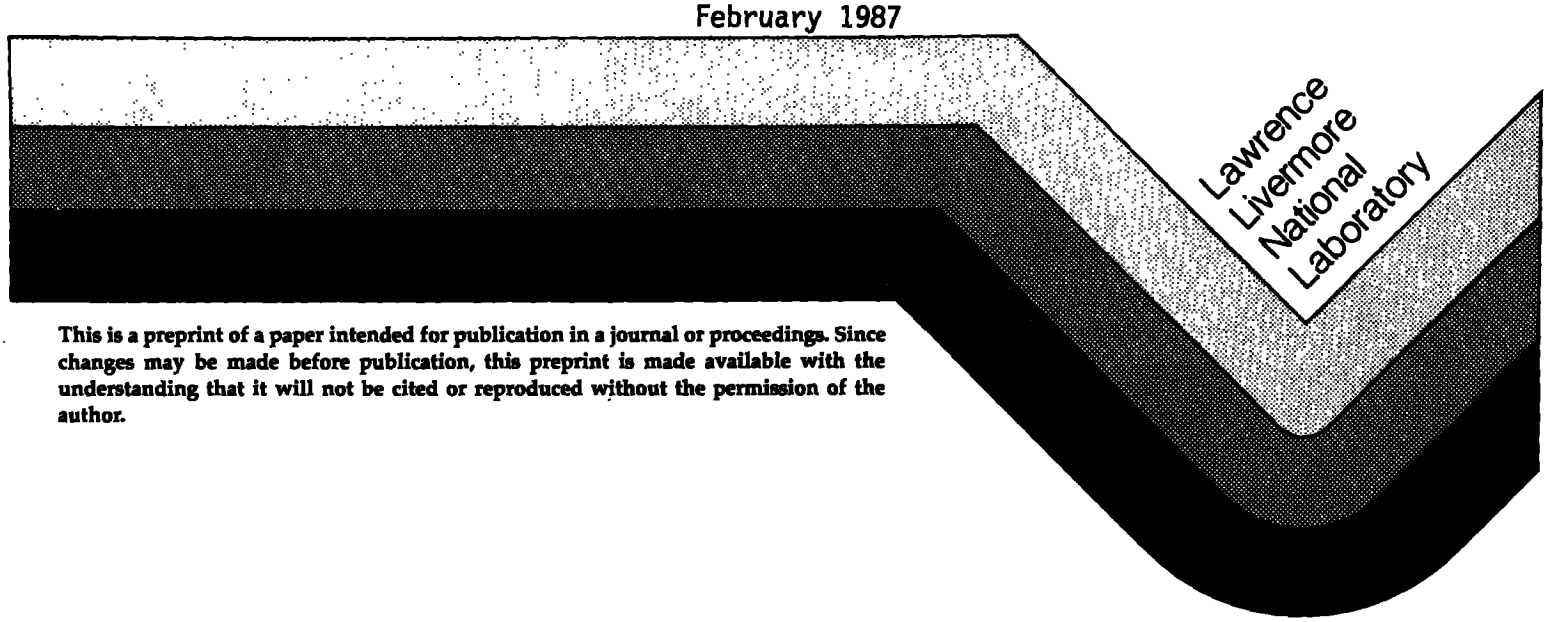
UCRL- 96261
PREPRINT

REDUCED-STORAGE TECHNIQUES IN THE NUMERICAL METHOD OF LINES

Alan C. Hindmarsh
Peter N. Brown

This paper was prepared for presentation at the
6th IMACS Symposium on Computer PDE Methods,
Bethlehem, PA, June 23-26, 1987.

February 1987



REDUCED-STORAGE TECHNIQUES IN THE NUMERICAL METHOD OF LINES*

Alan C. Hindmarsh
and
Peter N. Brown
Computing and Mathematics Research Division, L-316
Lawrence Livermore National Laboratory
Livermore, CA 94550
USA

Abstract

The method of lines replaces a PDE problem by an ODE initial value problem which is typically stiff and often solved by BDF methods. This normally requires the system Jacobian matrix. But Krylov subspace iteration methods solve linear systems without explicit need for the matrix. Within a BDF method and Newton (nonlinear) iteration, a Krylov method such as GMRES (Generalized Minimum Residual) can be used with the matrix involved only in operator form by way of a difference quotient. We present a scaled and preconditioned GMRES algorithm called SPIGMR. For reaction-transport PDE systems, several preconditioners arise in a natural way, using the reaction and transport operators separately, or in succession as in operator splitting. A variant of the general purpose solver LSODE, called LSODPK, contains various preconditioned Krylov methods. Tests on a reaction-diffusion system demonstrate their effectiveness.

1. Introduction

There has been much recent interest in the use of iterative linear equation techniques in the solution of large systems of stiff oridinary differential equations (ODE's). Since numerical methods for stiff ODE's are typically implicit, the integrator must solve a system of nonlinear equations at each integration step. Newton-like methods are most often used to solve these nonlinear systems, and hence require the solution of a linear system for each Newton iterate. The usual approach to solving the linear systems is to use a direct technique such as Gaussian elimination (or some variant of it). For large problems, the associated linear algebra often makes up most of the work and required core memory involved in the integration of the ODE's.

The Method of Lines (MOL) has long been an available technique for solving systems of time-dependent partial differential equations (PDE's). When solving PDE's by MOL techniques, the spatial operators are discretized while time is continuous, resulting in ODE systems which are often stiff. The high quality (and number) of available stiff ODE solvers makes this an attractive approach. However, the use of direct linear equation methods in most ODE codes has restricted the use of MOL techniques to problems of modest size, due to storage limitations and the cost of solving the linear equations. Iterative linear solvers have also been used in ODE solvers (e.g. GEARBI [9]), but until recently have assumed a certain form for the problem, have required some condition on the coefficient matrix in order to guarantee convergence, and in some cases have involved the estimation of certain method parameters which are problem-dependent.

