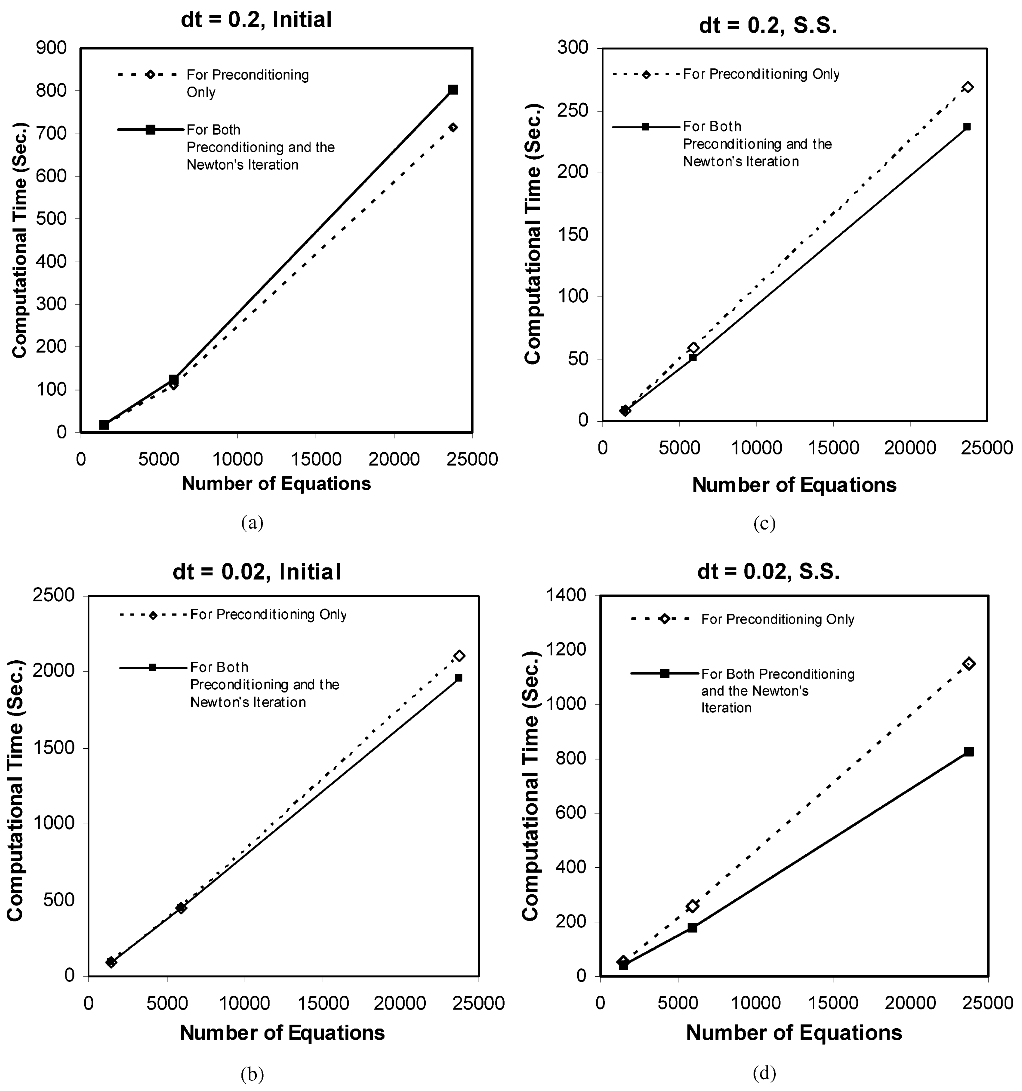

Figure 6. The performance of the linear-approximated analytical Jacobian with reaction corrections (P_4) as used for either preconditioning only or for both preconditioning and Newton's iteration. (10 s of simulations during the initial and the steady-state stage of the reactor start-up; $dt = 0.02$ and 0.2; level 2 fill-in ILU.)

efficiency of the transient simulation in our test examples. On the other hand, for the general reactor simulations where the reaction terms can be very important to the Jacobian matrix approximation, the Analytical Linear w/FD Reaction Correction preconditioner (P_4) is recommended. P_4 takes into account the reaction terms and therefore performs better than or at least comparable to P_2 and P_3.

