Journal of Scientific Computing, Vol. 4, No. 4, 1989

A Method for Exponential Propagation of Large Systems of Stiff Nonlinear Differential Equations[1]

Richard A. Friesner,[2] Laurette S. Tuckerman,[3,4] Bright C. Dornblaser,[4,5] and Thomas V. Russo[4,5]

Received November 20, 1989

A new time integrator for large, stiff systems of linear and nonlinear coupled differential equations is described. For linear systems, the method consists of forming a small (5–15-term) Krylov space using the Jacobian of the system and carrying out exact exponential propagation within this space. Nonlinear corrections are incorporated via a convolution integral formalism; the integral is evaluated via approximate Krylov methods as well. Gains in efficiency ranging from factors of 2 to 30 are demonstrated for several test problems as compared to a forward Euler scheme and to the integration package LSODE.

KEY WORDS: Stiff equations; Krylov methods; exponential propagation.

1. INTRODUCTION

The construction of an efficient algorithm for the solution of large, coupled sets of stiff ordinary differential equations has been a central concern of numerical analysis for many years (Byrne and Hindmarsh, 1987). At present, the most generally applicable and widely used methods are based on backward differentiation formulas (Gear, 1971; Hindmarsh, 1983). Stiff equation packages of this type also employ a complex control structure in which a variable time step and order are chosen automatically to satisfy user-specified error tolerances.

[1] Presented at the Third Nobeyama Workshop on Supercomputers and Experiments in Fluid Mechanics, September 1989.
[2] Department of Chemistry, University of Texas at Austin, Austin Texas 78712.
[3] Department of Mathematics, University of Texas at Austin, Austin Texas 78712.
[4] Center for Nonlinear Dynamics, University of Texas at Austin, Austin Texas 78712.
[5] Department of Physics, University of Texas at Austin, Austin Texas 78712.

In this paper, we propose a new algorithm for the solution of general stiff equation systems which potentially appears to offer significant improvement for a wide class of problems. We first describe the algorithm in some detail and indicate where gains in efficiency can be expected to arise. For linear systems, the advantages are easily understood and demonstrated. Applications to several types of standard nonlinear systems are then presented. For some problems, a reduction in CPU time of more than a factor of 10 is obtained for several levels of solution accuracy.

The algorithm is based upon an expansion of the solution in a Krylov subspace of the linearized operator, created by repeated action of the Jacobian on a trial vector. For a symmetric matrix, this corresponds to a Lanczos procedure. For nonsymmetric matrices, one simply forms the operator in an orthogonalized Krylov space, as in the work of Arnoldi (1951). As a relatively small (5–15) number of iterations are used per time step, neither orthogonalization nor diagonalization of the resulting reduced matrix constitute a prohibitively expensive portion of the calculation, so long as the number of equations is sufficiently large. Such an approach has been utilized by a number of workers to find eigenvectors of large linear systems (Saad, 1980; Goldhirsch, Orszag, and Maulik, 1987; Christodoulou and Scriven, 1988).

Krylov methods have also found application in the solution of differential equations. Gear and Saad (1983), and Brown and Hindmarsh (1986, 1989) have used a Krylov method to solve the large linear systems resulting from the backwards differentiation formulas. Park and Light (1986) and Leforestier *et al.* (1989) have achieved high efficiencies by using the Krylov algorithm to exponentially propagate the Schrödinger equation via approximate diagonalization. However, the use of exponential propagation to solve systems of nonlinear differential equations is, as far as we know, novel.

Propagation of a linear system is achieved by straightforward exponentiation of the Krylov matrix, followed by transformation back to the original vector space. For nonlinear systems, we correct the linearized results using a convolution integral. This requires solution of a nonlinear equation for the correction term, the most demanding aspect of which is evaluation of an integral over time (the independent variable) of the exponentiated linearized operator acting on the nonlinear terms. The integral is evaluated by expanding the nonlinear correction in polynomials in time, forming a Krylov subspace (of very low dimension) for the vector coefficient of each polynomial, and then analytically integrating the polynomial over the resulting exponentials. Exponentiation allows the use of very long time steps in many cases, for both the linear and nonlinear parts of the problem.

The availability of an analytic expression in terms of exponentials and powers of t for the solution also facilitates error analysis, as derivatives can be explicitly computed analytically. This leads to a method of selecting an appropriate time step. Results at intermediate times, as well as the approximate eigenvectors and eigenvalues are also available from the Krylov space expansions at a negligible cost.

The present article does not constitute a demonstration that the algorithm will run reliably and efficiently on a diverse set of problems; to show this, many more examples will be required. Rather, it is intended to illustrate the possibilities inherent in the approach. This has been done by achieving large reductions in computational effort for a few systems, substituting empirical optimization for an as–yet–incomplete internal control structure.

The article is organized as follows. In Section 2, the linear version of the algorithm is described. Section 3 outlines the nonlinear correction scheme. Section 4 discusses practical implementation, including error analysis, choice of time step, and selection of various other parameters of the algorithm. Sections 5 and 6 present results for the Krogh test problem, and for a reaction–diffusion system, with computations carried out on a Cray X-MP supercomputer. Section 7, the conclusion, suggests possible improvements of the algorithm that can be expected for the future.

2. LINEAR KRYLOV ALGORITHM

We consider a linear system of the form

$$\frac{dx}{dt} = Ax \tag{2.1a}$$

$$x(t_0) = x_0 \tag{2.1b}$$

where x is an N-dimensional vector and A is a real $N \times N$ matrix. The exact solution to this system is

$$x(t_0 + t) = e^{At}x_0 \tag{2.2}$$

The basic Krylov algorithm for propagation of x from t_0 to a later time $t_0 + t$ is as follows:

(1) We define $\psi^1 = x_0/\|x_0\|$. We form ψ^2 by computing $A\psi^1$, orthogonalizing to ψ^1, and normalizing. Simililarly, ψ^k is formed by computing $A\psi^{k-1}$, orthogonalizing to all previous ψ vectors, and normalizing. K iterations thus yields K Krylov vectors, which form a basis for the propagation. K will typically be on the order of 3–15. The K-dimensional

linear space spanned by the ψ^k is called the Krylov subspace. We can define an $N \times K$ matrix Ψ by $\Psi_{ik} \equiv \psi^k_i$. Ψ transforms from the Krylov subspace to the original space, i.e., multiplication by Ψ of a K-vector evaluates the linear combination of Krylov vectors whose coefficients are given by that K-vector; conversely $\Psi^\dagger$ can be viewed as a projection operator from R^N into the Krylov subspace.

(2) In the course of orthonormalizing the vectors ψ_k, the Krylov matrix $\tilde{A}$ defined by $\tilde{A}_{kl} \equiv \langle \psi^k | A | \psi^l \rangle$, i.e.,

$$\tilde{A} \equiv \Psi^\dagger A \Psi \tag{2.3}$$

is assembled, as described by Saad (1980). The $K \times K$ upper Hessenberg matrix $\tilde{A}$ is diagonalized using the QR algorithm, which is inexpensive for small K. That is,

$$\tilde{A} = U \Lambda U^{-1} \tag{2.4}$$

where U is the matrix of eigenvectors, and Λ the diagonal matrix of eigenvalues of $\tilde{A}$.

(3) The formula for $x(t_0 + t)$ is then

$$x(t_0 + t) = \Psi U e^{\Lambda t} U^{-1} \Psi^\dagger x_0$$

The products ΨU and $U^{-1} \Psi^\dagger$ transform between R^N and what can be called the diagonalized Krylov representation, in which the action of e^{At} is given by the K by K diagonal matrix $e^{\Lambda t}$. Note that by construction, $\Psi^\dagger x_0 = \Psi^\dagger \psi^1 |x_0| = e_1 |x_0|$ (where e_1 is the first unit vector); however, we retain the $\Psi^\dagger$ notation for clarity.

The difference between exponential propagation and the usual multistep methods can be summarized for the linear equation (2.1) as follows. Multistep methods approximate the exponential of A occurring in solution (2.2). Explicit methods use a polynomial approximation, whereas implicit methods (including the backwards differentiation formulas) employ rational approximations, which are more stable because they are bounded in large regions of the left half complex plane (Gear, 1971), but which require matrix inversion. Within the rational or polynomial approximations, however, the exact matrix A is used. In contrast, exponential propagation performs an *exact exponential*, but of the *approximate matrix* $\tilde{A}$ of (2.3). Exponentiation, while desirable, is far more expensive than inversion of a matrix of the same size. We have targeted our integration method to large systems because it is only then that the cost of exponentiation of a small approximate matrix can compete with that of inversion of a matrix of much larger size.