The development of Krylov subspace projection methods for general nonsymmetric linear systems requiring no parameter estimation (cf. Saad [13,14], and Saad and Schultz [15]) has sparked a renewed interest in using iterative linear methods in stiff ODE solvers. In addition, the Krylov methods only require the action of the coefficient matrix times a vector, not the matrix explicitly. In the nonlinear equations setting, the coefficient matrix is the Jacobian of a nonlinear function. Hence, the matrix-vector multiply can be approximated using a difference quotient. This has the effect of eliminating the need to store the Jacobian matrix, and so can lead to drastic reductions in storage over direct methods. For robustness, some preconditioning of the linear systems seems needed, and this typically requires some matrix storage, but still much less than that needed by direct methods. The work of Miranker and Chern [12], Gear and Saad [8], Chan and Jackson [6], and Brown and Hindmarsh [4,5] has centered on determining the overall effectiveness of this approach, and/or investigating the theoretical aspects of the combined stiff ODE method/Newton/Krylov algorithm. The initial results indicate this is a very promising approach, and the reduced storage costs and robustness of the Krylov methods offer the promise of solving much larger systems than previously possible using MOL techniques.

In the remainder of the paper, we attempt to give an overview of this approach. Section 2 contains the necessary background on Newton and Krylov methods. Section 3 discusses scaling and preconditioning when using the Krylov methods. In Section 4, we present several types of preconditioning one can use when solving reaction-diffusion PDE systems. Section 5 discusses the implementation of the Krylov methods in the code LSODPK, a variant of LSODE [10,11], and then in Section 6 we present an example illustrating the various approaches outlined in Section 4.

2. Newton/Krylov Methods in Stiff ODE's

We are interested here in the numerical solution of the ODE initial value problem

$$\dot{y} = f(t,y) \ , \ y(t_0) = y_0 \quad (\dot{} = d/dt \ , \ y \in R^N). \tag{2.1}$$

We will assume that the ODE in (2.1) is stiff, meaning that one or more strongly damped modes are present, and will use the popular Backward Differentiation Formula (BDF) methods to solve it. These methods have the general form

$$y_n = \sum_{j=1}^{q} \alpha_j \, y_{n-j} + h\beta_0 \dot{y}_n \ , \ \dot{y}_n = f(t_n,y_n) \ , \tag{2.2}$$

where q is the method order (normally $1 \le q \le 5$). The BDF methods are implicit, and so at each time step one must solve the algebraic system

$$y_n - h\beta_0 f(t_n,y_n) - a_n = 0 \ , \ a_n \equiv \sum_{j=1}^{q} \alpha_j \, y_{n-j} \tag{2.3}$$

for y_n ($\beta_0 > 0$). If x_n is defined by

$$x_n = h\dot{y}_n = (y_n - a_n)/\beta_0 \ ,$$

then (2.3) is equivalent to

$$F_n(x_n) \equiv x_n - hf(t_n, a_n+\beta_0 x_n) = 0 \ , \tag{2.4}$$

*This work was performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

and this is the system we will deal with. A Newton-type method applied to (2.4) has the following form:

$$\begin{cases} \text{Let } x_n(0) \text{ be an initial guess for } x_n. \\ \text{For } m = 0, 1, 2, \ldots \text{ until convergence, do:} \\ \quad \text{Solve } A\, s_n(m) = -F_n(x_n(m)) \\ \quad \text{Set } x_n(m+1) = x_n(m) + s_n(m) , \end{cases} \tag{2.5}$$

where the coefficient matrix A is some value of (or an approximation to a value of)

$$F_n'(x) = I - h\beta_0 J(t,y) \quad (y = a_n + \beta_0 x) ,$$

where $J(t,y) = \partial f/\partial y$, the system Jacobian matrix. Many ODE solvers attempt to save work by computing and storing A (and its decomposition factors if a direct method is used to solve (2.5)) once, and using it for all iterations on that step. Futhermore, A is usually also held fixed over several steps of the integration, only discarding the current A when it is determined to be sufficiently out of date. This results in <u>modified Newton</u> iteration, and typically gives a linear rate of convergence, whereas the full Newton scheme uses $A = F_n'(x_n(m))$ and gives a quadratic rate of convergence.

When using an iterative method to solve (2.5) approximately, one can view the basic iteration scheme as either modified Newton or the full Newton scheme. Modified Newton requires the forming and storing of A for use over several steps. One then solves (2.5) approximately by some iterative technique. One example of this approach is the ODE solver GEARBI [9], which uses Block-SOR as the basic iterative solver. Another is the solver developed by Chan and Jackson [6], which uses the Krylov methods called Conjugate Residual and Orthomin(k) as the iterative solvers.

Since the full Newton scheme requires the use of $A = F_n'(x_n(m))$, this approach only makes sense when A does not need to be formed explicitly. In fact, the Krylov methods typically only require the action of A times a vector v. Hence, using a difference quotient of the form

$$Av = F_n'(x)v \approx [F_n(x + \sigma v) - F_n(x)]/\sigma \tag{2.6}$$

($x = x_n(m)$) will approximate the desired multiplication. This approach has been taken by the authors in [4,5] and continued here. We have referred to these methods as <u>matrix-free</u> due to the absence of required storage for A. With either approach there is an error associated with solving (2.5) only approximately, and with the second approach there is an additional error resulting from the approximation (2.6). With this second approach, one can view the resulting nonlinear iteration scheme as being in the class of <u>Inexact Newton methods</u> as developed by Dembo, Eisenstat and Steihaug [7].

The particular Krylov subspace projection method we will consider here is the Generalized Minimum Residual Method (GMRES) due to Saad and Schultz [15]. (Other choices are possible [5], but GMRES seems to be the best overall choice.) To describe the algorithm, consider the general linear system

$$Ax = b, \tag{2.7}$$

where A is an N x N matrix, and x and b are N-vectors. Here, (2.7) represents the full Newton equations (2.5) with $A = F_n'(x_n(m))$, $b = -F_n(x_n(m))$ and the solution vector x represents the increment $s_n(m) = x_n(m+1) - x_n(m)$. If x_0 is an initial guess for the true solution $x_* = A^{-1}b$ of (2.7), then on letting $x = x_0 + z$ we get the equivalent system

$$Az = r_0 , \tag{2.8}$$

where $r_0 = b - Ax_0$ is the initial residual. Let K_ℓ be the <u>Krylov subspace</u>

$$K_\ell = \text{span}\{r_0, Ar_0, \ldots, A^{\ell-1}r_0\} .$$

By a <u>Krylov subspace projection method</u> on K_ℓ we mean a method which finds an approximation

$$x_\ell = x_0 + z_\ell, \quad \text{with } z_\ell \in K_\ell .$$

To specify uniquely x_ℓ (or z_ℓ), GMRES requires that

$$\begin{aligned} \|b - Ax_\ell\|_2 &= \min_{x \in x_0 + K_\ell} \|b - Ax\|_2 \\ &= \min_{z \in K_\ell} \|r_0 - Az\|_2 , \end{aligned} \tag{2.9}$$

where $\|\cdot\|_2$ denotes the Euclidean norm.