6.3. Using Linear-Approximated Jacobian with Reaction Corrections in both Preconditioning and Newton's Method. Since P_4 is an easily computable, good approximation to the actual Jacobian A in Newton's method, performance improvement is expected in certain cases when it is used as the Jacobian in Newton's method. On the other hand, Newton's iteration with P_4 takes longer to converge since P_4 is not the exact Jacobian. The question is: under what condition which of the competing factors is dominant. Here the speed of a transient simulation is compared using the actual Jacobian A against using the preconditioner P_4 as the Jacobian in Newton's method. In either case, the preconditioner P_4 is used for preconditioning. Results are shown in Figure 6.

It can be observed that the use of P_4 for both preconditioning and Newton's iteration shows improvements in computation time in all cases except for the "dynamic + dynamic" one (Figure 6a), where the convergence speed of Newton's iteration is most sensitive to the accuracy of the Jacobian matrix calculation. Since, for stable problems, the "dynamic + dynamic" case is normally just a small fraction in the entire duration of a transient simulation (e.g., the reactor start-up process considered), it can be concluded that using the preconditioner P_4 for both preconditioning and as the iteration Jacobian matrix for Newton's method is able to enhance the overall efficiency of transient reactor simulations, especially when high temporal and spatial resolutions are needed.

7. Conclusions

This study investigated the construction and efficiency of time-independent preconditioners based on the linear part of the governing equations of the reaction-transport systems. The idea is to use only the transport part of the governing equations to calculate the preconditioner. In our implementation, instead of applying the incomplete LU (ILU) factorization on the actual Jacobian (which gives preconditioner P_1), two new preconditioners are constructed by applying ILU on linear-approximated Jacobians: P_2 uses an approximated Jacobian by numerical differentiation and P_3 uses an approximated Jacobian with analytical formulas. Since the transport properties are commonly assumed constant in transient numerical simulations, the resulting preconditioners are constant matrixes, which need to be calculated only once, at the beginning of the simulation. Therefore, this approach essentially eliminated the preconditioner computation portion from the overall cost of the transient simulation. Upon this basic idea, preconditioner P_4 was proposed, which is a variation of the linear-approximated preconditioner P_3 achieved by adding the previously ignored chemical reaction contributions. To minimize the computation required by calculating P_4, the reaction correction terms are computed only at a few time steps during the entire transient simulation, when the convergence of the underlying sparse linear solver becomes slower at the current time step than at the previous step.

By performing transient simulations in both dynamic (the variation of the variables is large over each time step) and near-steady cases (the variation of the variables is small over each time step), it was demonstrated that the level 2 fill-in ILU preconditioner performs better in dynamic cases, while the level 1 fill-in ILU preconditioner performs better in near-steady cases. It was also shown that preconditioners P_2, P_3, and P_4 consistently outperform the standard complete ILU preconditioner P_1, due to their negligible computational cost in the overall computation. It was found that, for our transient reactor simulation examples, neglecting chemical reaction terms did not significantly affect the quality of the preconditioners; therefore, P_2, P_3, and P_4 performed similarly in all cases, although P_4 is slightly better than the others. However, P_4 is recommended for general transient reactor simulations since the chemical reactions are normally important. P_4 not only considers chemical reactions but also has all the advantages of P_2 and P_3 (easy to construct and compute), due to its adjustable updating frequency of the reaction terms.

It was also demonstrated in the study that using the preconditioner P_4 simultaneously for preconditioning and as the Jacobian of Newton's iteration may further speed up certain transient reactor simulations. One should be aware that, in strongly dynamic cases when the variation of the variables is significant over time steps, the use of any approximated matrix to replace the actual Jacobian in Newton's iteration could slow the overall simulation. However, if such a case is only a small fraction of the scope of the transient simulation, then it has been shown that using P_4 for both preconditioning and Newton's iteration significantly increases the overall speed of transient reactor simulations at high resolutions.

Acknowledgment

This work was supported in part by the National Science Foundation (PACI) and the National Computational Science Alliance (UNI 12581).