We now state several elementary properties of the Krylov propagation algorithm. First, if x_0 is composed of K or fewer eigenvectors of A, propagation is exact for all time. Secondly, it can be shown that accuracy to order t^K is achieved if one carries out K iterations. Finally, the method is clearly globally stable, as large negative eigenvalues are simply exponentiated and their components thus rapidly destroyed. This last property is what suggests the method as a useful stiff equation solver: one does not need to propagate these large negative eigenvectors accurately in order to achieve stability. The time step can therefore be chosen in accord with how well the eigenvectors on the (long) time scale of primary interest are advanced.

The convergence with iteration number of the eigenvalues and eigenvectors of a matrix for Lanczos-type algorithms is quite complicated, even in the case where the matrix is symmetric. We refer the reader to Cullum and Willoughby (1985) and to Saad (1980) for a detailed discussion. Here, in any case, the issues are rather different than in the usual employment of Lanczos, in that accurate time evolution for a restricted interval does not require accurate convergence of the eigenvectors.

The above considerations render a formal analytical error analysis of the method rather difficult. As the methodology is developed further, such an analysis may be worth pursuing despite the difficulties. The objectives of the present paper, however, do not require such work; our intention is to see whether or not the intuitive advantages of an algorithm based upon exponentiation in a restricted subspace can be translated into actual improvements in CPU time for an interesting and computationally demanding problem.

3. NONLINEAR ALGORITHM

3.1. Formal Equations

We now consider a general set of coupled autonomous nonlinear ODEs of the form

$$\frac{dx}{dt} = f(x) \tag{3.1a}$$

$$x(t_0) = x_0 \tag{3.1b}$$

Our first step is to linearize the evolution operator about the current state x_0:

$$\frac{d}{dt}(x - x_0) = f(x_0) + Df(x_0)(x - x_0) + [f(x) - f(x_0) - Df(x_0)(x - x_0)]$$

We then obtain the following equation for $y(t) \equiv x(t_0 + t) - x_0$:

$$\frac{d}{dt} y = b + Ay + R(y) \tag{3.2a}$$

$$y(0) = 0 \tag{3.2b}$$

where $b \equiv f(x_0)$, $A \equiv Df(x_0)$, and the remainder

$$R(y) \equiv f(x_0 + y) - f(x_0) - Df(x_0)\, y \tag{3.3}$$

is $O(y^2)$. The right-hand side of Eq. (3.2a) can be divided into two parts: a linear part, which can be solved with the algorithm of Section 2, and the remainder $R(y(t))$ which constitutes a time-dependent forcing term to be determined self-consistently. Our initial guess consists of setting the forcing term equal to zero, leading to the linear equation

$$\frac{d}{dt} y^{(0)} = b + Ay^{(0)} \tag{3.4a}$$

$$y^{(0)}(0) = 0 \tag{3.4b}$$

for the first solution $y^{(0)}$ [note the presence of the constant term b, modifying the solution from that of Eq. (2.1a)]. This leads to new approximation of $R(y)$ as $R(y^{(0)})$. The next approximation $y^{(1)}$ is the solution to

$$\frac{d}{dt} y^{(1)} = b + Ay^{(1)} + R(y^{(0)}) \tag{3.5a}$$

$$y^{(1)}(0) = 0 \tag{3.5b}$$

Techniques for evaluating $R(y)$ and for solving (3.5a) and (3.5b) are described below. Given that this can be done, a sequence of solutions $y^{(m)}$ can be generated until the desired degree of accuracy is achieved.

3.2. Solution of Linear Equation with Constant Term

The exact solution of Eqs. (3.4a) and (3.4b) is

$$y^{(0)}(t) = \frac{e^{At} - I}{A} b \tag{3.6}$$

We approximate this within the Krylov space via the expression

$$y^{(0)}(t) = \Psi U \frac{e^{\Lambda t} - I}{\Lambda} U^{-1} \Psi^{\dagger} b \tag{3.7}$$

Here, the Krylov space is formed by acting successively with A on b, i.e., $\psi^1 \equiv b/\|b\|$, and ψ^k is the result of orthonormalizing $A\psi^{k-1}$ to all previous ψ's. As in (2.3-4), $\Psi^\dagger A \Psi = U \Lambda U^{-1}$. The notation $(e^{\Lambda t} - I)/\Lambda$ serves as shorthand for diag $[(e^{\lambda_k t} - 1)/\lambda_k]$. For small values of $\lambda_k t$, this expression must be evaluated via a truncated Taylor series to avoid loss of accuracy.

Like many other iterative methods, the Krylov algorithm is especially advantageous when acting with A (the Jacobian of f) is inexpensive relative to other operations, such as inversion. At present, the user must write a routine to do this, although it is possible to use a finite difference approximation as do, for example, Brown and Hindmarsh (1986).

3.3. Evaluation of the Convolution Integral

The closed-form solution (3.6) for the linear problem is no longer available in solving equation (3.2). However, it now appears as part of an integral equation equivalent to (3.2):

$$y(t) = \frac{e^{At} - I}{A} b + \int_0^t e^{A(t-\tau)} R(y(\tau))\, d\tau \tag{3.8}$$

Our iteration procedure for solving (3.2) can be written formally as follows:

$$y^{(m)}(t) = \frac{e^{At} - I}{A} b + \int_0^t e^{A(t-\tau)} R(y^{(m-1)}(\tau))\, d\tau \tag{3.9}$$

using (3.6) as an initial guess.

The new problem here is the evaluation of the integral on the right-hand side of Eq. (3.9). Our strategy is to approximate the nonlinear functional $R(y^{(m-1)}(\tau))$ by an expansion of the form

$$R(y(\tau)) = \sum_{j=1}^{P} c_j \phi_j(\tau)$$

where the ϕ_j are smooth analytic functions of time and the c_j are vector coefficients. The formulation is quite general and any set of functions can be employed. In our present implementation, we utilize polynomials. Equations (3.2b) and (3.3) guarantee that both R and its time derivative vanish at $t = 0$. Thus $R(y(\tau))$ contains no constant or linear term in time and the polynomials used in the fitting procedure begin at second order, i.e., $\phi_j(\tau) = \tau^{j+1}$.

The coefficients c_j are obtained via collocation. We evaluate R on a grid of points τ_p, $p = 1, \ldots, P$, and solve the collocation equations

$$R(y(\tau_p)) = \sum_{j=1}^{P} c_j \phi_j(\tau_p) \tag{3.10}$$

for each spatial variable. The grid is constructed as follows. Given the next time t_1 to which we expect to advance the integration during this step, we choose a (possibly longer) time interval Δt such that $t_0 < t_1 \leqslant t_0 + \Delta t$. The procedure we use for choosing t_1 and Δt will be described in more detail below. The collocation points must lie within this interval, i.e., $0 < \tau_1 < \cdots < \tau_P = \Delta t$. The best placement is determined by the basis functions and properties of the integrand. The Chebyshev points

$$\tau_p = \frac{\Delta t}{2}\left[1 - \cos\left(\frac{p\pi}{P}\right)\right] \tag{3.11}$$

are theoretically optimal for polynomial interpolation. Indeed, we have found empirically (in a small number of cases) that this grid, in which points are more densely located near both end points of the interval, yields substantially better results than uniformly spaced points.

We next act with $\exp[A(t-\tau)]$ on each c_j by forming a small Krylov space. That is, for each N-vector c_j, we form Ψ_j and the Krylov matrix $\tilde{A}_j \equiv \Psi_j^\dagger A_j \Psi_j$, and its diagonal representation $\tilde{A}_j = U_j \Lambda_j U_j^{-1}$. Approximating the action of A as in (3.7), we arrive at

$$e^{A_j(t-\tau)} c_j \approx \Psi_j U_j e^{\Lambda_j(t-\tau)} U_j^{-1} \Psi_j^\dagger c_j$$

which leads to the following expression for the integral:

$$\begin{aligned}
&\int_0^t e^{A(t-\tau)} R(y(\tau))\, d\tau \\
&\quad = \int_0^t \sum_j \Psi_j U_j e^{\Lambda_j(t-\tau)} U_j^{-1} \Psi_j^\dagger c_j \phi_j(\tau)\, d\tau \\
&\quad = \sum_j \Psi_j U_j \left(\int_0^t e^{\Lambda_j(t-\tau)} \phi_j(\tau)\, d\tau\right) U_j^{-1} \Psi_j^\dagger c_j
\end{aligned} \tag{3.12}$$

For polynomial ϕ, each time integral

$$I_j \equiv \int_0^t e^{\lambda(t-\tau)} \tau^j\, d\tau \tag{3.13}$$

can be calculated analytically via

$$I_0 = \frac{1}{\lambda}\left[e^{\lambda t} - 1\right] \tag{3.14}$$

and the recursion relation

$$I_j = \frac{1}{\lambda}\left[jI_{j-1} - t^j\right] \tag{3.15}$$

For small λt, the integral must be evaluated instead by Taylor expansion of the exponential in (3.13) to prevent loss of accuracy, as was done in

(3.7) for the linear term. Another numerical pitfall arises in connection with the collocation (3.10): the matrix whose elements are $(\tau_p)^{j+1}$ can be ill-conditioned. A more stable procedure is to calculate coefficients $\tilde{c}_j$ satisfying

$$R(y(\tau_p)) = \sum_{j=1}^{P} \tilde{c}_j \left(\frac{\tau_p}{\Delta t}\right)^{j+1} \tag{3.16}$$

and then to rescale the coefficients.