The minimization problem (2.9) is solved by constructing an orthornormal basis for K_ℓ. This is accomplished by using an <u>Arnoldi process</u> [1], which can be stated as follows:

1. Compute $r_0 = b - Ax_0$ and set $v_1 = r_0/\|r_0\|_2$.
2. For $j = 1, \ldots, \ell$, do:

$$w_{j+1} = Av_j - \sum_{i=1}^{j} h_{ij}v_i , \quad h_{ij} = (Av_j, v_i)$$

$$h_{j+1,j} = \|w_{j+1}\|_2 , \quad v_{j+1} = w_{j+1}/h_{j+1,j} .$$

Here, $(\cdot,\cdot)$ is the usual Euclidean inner product. If we let $V_j = [v_1, \ldots, v_j]$ $(j=1,\ldots,\ell+1)$ denote the N x j matrix with columns v_i, and $\bar{H}_\ell = (h_{ij})$ the $(\ell+1)$ x ℓ upper Hessenberg matrix whose nonzero entries are given by the above h_{ij}, then Saad and Schultz [15] have noted that

$$\bar{H}_\ell = V_{\ell+1}^T A V_\ell , \quad V_j^T V_j = I_j , \text{ and } AV_\ell = V_{\ell+1}\bar{H}_\ell , \tag{2.10}$$

where I_j is the j x j identity matrix. Furthermore, $\text{span}\{V_j\} = K_j$ $(j=1,\ldots,\ell+1)$, where we have assumed that the dimension of K_j is j.

If we let $z = V_\ell y$ for $y \in R^\ell$, then in (2.9) we have

$$\begin{aligned} \|r_0 - Az\|_2 &= \|\beta v_1 - AV_\ell y\|_2 = \|V_{\ell+1}(\beta e_1 - \bar{H}_\ell y)\|_2 \\ &= \|\beta e_1 - \bar{H}_\ell y\|_2 , \end{aligned}$$

where $\beta = \|r_0\|_2$, $e_1 = (1, 0, \ldots, 0)^T \in R^{\ell+1}$, and we have used (2.10) and the fact that $V_{\ell+1}$ has orthonormal columns. Thus the solution of (2.9) is $z_\ell = V_\ell y_\ell$, where y_ℓ solves

$$\min_{y \in R^\ell} \|\beta e_1 - \bar{H}_\ell y\|_2 . \tag{2.11}$$

We solve (2.11) in terms of a QR factorization performed on $\bar{H}_\ell$ using Givens rotations. The GMRES algorithm can now be stated as follows:

Algorithm 2.1 (GMRES):

1. Compute $r_0 = b - Ax_0$ and set $v_1 = r_0/\|r_0\|_2$
2. For $\ell = 1, \ldots, \ell_{max}$, do:
   (a) $w_{\ell+1} = Av_\ell - \sum_{i=1}^{\ell} h_{i\ell} v_i$, $h_{i\ell} = (Av_\ell, v_i)$
   $h_{\ell+1,\ell} = \|w_{\ell+1}\|_2$, $v_{\ell+1} = w_{\ell+1}/h_{\ell+1,\ell}$
   (b) Evaluate $\rho_\ell = \|b - Ax_\ell\|_2$
   (c) If $\rho_\ell \le \delta$ go to Step 3; otherwise go to (a)
3. Compute $x_\ell = x_0 + V_\ell y_\ell$, where y_ℓ solves (2.11)

In Step 2(b) above, Saad and Schultz [15] have noted that ρ_ℓ may be calculated without forming x_ℓ or y_ℓ. See Brown and Hindmarsh [5] or Saad and Schultz [15] for more details. The parameters δ and ℓ_{max} are user-specified, ℓ_{max} normally being dictated by storage considerations, and δ specifying the amount of accuracy desired in the solution x_ℓ. See Brown and Hindmarsh [4,5] for more details regarding δ, ℓ_{max}, and other properties of the algorithm.

Saad and Schultz [15] have shown that GMRES will converge to the true solution x_* in at most N iterations. The only type of breakdown that can occur here is when $w_{\ell+1} = 0$. This is referred to as a happy breakdown since in this case one can show that $x_\ell = x_*$. In Algorithm 2.1, as ℓ gets large, much work is required to make $v_{\ell+1}$ orthogonal to all the previous vectors $v_1, \ldots, v_\ell$. Brown and Hindmarsh have considered an incomplete version of GMRES, denoted IGMRES, which differs from Algorithm 2.1 only in that the sum in Step 2(a) begins at $i = i_0$ instead of $i = 1$, where $i_0 = \max(1, \ell-p+1)$ and p is the number of vectors to which $v_{\ell+1}$ is to be made orthogonal, namely $v_{\ell-p+1}, \ldots, v_\ell$. (The complete case is $p = \ell_{max}$.) See [5] for more details on IGMRES, including an inexpensive (indirect) way to calculate the residual norm.

Finally, the theoretical aspects of the combined Newton/Krylov method involve analyzing how the errors associated with the difference quotient approximation (2.6) and solving (2.5) approximately affect the accuracy of the computed x_ℓ. The reader is referred to the papers by Brown [2], and Brown and Hindmarsh [4,5] for a detailed analysis.