Nomenclature

A_j = activation energy for reaction step j
A^* = Jacobian matrix, calculated
C_k = concentration of species k, mol/m^3
C_p = heat capacity of the reactor, J/g/K
C_{pf} = heat capacity of the fluid, J/g/K
D_e = effective diffusivity, m^2/s
D_t = diameter of the reactor tube, m
f = dimensionless concentration, $f_k = c_k/c_{ox,0}$
$F_{i,j,k}$ = mass transport residual function for species k at 2D grid node (i,j)
$G_{i,j}$ = heat transport residual function at 2D grid node (i,j)
G = flow rate, g/m^2/s
L = length of the reactor, m
h_w = wall heat transfer coefficient, J/s/m^2/k
H = energy generation term, J/m^3/s
M_k = mass generation term for species k, mol/m^3/s
N = total number of reaction steps
N_k = total number of reaction steps that involve species k
P = total pressure, atm
P = preconditioning matrix
r = distances in radial direction, m
R = radius, m
R = gas law constant, 8.3145 J/mol/K
R^* = partial Jacobian matrix that contains reaction terms only
r_j = surface reaction rate of step j, mol/g/s
T = temperature, K
T_c = coolant temperature, K
t = time, s
u = velocity in axial direction, m/s
$w_{k,j}$ = stoichiometric constant for reaction step j to species k
y = vector of concentration and temperature variables
z = distances in axial direction, m

Greek Symbols

ΔH_j = heat of reaction of step j, cal/mol
ϵ = porosity of the bed
ρ = density, g/m^3
θ = dimensionless temperature, $\theta = T/T_0$
λ_e = effective heat conductivity, J/s/m/K

Superscripts

$*$ = dimensionless length
$*$ = approximated matrix
0 = initial condition

Subscripts

0 = feed
A = *o*-xylene
f = fluid phase
k = partial value for species k
oxy = oxygen
p = pellet
r = radial direction
s = solid−fluid interface
z = axial direction
w = reactor tube wall

Literature Cited

(1) Greenbaum, A. *Iterative Methods for Solving Linear Systems*; Society for Industrial and Applied Mathematics: Philadelphia, 1997.

(2) Gutknecht, M. H. Variants of BICGSTAB for Matrices with Complex Spectrum. *SIAM J. Sci. Comput.* **1993**, *14*, 1020.

(3) Brown, P. N.; Hindmarsh, A. C.; Petzold, L. R. Using Krylov methods in the solution of large-scale differential-algebraic systems. *SIAM J. Sci. Comput.* **1994**, *15*, 1467.

(4) Langtangen, H. P. *Computational Partial Differential Equations: Numerical Methods and Diffpack Programming*; Springer: Berlin; New York, 1999.

(5) Lee, D. *Local Preconditioning of the Euler and Navier−Stokes Equations.*. Ph.D. Dissertation, The University of Michigan, 1996.

(6) Bird, R. B.; Stewart, W. E.; Lightfoot, E. N. *Transport Phenomena*; Wiley: New York, 1960.

(7) Calderbank, P. H.; Chandrasekhharan, K; Fumagalli, C. The Prediction of the Performance of Packed-Bed Catalytic Reactors in the Air-Oxidation of O-xylene. *Chem. Eng. Sci.* **1977**, *32*, 1435.

(8) Ascher, U. M.; Petzold, L. R. *Computer methods for Ordinary Differential Equations and Differential-Algebraic Equations*; Society for Industrial and Applied Mathematics: Philadelphia, 1998.

(9) Ebert, K. H., Deuflhard, P., Jäger, W., Eds. *Modelling of Chemical Reaction Systems*; Springer Series in Chemical Physics 18; Springer-Verlag: Berlin, 1981.

(10) Petzold, L. R. *A Description of DASSL: A Differential/algebraic System Solver*; Scientific Computing: North-Holland, Amsterdam, 1983.

(11) Alexander, R. Diagonally implicit Runge−Kutta methods for stiff O.D.E.'s. *SIAM J. Numer. Anal.* **1977**, *14*, 1006.

(12) Cash, J. R. Diagonally implicit Runge−Kutta formulae with error estimates. *J. Inst. Math. Appl.* **1979**, *24*, 293.

(13) He, Y. *Simulation Studies on a Non-isothermal, Non-adiabatic, Fixed-bed Reactor*. Ph.D. Dissertation, The University of Kansas, 2003.

(14) Varma, A.; Morbidelli, M.; Wu, H. *Parametric Sensitivity in Chemical Systems*; Cambridge University Press: New York, 1999.

Received for review September 28, 2004
Revised manuscript received April 12, 2005
Accepted April 25, 2005

IE049052D