There exists the option of using fewer functions ϕ_j than gridpoints τ_p, i.e., taking in the sum (3.12) an upper limit J, with $J < P$. The coefficients c_j may be determined by a least-squares fit. Alternatively, the coefficients may still be determined via (3.15) or (3.16), but then not used in the sum. This allows the neglected, high-order functions to be used for dealiasing purposes (including them can make the low-order coefficients more accurate) without the expense of forming the corresponding Krylov spaces and integrating; in this way we also retain the convenience of a single grid and collocation procedure.

Equation (3.8) is an integral equation for the exact correction to the linear approximation obtained from Eq. (3.7). The method of evaluation of the integral described above introduces limitations to the accuracy of the solution, dependent upon the size of each Krylov space and the number and type of time-dependent functions utilized in the expansion of R. In practice, a small number of iterations is often sufficient to yield good results.

4. IMPLEMENTATION

4.1. Error Analysis

The final functional form for y is assembled as

$$\begin{aligned} y(t) &= \Psi U \frac{e^{\Lambda t} - I}{\Lambda} U^{-1} \Psi^{\dagger} b \\ &\quad + \sum_{j=1}^{J} \Psi_j U_j \left(\int_0^t e^{\Lambda_j (t-\tau)} \phi_j(\tau)\, d\tau \right) U_j^{-1} \Psi_j^{\dagger} c_j \end{aligned} \tag{4.1}$$

Equation (4.1) can be differentiated analytically:

$$\begin{aligned} \frac{dy}{dt}(t) &= \Psi U e^{\Lambda t} U^{-1} \Psi^{\dagger} b \\ &\quad + \sum_{j=1}^{J} \Psi_j U_j \Lambda_j \left(\int_0^t e^{\Lambda_j (t-\tau)} \phi_j(\tau)\, d\tau \right) U_j^{-1} \Psi_j^{\dagger} c_j \\ &\quad + \sum_{j=1}^{J} \Psi_j U_j \phi_j(t)\, U_j^{-1} \Psi_j^{\dagger} c_j \end{aligned}$$

The first two terms in the above expression are easily evaluated at the gridpoints, i.e. for $t=\tau_p$, at negligible additional cost as by-products of the evaluation of the linear solution (3.7) and the integral (3.12). The third term is seen to be merely $R(y(t))$ if $J=P$, since

$$\Psi_j U_j \phi_j(t)\, U_j^{-1} \Psi_j^\dagger c_j = \Psi_j \Psi_j^\dagger c_j \phi_j(t) = \Psi_j e_1 \phi_j(t) = c_j \phi_j(t)$$

Thus dy/dt is given by

$$\frac{dy}{dt}(t) = \Psi U e^{\Lambda t} U^{-1} \Psi^\dagger b + \sum_{j=1}^{J} \Psi_j U_j \Lambda_j \left(\int_0^t e^{\Lambda_j (t-\tau)} \phi_j(\tau)\, d\tau \right) U_j^{-1} \Psi_j^\dagger c_j + R(y(t)) \tag{4.2}$$

Given values $y(\tau_p)$, the right-hand side of the original differential equation (3.1) can also be computed as $f(x_0 + y(\tau_p))$, to be compared with the evaluation of (4.2). We define a residual vector via

$$\Delta y \equiv \tau_p \left[\frac{dy}{dt}(\tau_p) - f(x_0 + y(\tau_p)) \right] \tag{4.3}$$

This local error estimator has proven to be quite accurate in numerous test cases. We have chosen at present to use as our criterion for acceptable accuracy the r.m.s. norm of the relative error used by LSODE (Hindmarsh and Brown, 1987):

$$\left[\frac{1}{N} \sum_{i=1}^{N} \left(\frac{\Delta y_i}{|x_i| + \delta} \right)^2 \right]^{1/2} \leqslant \text{RTOL} \tag{4.4}$$

where RTOL is a relative error tolerance, and $\delta = \text{ATOL}/\text{RTOL}$ the ratio of absolute to relative error tolerances [necessitated by small values of $|Y(i)|$], both supplied by the user. We have also obtained satisfactory results using a maximum component error of the form

$$\frac{\max_{1 \leqslant i \leqslant N} |\Delta y_i|}{\left(\frac{1}{N} \sum_{i=1}^{N} x_i^2 \right)^{1/2}} \leqslant \text{RTOL}$$

4.2. Control Structure

We first describe the step-by-step procedure that is used to advance the solution from a time t_0 where it is known to a new time t_1. The interval Δt determines a grid via (3.11). Evaluation of the convolution integral via the collocation procedure outlined above requires that the solution be calculated at each of the grid points. Thus a set of possible t_1 values are available at each stage of the calculation. At the end of a step, the latest time $\tau_{\bar{p}}$ compatible with the user-specified error tolerance is selected. The new time interval is chosen so that the middle point τ_{mid} of its collocation grid is equal to the previously accepted time step. If none of the solutions is acceptable, the solution is not advanced: a smaller interval is chosen and the solution calculated on the new grid.

At present, the parameters determining the size of the Krylov spaces and number of fitting polynomials in the convolution term are fixed at the start of the simulation for all time. We have optimized the parameters by hand for each problem studied in this article The objective of this is to demonstrate the potential of the method given a proper optimization scheme. If a closely related set of problems is to be extensively studied, one can imagine this approach being worthwhile for real applications. However, for use as a general package it is clearly essential to install automatic optimization methods. This will be the subject of future communications.

For future reference, we define the complete parameter set needed to concretely specify the algorithm. The number of self-consistent convolution iterations is M, and the number of polynomial powers in a given iteration m is J_m. The dimension of the linear Krylov space is denoted K_L, while those for the convolution integral are labeled K_{mj}, where m is the iteration number and j labels the polynomial in time.

With these values defined, the algorithm proceeds as follows:

(1) Initialize t_0 and $x_0 \equiv x(t_0)$. Define $y(t) \equiv x(t_0 + t) - x_0$. Estimate a time interval Δt.

(2) Evaluate the right-hand side of the differential equation $b \equiv f(x_0)$ at the current time.

(3) Act repeatedly with the Jacobian $A \equiv Df(x_0)$ on b to form the linear Krylov space of size K_L, i.e., form Ψ, U, Λ according to Eqs. (2.3) and (2.4).

(4) Form the Chebyshev grid τ_p of size P corresponding to Δt according to Eq. (3.11).

(5) Obtain the linear solution $y^{(0)}(\tau_p)$ via Eq. (3.7) at each grid point.

For $m = 1, 2, ..., M$:

(6) Evaluate the remainder term $R(y^{(0)}(\tau_p))$ at each grid point.

(7) Solve the collocation equations (3.16) for the vector polynomial coefficients c_j.

For $j = 1, 2, \ldots, J_m$:

(8) Form a Krylov space of size K_{mj} for each polynomial.

(9) Evaluate the convolution integrals via Eqs. (3.13)–(3.15) at each grid point and add to $y^{(0)}$.

(10) Determine the local error at each grid point via Eqs. (4.2)–(4.4).

(11) Select the largest time $\tau_{\bar{p}}$ on the grid whose error is less than the value RTOL set by the user, and advance the solution to this point. That is, set

$$t_1 \leftarrow t_0 + \tau_{\bar{p}}$$

$$x_1 \leftarrow x_0 + y(\tau_{\bar{p}})$$

$$\Delta t \leftarrow \Delta t(\tau_{\bar{p}}/\tau_{\text{mid}})$$

Redefine $y(t) \equiv x(t_1 + t) - x_1$

(12) If none of the errors are sufficiently small, set $\Delta t \leftarrow \Delta t/2$ and repeat steps (4)–(11).