3. Scaling and Preconditioning

In the context of the solving $\dot{y} = f(t,y)$ by BDF method (2.2), and of a Newton iteration (2.5) on the nonlinear algebraic system (2.4) at each step, recall that the job of the GMRES algorithm is to solve (to within some precision) the linear systems $Ax = b$ in which

$$\left.\begin{aligned} A &= I - h\beta_0 J(t_n, a_n + \beta_0 x_n(m)) \\ b &= -F_n(x_n(m)) = -x_n(m) + hf(t_n, a_n + \beta_0 x_n(m)) \\ x &= x_n(m+1) - x_n(m) . \end{aligned}\right\} \quad (3.1)$$

That is, the solution vector x is the Newton increment for the m^{th} Newton (nonlinear) iteration, and A is a value of $I - h\beta_0 J$, J being the system Jacobian, $J = \partial f/\partial y$. However, the GMRES algorithm as it stands is only of limited use for this purpose. For realistic problems, it is unlikely to give acceptable performance unless it is enhanced. We consider two kinds of enhancements here--scaling and preconditioning.

Scaling simply means applying weight factors to the components of a vector before computing its norm, and may be dictated by the fact that different components are in different physical units. In the ODE setting, weight factors are already available in terms of tolerance parameters RTOL and ATOL, namely

$$w_i = RTOL_i \left|y^i_{n-1}\right| + ATOL_i . \quad (3.2)$$

To be specific, a weighted root-mean-square (WRMS) norm with these weights is used in place of the Euclidean norm. The two norms can be related by way of the diagonal scaling matrix

$$D = \sqrt{N}\, \mathrm{diag}(w_1, \ldots, w_N) , \quad (3.3)$$

so that for any vector x,

$$\|x\|_{WRMS} = \|D^{-1} x\|_2 .$$

The linear system $Ax = b$ can then equivalently be expressed in scaled form

$$(D^{-1}AD)\,(D^{-1}x) = (D^{-1}b), \quad \text{or } \hat{A}\hat{x} = \hat{b} ,$$

such that Euclidean norms are appropriate on $\hat{x}$ and the associated vectors in a Krylov method for this system.

Scaling allows the various components of an error vector or residual to be compared with one another, but it may or may not help to speed convergence of a Krylov iteration method for $Ax = b$. To do that, we enhance the iteration by preconditioning. In general, this means we seek two matrices, P_1 and P_2 (the left and right preconditioners), for which the iteration is more effective on the equivalent system

$$(P_1^{-1}AP_2^{-1})\,(P_2 x) = (P_1^{-1}b), \quad \text{or } \bar{A}\bar{x} = \bar{b} .$$

Whether or not preconditioning really does enhance the method depends on the extent to which these matrices can be found so that

(1) linear systems $P_1 x = c$ and $P_2 x = c$ are economically solved, and

(2) the product $P_1 P_2$ approximates A in some sense ($\bar{A}$ is close to the identity matrix).

Preconditioning can be done on both sides, or on one side only (the other matrix being the identity).

Scaling and preconditioning can be applied to the system together by composing the above two transformations of the matrix and vectors involved. The result is a system

$$\bar{A}\bar{x} = \bar{b}, \quad \text{with} \tag{3.4}$$

$$\bar{A} = D^{-1}\bar{A}D = D^{-1}P_1^{-1}AP_2^{-1}D ,$$

$$\bar{x} = D^{-1}\bar{x} = D^{-1}P_2 x, \quad \bar{b} = D^{-1}\bar{b} = D^{-1}P_1^{-1}b .$$

The GMRES iteration, or any other Krylov iteration, can then be applied to this transformed system (3.4). The initial residual $r_0 = b - Ax_0$ is first transformed to $\bar{r}_0 = D^{-1}P_1^{-1}r_0$. Then the Krylov sequence $\bar{K} = \{\bar{r}_0, \bar{A}\bar{r}_0, \bar{A}^2\bar{r}_0, \ldots\}$ is generated, with the finite-dimensional subspaces $\bar{K}_\ell$ represented by basis vectors $\bar{v}_1, \bar{v}_2, \ldots, \bar{v}_\ell$. When a computed solution $\bar{x}_\ell$ is chosen from $\bar{x}_0 + \bar{K}_\ell$, it is back-transformed to $x_\ell = P_2^{-1}D\bar{x}_\ell$, and x_ℓ lies in $x_0 + P_2^{-1}D\bar{K}_\ell$. A fact that is perhaps surprising is that this latter ℓ-dimensional subspace is independent of D, and also independent of whether the preconditioners are arranged with P_1 on the left and P_2 on the right, or P_1P_2 on the left, or P_1P_2 on the right. Although the subspace has this independence, the choice of x_ℓ made within the subspace does not, since the scaling and preconditioners all enter into the minimality condition (2.9) and the convergence test (Step 2(c) of Algorithm 2.1).

The convergence test for the preconditioned iteration requires further attention, because we compute (cheaply) the residual norm

$$\rho_\ell = \|\bar{r}_\ell\|_2 = \|D^{-1}P_1^{-1}r_\ell\|_2 = \|P_1^{-1}r_\ell\|_{WRMS}$$

rather than $\|r_\ell\|_{WRMS}$. In order to impose a test that is similarly inexpensive but excludes the bias associated with the factor P_1^{-1}, we scale the test constant δ (which ρ_ℓ is tested against) by the factor $\|P_1^{-1}r_0\|_{WRMS}/\|r_0\|_{WRMS}$.

The incomplete version of the GMRES algorithm, denoted IGMRES, when combined with scaling and preconditioning, produces an algorithm denoted SPIGMR. This algorithm can now be stated as follows:

Algorithm 3.1 (SPIGMR):