5. KROGH MODEL

5.1. Description of the Model

Our first test of the new integrator is based on a set of equations proposed by Krogh (Gear, 1971). It is particularly well suited as a test case since an exact solution is known at all times, and it has been used previously for testing BDF schemes (Gear and Saad, 1983). Following Gear and Saad we begin with a set of N functions, z_i, $i = 1, 2, \ldots, N$ defined by

$$\frac{dz_i}{dt} = \beta_i z_i + \gamma z_i^2 \tag{5.1}$$

where the β_i are a set of negative constants and γ is a parameter determining the magnitude of the nonlinearity. The exact solution for all times is

$$z_i(t) = \frac{-\beta_i}{\gamma + c_i e^{-\beta_i t}} \tag{5.2}$$

with c_i determined by the initial values $z_i(0)$. A linear transformation is made by defining a vector

$$x = Vz \tag{5.3a}$$

$$V \equiv I - \frac{2uv^\dagger}{v^\dagger u} \tag{5.3b}$$

Table I. Optimal Parameter Sets for the Krogh Problem

RTOL	10^{-6}							10^{-4}					10^{-2}			
	m		K_L					m		K_L			m		K_L	
	m	J_m	K_{m1}	K_{m2}	K_{m3}	K_{m4}	K_{m5}	m	J_m	K_{m1}	K_{m2}	K_{m3}	m	J_m	K_{m1}	K_{m2}
$\gamma = 3$	0		12					0		14			0		15	
$\beta_{\min} = -1000$	1	5	2	2	1	1	1	1	2	2	1		1	2	2	2
	2	5	5	3	3	2	2	2	2	3	2					
$\gamma = 10$	0		10					0		10			0		15	
$\beta_{\min} = -1000$	1	4	2	1	1	1		1	3	2	2	2	1	2	2	2
	2	4	4	2	2	2										
$\gamma = 100$	0		11					0		12			0		12	
$\beta_{\min} = -1000$	1	4	2	1	1	1		1	2	2	1		1	2	2	2
	2	4	3	3	3	2		2	2	2	2					
$\gamma = 3$	0		11					0		12			0		15	
$\beta_{\min} = -5000$	1	4	2	1	1	1		1	2	2	1		1	2	2	2
	2	4	2	2	2	1		2	2	2	2					
$\gamma = 10$	0		10					0		12			0		13	
$\beta_{\min} = -5000$	1	3	2	1	1			1	2	2	1		1	2	2	2
	2	3	3	2	2			2	2	2	2					
$\gamma = 100$	0		10					0		10			0		13	
$\beta_{\min} = -5000$	1	3	2	1	1			1	2	2	1		1	2	2	2
	2	3	2	2	2			2	2	2	2					

where u and v are N-vectors. It is easy to construct an equivalent set of differential equations for x, which are to be integrated numerically, and then to transform x back to the variables z_i for comparison with the exact solution (5.2). We investigate several values of γ and the β_i in what follows. Although V is an $N \times N$ matrix, multiplication by V is inexpensive because of its form (5.3b). Our choice of V follows that of Gear and Saad.

5.2. Implementation of the Krylov Integrator

As our methods are principally aimed at integrating large sets of equations, we study Krogh models with $N = 800$. An explicit Euler code and LSODE were used for comparison with the Krylov integrator. All methods were given the same initial conditions and integration was carried out for 2 model seconds using a Cray X-MP supercomputer.

Because of the large number of equations to be integrated and the lack of sparsity in the Jacobian, the most efficient version of LSODE is one that utilizes an internal diagonal approximation to the Jacobian (method flag 23): we have verified this explicitly. We examined six cases, characterized by three possible values of $\gamma (\gamma = 3, \gamma = 10$, and $\gamma = 100)$ and two possible sets of the β_i:

$$\beta_1 = -5000, \qquad \beta_2 = -4000, \qquad \beta_3 = -2500, \qquad \beta_4 = -1500$$

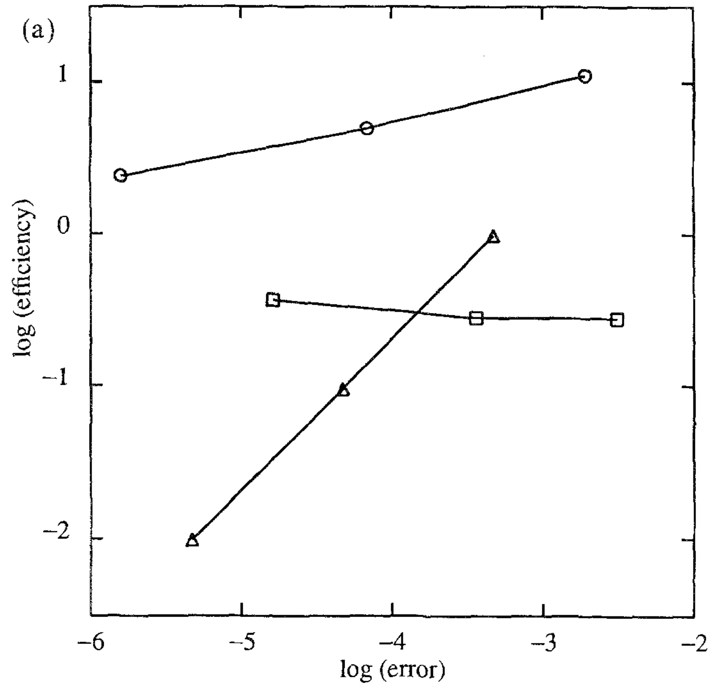
or

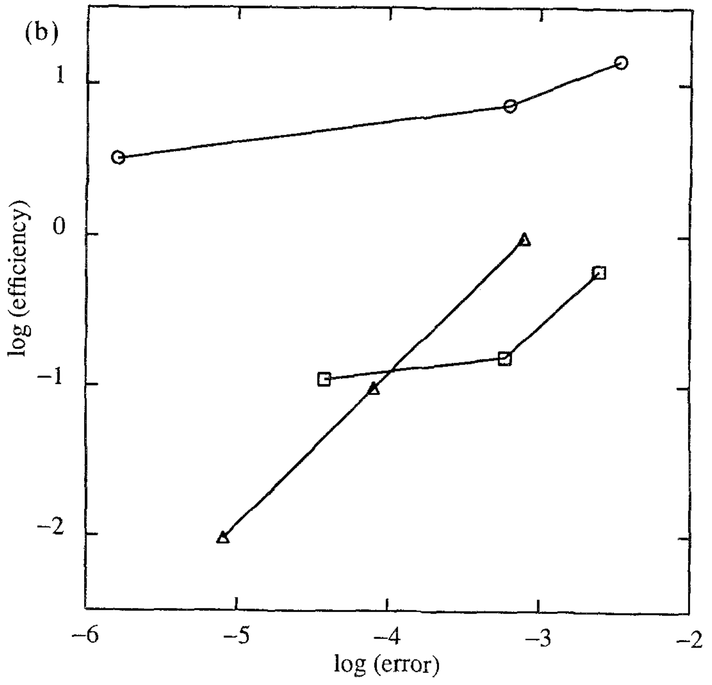
$$\beta_1 = -1000, \qquad \beta_2 = -800, \qquad \beta_3 = -500, \qquad \text{and } \beta_4 = -300$$

All remaining $\beta_i \equiv -100(N-i+1)/(N-5)$ for $i \geqslant 5$. The equations were integrated in the transformed variable x, and comparison to the exact solution (5.2) for z could be made at any time via the transformation $z = V^{-1}x$.

Table I presents a "manual" determination of the optimal Krylov integrator parameters for each of the six combinations of γ and β_i presented above. The optimal set of parameters is that which yields the maximum efficiency, i.e., model seconds per CPU seconds. Three levels of accuracy were investigated, controlled by the three values of RTOL [see Eq. (4.4)] given in the first row. For all runs, we set ATOL $= 10^{-10}$. The first column

Fig. 1. (a) Efficiency versus global error for Krogh problem (5.1) using three integration methods. Efficiency is defined as number of model seconds per CPU seconds. The global error is defined by Eq. (5.4). Model parameters are $\gamma = 100$, $\beta_{\min} = -5000$. Triangles denote Euler integration, squares denote LSODE integration using an internal diagonal approximation to the Jacobian (method flag 23), and circles denote Krylov integration using the optimized parameters given in Table I. (b) Same as Fig. 1a with model parameters $\gamma = 3$ and $\beta_{\min} = -1000$.


(a)
log (efficiency)
log (error)
1
0
−1
−2
−6
−5
−4
−3
−2


(b)
log (efficiency)
log (error)
1
0
−1
−2
−6
−5
−4
−3
−2

in each box gives the values of m, the number of self-consistent convolution iterations, as in Eq. (3.9) (here, $m=0$ specifies linear propagation). The second column lists the number of polynomial powers, J_m, for each convolution iteration, $m \geqslant 1$. Column three gives the size K_L of the linear Krylov space; below it is the size K_{m1} of the Krylov space used to propagate the lowest power in the nonlinear term. Subsequent columns give the sizes K_{mj} corresponding to higher powers. We began with $M=2$ and adjusted the values of J_m, K_{mj}, M, and K_L. Each parameter was varied in turn so as to maximize efficiency while retaining the desired global accuracy. This rather crude approach to optimization has undoubtedly not led to definitive results; however, it is sufficient to illustrate the order of magnitude improvement in performance of which the exponential propagation method is capable.

Figures 1a and 1b compare the performance of the Krylov integrator with forward Euler integration and LSODE for the model values $\gamma=100$ and $\beta_1=-5000$, and for $\gamma=3$ and $\beta_1=-1000$, respectively. The measure used for the error requires some explanation. The local error estimator (4.4) used internally by both LSODE and the Krylov integrator is a function of δ, which in turn depends on the absolute and relative error tolerances selected by the user, i.e., $\delta \equiv \text{ATOL}/\text{RTOL}$. In order to define an exact global error for the evaluation of each method, regardless of the values of RTOL or ATOL used for local error control during the integration, as well as for the Euler integration with fixed step size, we fix the value $\delta = 10^{-4}$ for this purpose, and define the exact global error in all cases to be

$$\left[\frac{1}{N}\sum_{i=1}^{N}\left(\frac{x_i^{\text{exact}}-x_i^{\text{computed}}}{|x_i^{\text{exact}}|+10^{-4}}\right)^2\right]^{1/2} \tag{5.4}$$

where x^{exact} is computed from (5.2) and (5.3).

The Krylov method tended to produce exact global errors close to the imposed value of RTOL. Indeed, all solutions considered for the optimizations in Table I had an exact global error of less than $10 \times \text{RTOL}$. This was not the case for LSODE: surprisingly, it proved to be difficult to improve the accuracy of LSODE beyond the limits shown in Figs. 1a and 1b. To achieve a global error of 10^{-5} in the case of Fig. 1a, RTOL was chosen to be 10^{-6}, but smaller values of RTOL did not reduce this error by another order of magnitude. In the case of Fig. 1b, attaining an exact global error of 4×10^{-5} using LSODE required setting $\text{ATOL}=10^{-12}$ and $\text{RTOL}=10^{-14}$, and the global error could not be further reduced. This difficulty in integrating the second case is unexpected, since the nonlinearity is smaller and the set of differential equations less stiff than in the first case.

Figures 1a and 1b show that the efficiency of the Krylov integrator exceeds that of LSODE by at least a factor of 10. The efficiency of the Krylov integrator is also at least 5 times that of Euler integration, with a much higher advantage for high accuracy. While the efficiency of Euler integration increases rapidly with decreased demands in accuracy and correspondingly increased step size, it becomes unstable and so is useless in obtaining global errors larger than 10^{-3}. Results for the remaining sets of model parameters γ and β_i (which lie between the two cases of Fig. 1) are quite similar.

We turn now to a study of the efficiency and accuracy of exponential propagation as a function of the different internal parameters of the integrator. Only one Krogh model ($\gamma = 100$, $\beta_1 = -5000$) optimized to an accuracy level around 10^{-6} is discussed below; other cases yield similar results. All parameters except the one being varied are held at the previously optimized values given in Table I.

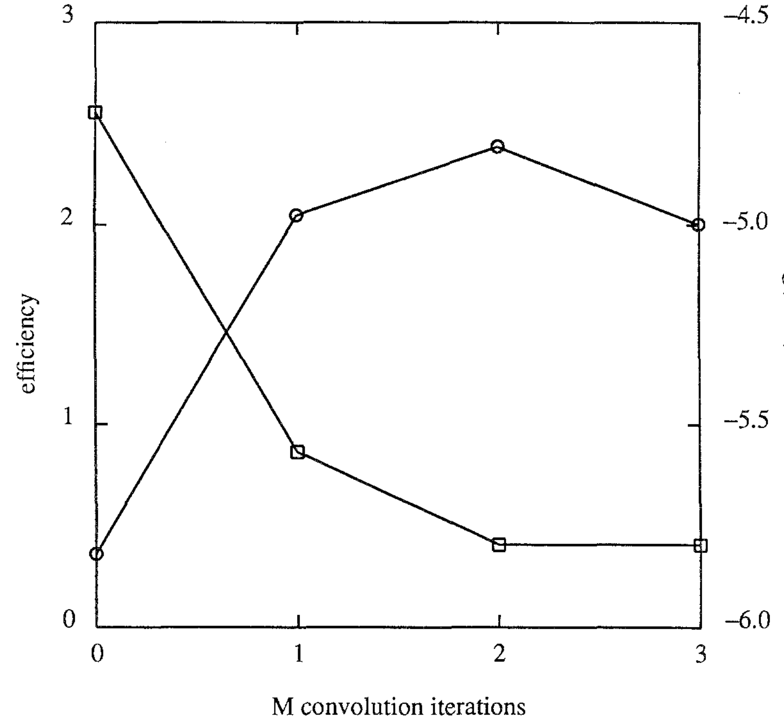


Fig. 2. Efficiency and global error for the Krylov method as a function of M, the total number of convolution iterations. Model parameters here and in all subsequent figures are $\gamma = 100$, $\beta_{\min} = -5000$, and $N = 800$; RTOL is set to 10^{-6}. Efficiency (scale on left) is denoted by circles; error (scale on right) is indicated by squares. Optimal efficiency is attained at $M = 2$ (see also Table I). When $M = 0$ (linear propagation only), efficiency is an order of magnitude lower, and error an order of magnitude higher than cases with nonzero M.

Figure 2 plots performance as a function of M, the number of self-consistent iterations [Eq. (3.9)]. Note that the efficiency and accuracy are substantially inferior for $M = 0$; this demonstrates the importance of the nonlinear correction procedure. The maximum in efficiency coupled with no loss of accuracy at $M = 2$ indicates that this value is optimal, given the assumed values of the parameters; see also Table I. The fact that $M = 1$ and $M = 3$ are nearly as good suggests that the efficiency of the method is not overly sensitive to parameter choice, an important consideration in designing robust automatic procedures in which the parameters are chosen without human intervention.

The sharpest behavior is observed in Fig. 3, for the number J_m of polynomials in the time-dependent fitting. The optimization of this aspect of the method will require careful consideration. Figure 5 displays a curious feature not shared by Figures 2, 3, or 4: the global error actually increases somewhat with K_L, especially after the maximum efficiency is attained at $K_L = 10$; if it persists, this behavior could prove useful in automating the optimal choice of K_L.

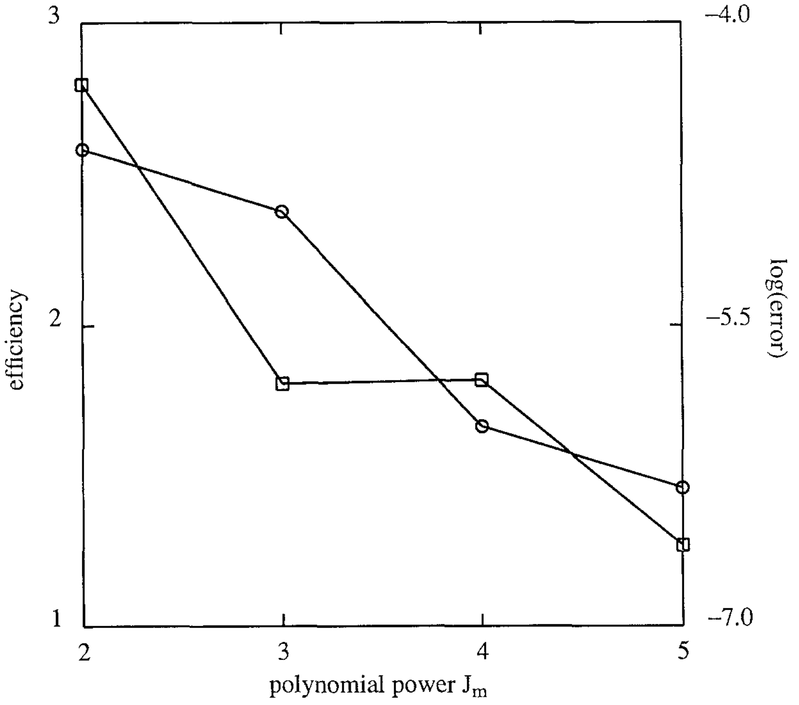


Fig. 3. Efficiency (circles) and global error (squares) for the Krylov method as a function of J_m, the number of polynomial powers used to calculate the convolution integral, with conventions as in Fig. 2. Both quantities display a strong dependence on J_m.