1. (a) Compute $r_0 = b - Ax_0$; stop if $\|r_0\|_{WRMS} < \delta$
   (b) Compute $\bar{r}_0 = D^{-1}P_1^{-1}r_0$, $\|\bar{r}_0\|_2 = \|P_1^{-1}r_0\|_{WRMS}$; set $\delta' = \delta\,\|P_1^{-1}r_0\|_{WRMS}/\|r_0\|_{WRMS}$
   (c) Set $\bar{v}_1 = \bar{r}_0/\|\bar{r}_0\|_2$
2. For $\ell = 1, 2, \ldots, \ell_{max}$, do:
   (a) Compute $\bar{A}\bar{v}_\ell = D^{-1}P_1^{-1}AP_2^{-1}D\bar{v}_\ell$
   (b) Set $\bar{w}_{\ell+1} = \bar{A}\bar{v}_\ell - \sum_{i=i_0}^{\ell} \bar{h}_{i\ell}\bar{v}_i$, where $i_0 = \max(1, \ell-p+1)$, $\bar{h}_{i\ell} = (\bar{A}\bar{v}_\ell, \bar{v}_i)$
   (c) Set $\bar{h}_{\ell+1,\ell} = \|\bar{w}_{\ell+1}\|_2$, $\bar{v}_{\ell+1} = \bar{w}_{\ell+1}/\bar{h}_{\ell+1,\ell}$
   (d) Update QR factorization of $(\bar{h}_{ij})$
   (e) Compute indirectly $\rho_\ell = \|P_1^{-1}r_\ell\|_{WRMS}$
   (f) If $\rho_\ell < \delta'$ go to Step 3; otherwise go to (a)
3. Compute $\bar{z}$ in $\bar{K}_\ell$ via (2.11); set $x_\ell = x_0 + P_2^{-1}D\bar{z}$

4. Reaction-Transport Systems

The SPIGMR algorithm described above, when combined with a stiff ODE solver such as LSODE, represents a combination of rather powerful general purpose techniques--a BDF method for the ODE integration, Newton iteration for the nonlinear algebraic system, and a Krylov iteration for the linear systems. However, to be effective on challenging problems arising from the method of lines, it is necessary to take advantage of particular features of the problem. The preconditioner matrices provide an ideal means for doing that. The combination of BDF, Newton, and GMRES methods with a problem-dependent preconditioner matrix (or matrices) represents a compromise approach to the problem. It is not totally ad hoc, and thus retains flexibility with respect to the scope of problems of interest, but is not totally general either, and so can exploit problem structure. In contrast to a completely ad hoc approach, the consequence of an erroneous or unwise choice of preconditioner here is a higher cost for the solution, not a failure to get a solution.

Consider a reaction-transport PDE system in which a vector $u = u(t,x)$ with p components varies with time t and a space variable x (of any dimension), according to

$$\partial u/\partial t = R(t,x,u) + S(t,x,u) \tag{4.1}$$

plus initial and boundary conditions. Here R represents reaction terms and is a point function of u, while S represents a spatial transport operator. In our motivating model problems, S is dominated by diffusion, but it need not be in general.

Next, suppose a MOL treatment of the problem is done, with a traditional finite difference discretization of (4.1) on a mesh $\{x_i\}$ of q points. (Other choices, including moving grids and finite elements, are possible, and our methods apply or are easily extended to cover them, but we present only the traditional case for the sake of simplicity.) The result is an ODE initial value problem in a vector $y = (y_1, \ldots, y_q)^T$, where y_i approximates $u(t,x_i)$. The

vector y has length N = pq and satisfies an ODE system

$$\dot{y} = f(t,y) = \bar{R}(t,y) + \bar{S}(t,y) \ . \qquad (4.2)$$

Here $\bar{R}$ involves no spatial coupling, while $\bar{S}$ (the discretization of S, including discretized boundary conditions) is dominated by spatial coupling. Thus block i of (4.2) has the form

$$\dot{y}_i = f_i(t,y) = \bar{R}_i(t,y_i) + \bar{S}_i(t, y_1, \ldots, y_q). \qquad (4.3)$$

A preconditioner matrix P needs to approximate the Newton matrix $A = I - h\beta_0 J$ in (3.1), but at the same time be easily solvable (i.e., systems $Px = b$ need to be easily solvable). Thus any approximation $\bar{J}$ to J that includes the dominant features of J can be used to form a preconditioner $P = I - h\beta_0\bar{J}$. We use this idea on reaction-transport problems by fixing our attention on the reaction process or on the transport process separately.

Suppose first that the reaction process is dominant, as far as the stiffness reflected in J is concerned. Then we can consider approximating J by the block-diagonal matrix

$$B = \partial\bar{R}/\partial y = \text{diag}\,(\partial\bar{R}_1/\partial y_1, \ldots, \partial\bar{R}_q/\partial y_q) \qquad (4.4)$$

whose q diagonal blocks (each p x p) are the Jacobians of the one-point reaction terms alone. This gives a preconditioner $P = I - h\beta_0 B$ which is likely to enhance the GMRES iteration considerably, but yet is relatively easy to solve. Some further improvement might be possible at almost no extra cost by including in B the diagonal blocks of the transport Jacobian $\partial\bar{S}/\partial y$ as well, i.e. by setting the i^{th} diagonal block of B to

$$B_i = \partial\bar{R}_i/\partial y_i + \partial\bar{S}_i/\partial y_i \ , \qquad (4.5)$$

so that B is all of the block-diagonal part of J.

Even the reaction Jacobian (4.4) may still be difficult to compute, store, and solve, for a complicated problem. In most cases, supplying partial derivatives of $\bar{R}_i$ in closed form is out of the question. However, as long as the individual $\bar{R}_i$ (or $\bar{R}_i + \bar{S}_i = f_i$ in the case of (4.5)) can be evaluated separately, then a difference quotient scheme to approximate B can be used, and each such evaluation of B would cost about the same as p evaluations of f. The cost of solving linear systems $Px = b$ with a block-diagonal matrix P can be kept to a minimum by constructing and storing the LU factors of the diagonal blocks of P when formed, and backsolving to get $P^{-1}b$. Reevaluation of P would only need to be done periodically, by the same strategy normally used to reduce evaluations of J when all of J is formed. The evaluation and factoring of P can be further speeded up on a multiprocessor, because the q blocks can be processed independently.

Both storage and computational costs for block-diagonal preconditioning can be reduced greatly by a block-grouping scheme. We consider grouping the q spatial points into g groups, such as by a decomposition of the spatial domain, such that only one value of B_i from each such group is a reasonable approximation for the whole group. Then we have reduced both cost and storage by a factor of g/q, with perhaps only a slight loss in convergence speed. If convergence is degraded too much, the grouping needs to be refined or revised. This idea is the analog in space of the current and very effective strategy of periodic evaluations of Jacobians in time.