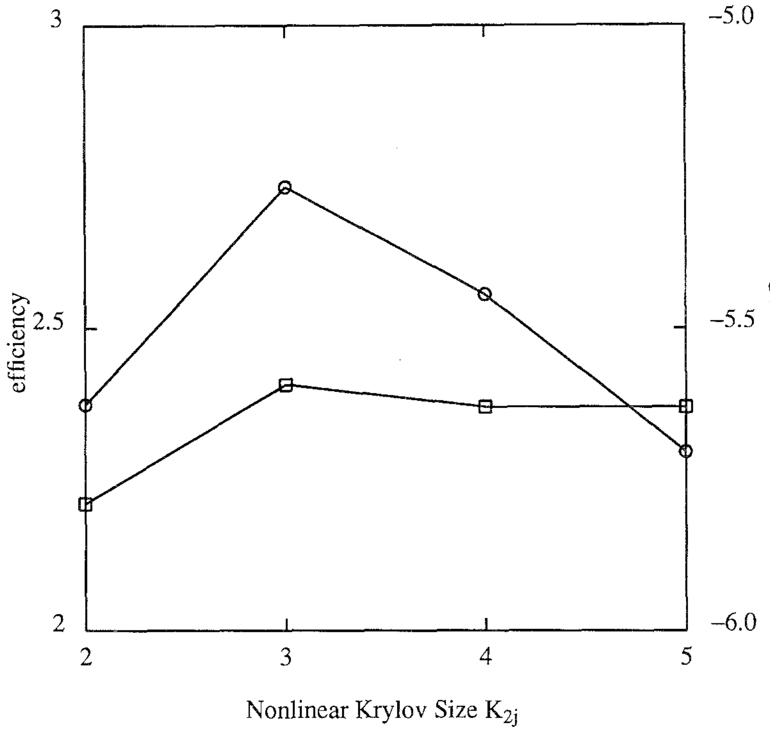


Fig. 4. Efficiency (circles) and global error (squares) as functions of K_{2j}, the size of the Krylov spaces used for propagating the nonlinear term in the final convolution integration. J_m is held constant, and K_{2j} is increased or decreased by the same amount for each j.

6. A REACTION-DIFFUSION SYSTEM

6.1. Motivation

The formation of spatiotemporal patterns in chemical reaction–diffusion systems has been the subject of much research in recent years. In the course of this research it is natural to study simplified models in order to understand the basic mechanisms leading to spatial pattern formation and loss of spatiotemporal coherence. The basic form of a reaction–diffusion equation is

$$\frac{d\mathbf{u}}{dt} = \mathbf{f}(\mathbf{u}) + \mathbf{D}\nabla^2\mathbf{u} \tag{6.1}$$

where **u** is a vector of chemical concentrations, **f**(**u**) is a function describing the homogeneous chemical kinetics, and **D** is a diagonal matrix of diffusion

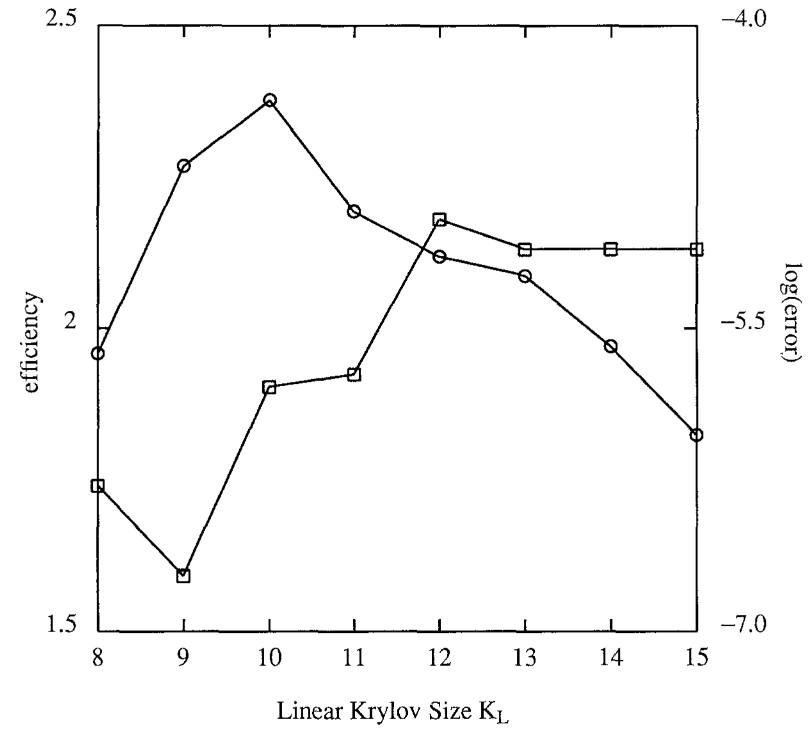


Fig. 5. Efficiency (circles) and global error (squares) as functions of K_L, the size of the linear Krylov space. As in Figs. 2 and 4, there is a choice of parameter, here $K_L = 10$, that maximizes efficiency for other constants fixed. In contrast to the dependence on M, J_m, and K_{2j}, the global error increases with K_L.

coefficients. Numerical treatment of such models is typically hampered by at least two complications:

- The Laplacian operator is often discretized using a finite difference approximation. The most negative eigenvalue of such an operator discretized in this way grows quadratically with the number of grid points chosen, while the eigenvalue of smallest magnitude does not change with the grid size; the system is thus stiff for a sufficiently fine grid.
- The dynamics of reaction–diffusion systems often involves motion of extremely steep fronts. Such fronts are characterized by very rapid variations in time-series taken at individual locations when the front passes. Whenever the time-series at a location undergoes a rapid excursion, the error control used by integration packages automatically decreases the time step. Since some point is under-

going such a rapid excursion at any time the time step usually remains small.

We chose a particular simplified "chemical" model for our tests of the Krylov method. The kinetic law is chosen not for its realistic modeling of a particular chemical reaction, but rather for its simplicity. This model has been used successfully by Arneodo and Elezgaray (1989) in simulating one-dimensional chemical patterns. The kinetic law in the absence of diffusion is

$$\frac{du}{dt}=\frac{1}{\varepsilon}\left[v-(u^2-u^3+u^5)\right] \tag{6.2a}$$

$$\frac{dv}{dt}=\alpha-u \tag{6.2b}$$

Adding diffusion causes the formation of a sharp front parallel to the x axis, given the boundary conditions

$$\begin{aligned}
u_0&=u(x,\ y=0)=1.1,\\
u_1&=u(x,\ y=1)=-1.5,\\
v(x,\ y=0)&=u_0^2-u_0^3+u_0^5,\\
v(x,\ y=1)&=u_1^2-u_1^3+u_1^5.
\end{aligned}$$

Periodic boundary conditions are imposed in x. We set $D=0.045$ for both species, and $\alpha=0.01$, $\varepsilon=0.01$. The front oscillates in the y direction with a period of roughly 0.2 model seconds. In all simulations described below the grid was 50×50 grid points. The initial condition was chosen to be the state 10 model seconds after a uniform initial condition (LSODE having been used to integrate to this state). By this time the front is well formed. The same initial condition was used in every run.

6.2. Results

Our choice of grid size leads to a total number of ODEs of $N=50\times 50\times 2=5000$, and therefore to a large, but sparse, Jacobian. This constitutes the primary obstacle for integrators of the Gear type. One approach to this is employed by LSODE: the user can select an internally generated diagonal approximation to the Jacobian. Another approach is that used by Brown and Hindmarsh (1986, 1987, 1989) in the experimental package LSODPK, where iterative Krylov subspace methods are used to solve the linear systems. LSODPK allows the user to precondition the Jacobian for improved convergence.

All runs of LSODE were performed with the diagonal approximation to the Jacobian. In early runs of LSODPK we used two-sided preconditioning, partitioning the Jacobian into reaction (R), x-diffusion (D_x), and y-diffusion (D_y) parts. The left preconditioner was chosen to be $(I-hR)$; the right preconditioner was $(I-hD_x)(I-hD_y)$, where h is a scale factor. Later we performed runs of LSODPK without any preconditioning and obtained similar solutions, but with considerably less computational effort. We report here only those results of LSODPK without preconditioning. All timing runs were run for five model seconds.

Since no exact solution to this problem is known, we could not perform the same optimizations that were applied to the Krogh problem. We attempted to form an accurate solution by means of LSODE with a small value of the relative error tolerance RTOL, and an absolute error tolerance ATOL $=1.0\times10^{-10}$, and also with the Krylov method using a small RTOL, but found that the phase of the solution was quite sensitive to the setting of RTOL, and to the various Krylov space parameters, so the difference between a test solution and the "accurate" solution was dominated

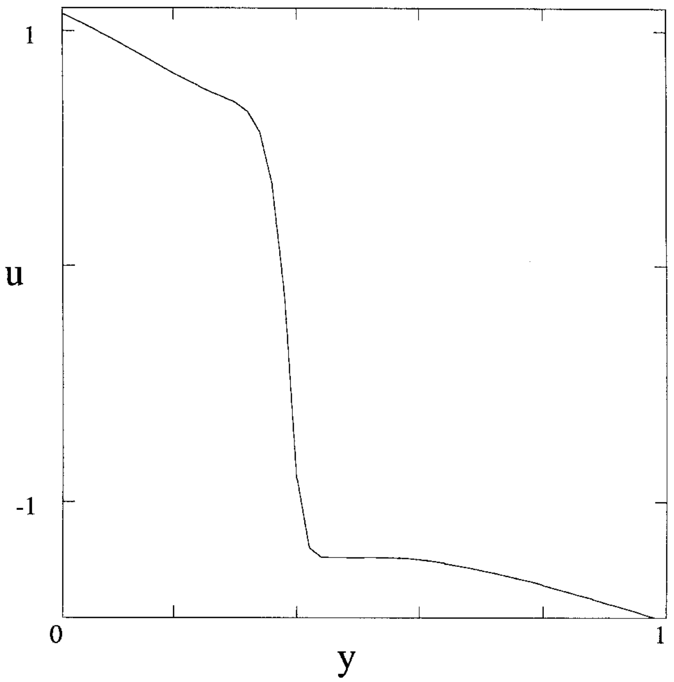


Fig. 6. The concentration of species u as a function of spatial variable y for the reaction-diffusion problem (6.1) and (6.2). This profile is characteristic of the solution of the model system at the parameter values stated. The sharp front oscillates back and forth in the y direction.

by a line of points near the front where the two solutions were out of phase; the steepness of the front guarantees that any small phase error will cause two such solutions to differ greatly near the front. Thus we abandoned the tactic of comparing test solutions against a more accurate solution. Since in our study of this model we would be interested in qualitative, rather than quantitative, behavior, we chose instead to compare the efficiency of LSODE, LSODPK, and the Krylov method as RTOL is *increased* until the methods become unstable or until the *character* of the solution no longer agrees with that obtained at tighter tolerance.

Runs with the Krylov method were done with a single self-consistent iteration, i.e., $M = 1$. It was found that increasing the number of such iterations did not qualitatively improve the solution, but increased computational effort significantly. The size K_L of the linear Krylov space was varied between 5 and 7. The number of powers J was varied between 3 and 5, and the sizes K_1, K_2, and K_3 of the polynomial Krylov spaces were also varied between 3 and 5. It was found that the choice of K_L influenced the efficiency of the method most. The optimum choice of the other K's was 3.

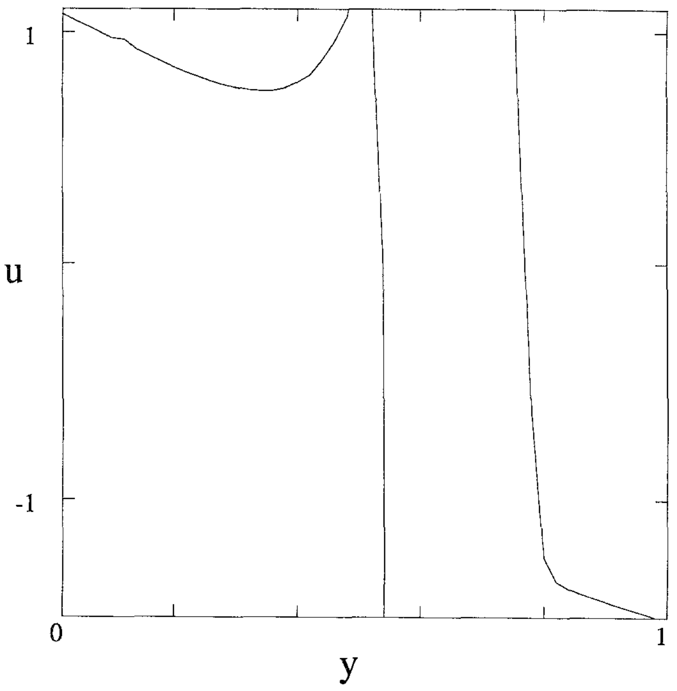


Fig. 7. Solution produced by LSODE with RTOL = 0.05 after 3.5 model seconds, illustrating the numerical instability. The solution at 3.0 sec was the same as that in Fig. 6.

A typical solution generated by this model is shown in Fig. 6. It is interesting to note that when LSODE or LSODPK began to give poor results it usually followed that they were unstable, and the solutions grew rapidly without bound. Figure 7 shows the field after 3.5 model seconds using LSODE and an RTOL of 0.05. When the Krylov method began to give unsatisfactory results, it typically produced fields in which there was a large line of spikes parallel to the front, which died down within a few time steps (see Fig. 8). Thus, while the solution was no longer satisfactory, the method continued to integrate without instability.

A summary of the results of our comparison is given in Table II. It can be seen from these results that whereas LSODPK and LSODE required at least 62.5 CPU seconds and 39.8 CPU, respectively, satisfactory results were obtained using the Krylov method in as few as 8.8 CPU seconds. For this problem, it appears that the iterative solution (either with or without preconditioning) of the backwards difference formulas performed by LSODPK is not advantageous.

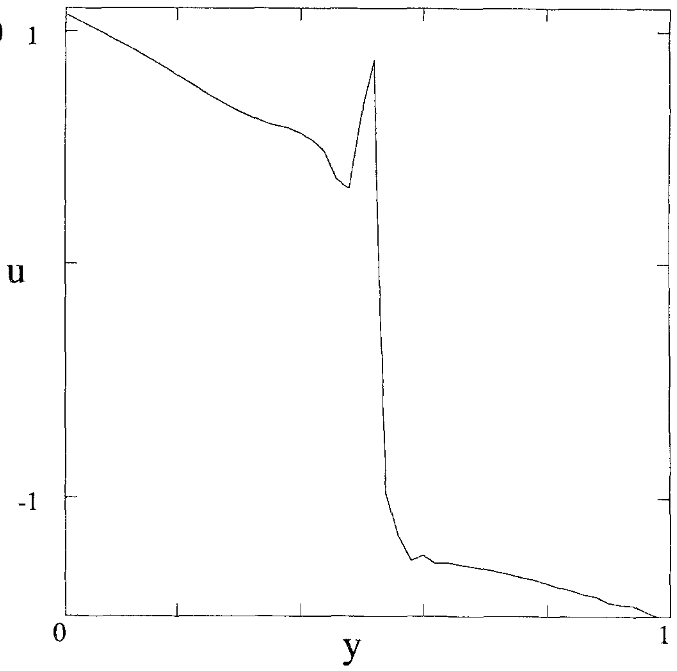
We have also carried out a similar set of numerical experiments for this model using a forward Euler code. This is typically the method of choice (e.g., Jahnke, Skaggs, and Winfree, 1989) for problems of the type considered in this section, viz., those involving a large number of variables, rapid spatial and temporal variation of the solution at every time step, and an accuracy requirement only for qualitatively faithful simulation. The maximum time step before the onset of instability is 1.0×10^{-3} sec, in agreement with the largest eigenvalue of the Jacobian found by the Krylov integrator. A total of 14 CPU seconds were required to integrate the five model seconds, as compared to 8.8 CPU seconds for the Krylov method. This again illustrates the fact that the Krylov method is able to take time steps considerably larger than that prescribed by the explicit stability limit.