Now suppose that the transport process is dominant. A natural choice of preconditioners is then one that ignores the coupling between the PDE components at each spatial point, and instead arises from the discretized transport terms $\bar{S}_i$. To be specific, suppose for the sake of the presentation that the spatial derivatives occur linearly in S, so that we can write the finite-differenced term $\bar{S}_i$ as

$$\bar{S}_i(t,y_1,\ldots,y_q) = \sum_{j<i} L_{ij}y_j + B_i y_i + \sum_{j>i} U_{ij}y_j \ . \qquad (4.6)$$

Assume further that an approximate representation of the form (4.6) exists in which all of the p x p coefficient blocks are diagonal, reflecting the fact that component interaction is being ignored here. Then the corresponding approximation to $A = I - h\beta_0 J$ is a matrix P whose p x p blocks are $I - h\beta_0 B_i$ (on the diagonal), $-h\beta_0 L_{ij}$ (below), and $-h\beta_0 U_{ij}$ (above), and all of these blocks are diagonal. This means that the system $Px = b$ decouples into p separate q x q systems $P^{(k)}x^{(k)} = b^{(k)}$ $(k = 1, \ldots, p)$, in which the matrix $P^{(k)}$ for the k^{th} system (for the k^{th} PDE component) has the structure of the discrete spatial differencing used, but is totally independent of the systems for the other components.

Having defined a preconditioner based entirely on transport processes for decoupled components, a number of choices may be available for solving the systems $P^{(k)}x^{(k)} = b^{(k)}$, depending on the nature of the transport. This choice is simplified by the fact that interactions amoung the p components are absent, as if only a transport PDE in a scalar variable u were to be solved. However, the choice could be made differently for different components k, if variations among the $P^{(k)}$ dictated this. For transport dominated by diffusion, one might choose SOR (successive over-relaxation) iteration, using an estimated acceleration parameter ω if possible, or simply using

$\omega = 1$ (i.e., Gauss-Seidel iteration) otherwise. The number of SOR iterations could be fixed or based on a convergence test. Alternatively, it may be more effective to use symmetric SOR (SSOR), where each iteration has a forward sweep through the q points followed by a backward sweep. In special cases, a direct solution of the system may be possible. For example, for simple diffusion on a uniform mesh, a fast Poisson solver may be applicable. To see this, write

$$P^{(k)}x^{(k)} = x^{(k)} - h\beta_0 J^{(k)}x^{(k)} ,$$

where $J^{(k)}x^{(k)}$ represents a scalar discretized diffusion operator $d\Delta^2 u$ (identifying $x^{(k)}$ as the discretization of a scalar function u), and then regard $P^{(k)}x^{(k)} = b^{(k)}$ as a discrete form of the scalar Helmholtz equation

$$\lambda u + \Delta^2 u = s , \qquad (4.7)$$

where $\lambda = -1/h\beta_0 d$ and s is a source function whose discrete form is $\lambda b^{(k)}$. Solvers using FFT techniques are available for discretizations of (4.7).

If only one or the other of the two processes (reaction and transport) is dominant in causing stiffness in the problem, the best approach would seem to be to use only a single preconditioner based on that process. However, if both are important, and preconditioners P_1 and P_2 can be formed economically for the two processes considered separately, then using both is likely to be better than using either one alone. One could precondition with P_1 on the left and P_2 on the right, or the reverse, or their product (in either order) on one side only; there seems to be no clear best choice. In any case, the resulting preconditioned iteration can be regarded as a combination of Krylov iteration and operator splitting, since the use of the P_k amounts to applying each of the two operators present (reaction and transport) separately, while the Krylov iteration (whether GMRES or another method) is to account for the errors committed thereby. If in general we write $\dot{y} = f = f_1 + f_2$ and compute $J_1 = \partial f_1/\partial y$ and $J_2 = \partial f_2/\partial y$, then the operator splitting preconditioner is the product $(I - h\beta_0 J_1)(I - h\beta_0 J_2)$, while the true Newton matrix is $I - h\beta_0(J_1 + J_2)$. A traditional operator splitting approach would proceed without accounting for the difference between these two, but here we demand convergence of the Krylov iteration that is driven by that difference.

5. Implementation

The general purpose initial value ODE solver LSODE [10,11] contains BDF methods for stiff problems and Adams methods for nonstiff problems, with direct full or banded treatment of the Jacobian. An experimental variant of LSODE called LSODPK was written with preconditioned Krylov methods substituted for the direct matrix solvers. In addition to GMRES, LSODPK contains methods denoted IOM (Incomplete Orthogonalization Method), PCG (Preconditioned Conjugate Gradient), and SPCG (Scaled Preconditioned Conjugate Gradient), all with scaling and preconditioning, but in general these offer no advantages over GMRES. The structure of LSODPK is analogous to that of LSODE (see [5] for details). The user must supply a subroutine F for f(t,y) as in LSODE (the name is arbitrary). In addition, the user supplies two routines, with dummy names JAC and PSOL, to provide the preconditioning. JAC is to compute and preprocess any Jacobian-related data used, and PSOL is to compute the solution of linear systems $P_1 x = c$ or $P_2 x = c$ with preconditioner matrices P_k as needed. Thus if some part of the Jacobian is to be used to form a preconditioner, it is evaluated and LU factorizations are done (if appropriate) in JAC, while the backsolve and/or iterative operations to effect the P_k^{-1} are done in PSOL.

Several details are necessary to complete the description of the SPIGMR algorithm of Sec. 3, as used in LSODPK. First, we take the initial guess vector to be $x_0 = 0$, as no clearly better choice is readily available. The difference quotient approximation (2.6) to $F_n'(x)v$ becomes

$$Av = v - h\beta_0[f(t,y+\sigma v) - f(t,y)]/\sigma$$

with $t = t_n$, $y = a_n + \beta_0 x_n(m)$, and a value of $\sigma = 1/\|v\|_{WRMS}$. The limit ℓ_{max} on Krylov iterations and the incompleteness parameter p are set to $\ell_{max} = p = 5$ on default, but both can be reset by the user. The convergence test constant δ is the product $\delta = \delta_1\epsilon_1$ of a heuristic constant δ_1, set to .05 on default (but optionally set by the user), and the tolerance constant ϵ_1 that is imposed on the error in the solution of the nonlinear system $F_n(x) = 0$ (see [4] for details). Finally, the orthogonalization in Step 2(b) of the algorithm is done by a modified Gram-Schmidt procedure, with conditional reorthogonalization if roundoff effects are severe.