7. CONCLUSION

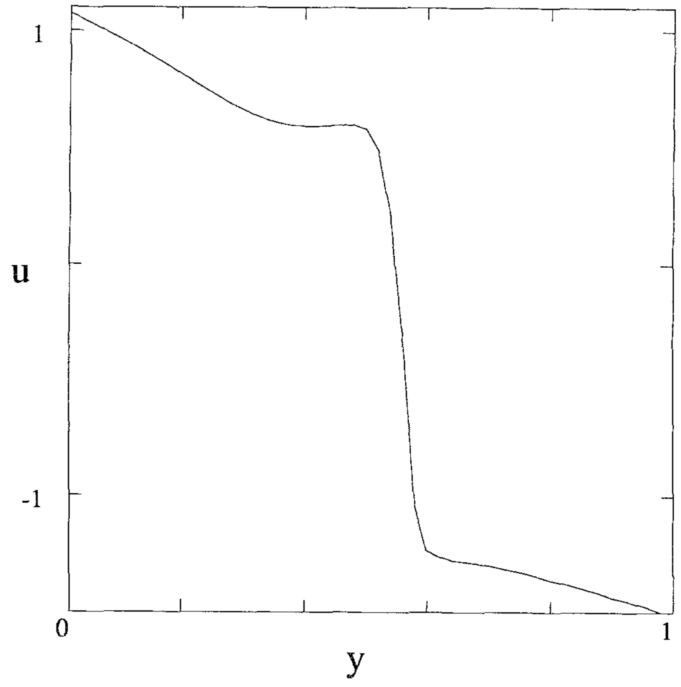
The Krogh test problems best illustrate the power of the exponential propagation method as it is presently constituted. The solutions of these equations are relatively smooth, and efficient propagation of the linear part of the problem is crucial to the overall efficiency. The exponential method is able to take extraordinarily long time steps under these conditions, thus leading to the large gains in efficiency over the alternatives that we find here. As there are a substantial number of important large-scale problems

Fig. 8. (a) Solution produced by the Krylov method at RTOL = 5.0, illustrating the typical behavior when the solution was no longer acceptable. This profile was obtained after 2.41 model seconds. (b) The solution at $t = 2.84$; the spike evident in (a) has disappeared.

(a)



(b)



of this type, we expect that the exponential method will at least find a niche in this area. Evaluation of the performance of the method as a robust, problem-independent program, as is LSODE, awaits further development and automation, which may incur additional computational costs.

An important point is that in order to realize these advantages the set of equations must be larger than the size of the Krylov space. Otherwise, the "reduction" to an Arnoldi matrix is insignificant, and the Jacobian may just as well be diagonalized directly. Numerical experiments confirm that exponential propagation with diagonalization of the full Jacobian is inefficient as compared to backward differentiation approaches, because diagonalization is much more expensive than solution of an equivalent set of linear equations. The larger time steps permitted cannot compensate for this expense when nonlinearities are substantial.

Table II. Reaction–Diffusion Results

LSODE results

RTOL	CPU seconds	Notes
0.005	46.3	
0.01	39.8	
0.05	41.9	UNSTABLE

LSODPK results

RTOL	CPU seconds	Notes
0.01	62.5	
0.05	33.4	POOR solution
0.1	35.1	POOR solution
0.5	39.1	UNSTABLE

Krylov results

RTOL	K_L	CPU seconds	Notes
0.5	5	11.9	
1.0	5	8.8	
5.0	5	7.1	POOR solution
0.5	6	13.2	
1.0	6	10.9	
5.0	6	6.1	POOR solution
0.5	7	13.1	
1.0	7	11.6	Short-lived kink which later recovers
5.0	7	10.4	POOR solution

The chemical front simulation illustrates a major difficulty in using the method for systems where the Jacobian itself is changing rapidly at all times in the simulation. Under such conditions, it is very difficult to accurately predict the field at later times using a linearized approximation to the evolution equations; thus, the exponential method is limited to much shorter time steps than were possible in the previous case. The Gear package is plagued by a similar difficulty, being based on a predictor–corrector methodology as well, and is similarly unable to take long time steps. This is why the advantage over forward Euler at the stability limit is only a factor of 2, as compared to the factor of 5–10 achieved for the Krogh equations. The present results are nevertheless encouraging, in that an advantage over both Euler and Gear methods is in fact obtained.

In addition to the obvious course of automation and optimization of control parameters, a number of improvements will be pursued in the future. The use of time-dependent fitting functions other than polynomials will be investigated; such functions could be generated by various means, e.g., from information in the linearized solution, or from knowledge of the physics of the problem. The Krylov approach meshes well with adaptive grid technology, as only multiplication by the Jacobian is necessary, so that an ordered structure for the differential operators is not crucial. This also suggests treating different regions of the grid with different time-dependent methods. For example, one could carry out accurate, short time-step integration at a wavefront to generate a predictor in this region, use the exponential method as a predictor elsewhere, and insert these results into the integral equation to synthesize the predictors into a globally accurate answer via the corrector. Such multilevel strategies are probably the only hope of integrating equations like those in Section 6 with a qualitatively greater efficiency than Euler integration.

In summary, we have provided evidence that an algorithm based upon direct exponentiation in a Krylov subspace is a plausible candidate for a general stiff equation integrator for large systems. The method is easy to implement and apply to an arbitrary system of equations, on the basis of the description provided in this article. Further development of the method should improve its performance and delineate the range of problems for which it is the method of choice.

ACKNOWLEDGMENTS

We are indebted to Alan Hindmarsh for extremely helpful discussions, and for providing us with a version of the program LSODPK, as well as numerous references. We thank Harry Swinney for his interest and for motivating this project. We are grateful to Xavier Roussy for participating in an early phase of this project. This research was begun while R.A.F. and L.S.T. were visitors at the Institute for Theoretical Physics at the University of California at Santa Barbara, which is supported in part by the NSF under grant No. PHY82-17853, and supplemented by funds from NASA. R.A.F. is a Camille and Henry Dreyfus Teacher-Scholar and the recipient of a Research Career Development Award from the National Institutes of Health, Institute of General Medical Sciences. L.S.T. is supported in part by NSF grant No. DMS-8901767 and by an SRA grant from the University of Texas. Support for this research was also provided by a Texas Advanced Research Project, grant No. 1100 (B.C.D) and by the British Petroleum Venture Research Unit (T.V.R). Computing resources for this work were provided by the University of Texas System Center for High Performance Computing.

REFERENCES

Arneodo, A., and Elezgaray, J. (1990). *Phys. Lett. A.* **143**, 25.

Arnoldi, W. E. (1951). *Q. Appl. Math.* **9**, 17.

Brown, P., and Hindmarsh, A. C. (1986). *SIAM J. Num. Anal.* **23**, 610.

Brown, P., and Hindmarsh, A. C. (1989). *J. Appl. Math. Comp.* **31**, 40.

Byrne, G. D., and Hindmarsh, A. C. (1987). *J. Comput. Phys.* **70**, 1.

Christodoulou, K. N. and Scriven, L. E. (1988). *J. Sci. Comput.* **3**, 355.

Cullum, J., and Willoughby, R. (1985). *Lanczos Algorithms for Large Symmetric Eigenvalue Computations*, Birkhauser, Boston.

Gear, C. W. (1971). *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, Englewood Cliffs, New Jersey.

Gear, C. W., and Saad, Y. (1983). *SIAM J. Sci. Stat. Comput.* **4**, 583.

Goldhirsch, I., Orszag, S. A., and Maulik, B. K. (1987). *J. Sci. Comput.* **2**, 33.

Hindmarsh, A. C. (1983). In R. S. Stepleman *et al.* (eds.), *Scientific Computing*, North-Holland, Amsterdam, pp. 55–64.

Hindmarsh, A. C., and Brown, P. (1987). In R. Vichnevetsky and R. S. Stepleman (eds.), *Advances in Computer Methods for Partial Differential Equations—VI*, IMACS, New Brunswick, pp. 355–362.

Jahnke, W., Skaggs, W. E., and Winfree, A. T. (1989). *J. Phys. Chem.* **93**, 740.

Leforestier, C., Bisseling, R., Cerjan, C., Feit, M., Friesner, R., Guldberg, A., Hammerich, A., Jolicard, G., Karrlein, W., Meyer, H. D., Lipkin, N. Roncero, O., and Kosloff, R. (1989). *J. Comput. Phys.*, submitted.

Park, T., and Light, J. C. (1986). *J. Chem. Phys.* **85**, 5870.

Saad, Y. (1980). *Lin. Alg. Appl.* **34**, 269.