While the LSODPK solver, as it stands, leaves the choice and details of the preconditioning entirely up the user, we have written some modules to achieve the various choices that are natural for MOL solution of PDE's on a rectangular grid in two dimensions. One JAC/PSOL pair uses the total block-diagonal part of the Jacobian. Another uses the reaction-only Jacobian together with a fixed number of SOR (actually Gauss-Seidel) iterations on the transport contributions. A third uses the reaction-only Jacobian and a fast Poisson solver (HWSCRT [16]) for the discrete diffusion

operator. Three other pairs do the same as these three but with block grouping. The grouping is specified by a static Cartesian product partitioning of the 2-D mesh. Work on dynamic grouping schemes is in progress.

The work space storage required by LSODPK, excluding small fixed-size arrays, is 17N words plus the storage needed for preconditioning, when SPIGMR is selected and default values of all parameters are used. This figure includes $\ell_{max}N = 5N$ for the Krylov subspace basis vectors. Thus in order to keep storage under control for large N, it is important to avoid increasing ℓ_{max} by much and to use preconditioners that do not require excessive storage.

6. Example Problem

Tests on a variety of reaction-transport problems are documented in [4] and [5] and show both the effectiveness and some limitations of the methods in LSODPK. We found that without preconditioning, success of the methods correlates with tight clustering in the spectrum of the system Jacobian, whereas carefully preconditioned methods are effective on a wide class of problems.

We present here only one of the test problems from [5]. It is a multi-species food web model [3] in which mutual population competition and predator-prey interaction and diffusion are simulated in a 2-D domain. For a model with s species, the equations in $c = (c^1,\dots,c^s)$ are

$$\partial c^i/\partial t = c^i(b_i + \Sigma a_{ij}c^j) + d_i\nabla^2 c^i \quad (i=1,\dots,s). \quad (6.1)$$

For the tests, we choose a model with 10 prey and 10 predator species (s = 20), with coefficients

$$a_{ii} = -1 \quad (i = 1,\dots,20)$$

$$a_{ij} = -5\cdot10^{-7} \quad (i \le 10,\ j > 10)$$

$$a_{ij} = 10^4 \quad (i > 10,\ j \le 10)$$

$$b_i = 1 + 50xy, \quad d_i = 1 \quad (i \le 10)$$

$$b_i = -(1 + 50xy), \quad d_i = .05 \quad (i > 10)$$

(all other $a_{ij} = 0$). The spatial domain is the square $0 \le x,y \le 1$, and the boundary conditions are all of Neumann type (zero normal derivatives). The relevant time interval (to reach a steady state) is $0 \le t \le 10$. Initial conditions are given by the polynomials

$$c^i(x,y) = 10 + i[16x(1-x)y(1-y)]^2 \quad (1 \le i \le 20).$$

We discretize the PDE system (6.1) on a simple regular 12 x 12 mesh with standard central difference representations of $\nabla^2 c^i$ and the boundary conditions. The resulting ODE system has size $N = 20\cdot12\cdot12 = 2880$. It is stiff because of both the interaction terms and the diffusion. The strong dependence on xy in the b_i produces a wide spread in the steady state values, and in the spectrum of the Jacobian. The tolerances used in these tests are RTOL = 10^{-5} and ATOL = 10^{-8}.

We show here test results for three preconditioners, denoted RO, BD, and OS, and described as follows:

- RO: reaction-only Jacobian, given by Eq. (4.4), calculated from difference-quotient approximations, giving a right preconditioner P_2 ($P_1 = I$).
- BD: block-diagonal part of J, given by Eq. (4.5), also calculated by difference quotients, giving a right preconditioner P_2 ($P_1 = I$).
- OS: operator splitting, using 5 Gauss-Seidel iterations on the diffusion terms as a left preconditioner P_1 (following Eq. (4.6)), together with the reaction-only Jacobian (by difference quotients) for a right preconditioner P_2.

Note that the preconditioner in BD is equal to that in RO plus the contributions of the diagonal transport coefficients. In all three cases we tested these preconditioners both with and without block grouping. Grouping into 36 or 16 groups was done simply by partitioning each direction in the 12 x 12 mesh into 6 or 4 uniform groups.

In performing tests with LSODPK, we collect the following statistics:

- NST = number of time steps
- NFE = number of f evaluations
- NPE = number of preconditioner evaluations (and of LU factorizations of block-diagonal matrices)
- NNI = number of nonlinear iterations
- NLI = number of linear iterations
- NPS = number of preconditioner solves (of calls to PSOL to evaluate vectors $P_k^{-1}v$)
- AVDIM = NLI/NNI = average Krylov subspace dimension ℓ in SPIGMR algorithm
- RT = run time (CPU sec) on a Cray-1 computer

The ratio AVDIM, or the analogous ratio restricted to a subinterval in t, is useful in measuring the success of SPIGMR. If it comes close to ℓ_{max} (= 5 here), then the performance of SPIGMR would probably improve if either ℓ_{max} were increased or the preconditioning were improved. The counter NFE is equal to NNI+NLI+1 (plus the number of internal restarts at order 1, if any).

Test results for this problem are tabulated in Table 1, as a function of the preconditioner choice PRE and the number of groups NGR in the block grouping scheme, if used. All of the choices shown were successful. Other runs (not shown) without preconditioning or with only a Gauss-Seidel preconditioner were unable to finish this problem in a reasonable time. Runs with the BD preconditoner applied on the left seemed to do no better nor worse than those with it on the right. Runs with a fast Poisson solver in place of Gauss-Seidel in OS were considerably slower than OS for this problem. Runs with the diagonal Jacobian blocks supplied in closed form (instead of difference quotients) were up to 20% faster.

TABLE 1. Test Results

PRE	NGR	NST	NNI	NLI	NPE	AVDIM	RT
RO	-	318	363	658	40	1.81	41.9
"	36	330	380	776	43	2.04	37.1
"	16	365	432	1044	50	2.42	45.0
BD	-	331	380	738	42	1.94	46.8
"	36	323	371	715	42	1.93	35.3
"	16	324	378	754	45	1.99	34.8
OS	-	322	367	466	39	1.27	46.6
"	36	323	368	518	40	1.41	39.9
"	16	327	373	560	40	1.50	39.9

All of the choices produce about the same level of accuracy in the solution. Without grouping, all three preconditioners give about the same statistics, except that OS has a significantly lower value of NLI and hence AVDIM. That is, using both the reaction Jacobian and all of the diffusion coefficients in the preconditioning is more helpful in achieving convergence of SPIGMR than any of the less complete choices.

By itself, the AVDIM figure is somewhat deceptive here, however, because it is an average of small values early in the run and large values near the end. In fact, in all of the runs on this problem, the interval average $\Delta NLI/\Delta NNI$ was equal to $\ell_{max} = 5$ exactly for the interval $4 \le t \le 10$. But the various cases differed in their behavior in this interval in that the relatively poor choices suffered from more reductions in step size, forced by convergence failures in SPIGMR, while the better choices had few or no such step size reductions.

The effect of block grouping is to reduce the cost of computing parts of the Jacobian, at the expense of a somewhat slower rate of convergence of the SPIGMR iteration (reflected in NLI and AVDIM). The tradeoff is beneficial in run time until the number of groups is too small. Here 36 groups appears to be optimal for RO, 16 groups for BD, and either value for OS. The least costly choice of all was BD with 16 groups, showing a beneficial tradeoff relative to OS from less costly preconditioning in return for slower convergence, and also showing a slight advantage in convergence speed over RO from the inclusion of the diagonal diffusion coefficients.

The storage costs are reduced considerably with block grouping. The total of the work space lengths is 109,533, or about 38N for any of the three preconditioners without grouping. With 16 groups, this is reduced to about 19.4N, or nearly halved. The portion of this space devoted to preconditioning is reduced from 21N without grouping to (7/3)N with 16 groups. This savings in storage increases roughly as p^2 as the number of species p increases. On the other hand, in a problem like this, where many of the Newton iterations require the full 5 iterations in SPIGMR, it is likely that run times could be reduced by increasing ℓ_{max} above its default value, but this would impose a corresponding additional storage cost of $(\ell_{max}-5)N$.

For comparison, consider a solution of this problem by a more traditional approach, using LSODE, where a direct band solver would be applied to the linear systems. In this example, the half-bandwidths of the matrices are ML = MU = 20·12 = 420, and the total work space required is roughly (11 + 2ML + MU)N = 1271N. This exceeds the LSODPK figure (with SPIGMR and 16 groups) by a factor of more than 65, and it goes up roughly as 3·p·M·N for p species on an MxM mesh. For a problem of this type, even if the storage requirement were not prohibitive, the computational cost (for the band matrix solves) would be.

References

[1] W. E. Arnoldi, "The Principle of Minimized Iterations in the Solution of the Matrix Eigenvalue Problem," Quart. J. Appl. Math., vol. 9, pp. 17-29, April 1951.

[2] P. N. Brown, "A Local Convergence Theory for Combined Inexact Newton/Finite-Difference Projection Methods," to appear in SIAM J. Num. Anal.

[3] P. N. Brown, "Decay to Uniform States in Food Webs," SIAM J. Appl. Math., vol. 46, pp. 376-392, June 1986.

[4] P. N. Brown and A. C. Hindmarsh, "Matrix-Free Methods for Stiff Systems of ODEs," SIAM J. Num. Anal., vol. 23, pp. 610-638, June 1986.

[5] P. N. Brown and A. C. Hindmarsh, "Reduced Storage Matrix Methods in Stiff ODE Systems," to appear in J. Appl. Math. & Comp.

[6] T. F. Chan and K. R. Jackson, "The Use of Iterative Linear Equation Solvers in Codes for Large Systems of Stiff IVPs for ODEs," SIAM J. Sci. Stat. Comp., vol. 7, pp. 378-417, April 1986.

[7] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, "Inexact Newton Methods," SIAM J. Num. Anal., vol. 19, pp. 400-408, April 1982.

[8] W. W. Gear and Y. Saad, "Iterative Solution of Linear Equations in ODE Codes," SIAM J. Sci. Stat. Comp., vol. 4, pp. 583-601, December 1983.

[9] A. C. Hindmarsh, "Preliminary Documentation of GEARBI: Solution of ODE Systems with Block-Iterative Treatment of the Jacobian, LLNL Report UCID-30149, December 1976.

[10] A. C. Hindmarsh, "LSODE and LSODI, Two New Initial Value Ordinary Differential Equation Solvers," in ACM SIGNUM Newsletter, vol. 15, no. 4, pp. 10-11, December 1980.

[11] A. C. Hindmarsh, "ODEPACK, A Systematized Collection of ODE Solvers," in Scientific Computing, R. S. Steplemean et al. (eds.), Amsterdam: North-Holland, 1983, pp. 55-64.

[12] W. L. Miranker and I-L. Chern, "Dichotomy and Conjugate Gradients in the Stiff Initial Value Problem," J. Lin. Alg. Appl., vol. 36, pp. 57-77, March 1981.

[13] Y. Saad, "Krylov Subspace Methods for Solving Large Unsymmetric Linear Systems." Math. Comp., vol. 37, pp. 105-126, July 1981.

[14] Y. Saad, "Practical Use of Some Krylov Subspace Methods for Solving Indefinite and Nonsymmetric Linear Systems," SIAM J. Sci. Stat. Comp., vol. 5, pp. 203-228, March 1984.

[15] Y. Saad and M. H. Schultz, GMRES: A Generalized Minimum Residual Algorithm for Solving Nonsymmetric Linear Systems, SIAM J. Sci. Stat. Comp., vol. 7, pp. 856-869, July 1986.

[16] P. N. Swartztrauber and R. A. Sweet, "Algorithm 541: Efficient Fortran Subprograms for the Solution of Separable Elliptic Partial Differential Equations," ACM Trans. Math. Softw., vol. 5, pp. 352-364, September 1979.