SIAM J. NUMER. ANAL.
Vol. 23, No. 3, June 1986

009

MATRIX-FREE METHODS FOR STIFF SYSTEMS OF ODE'S*

PETER N. BROWN† AND ALAN C. HINDMARSH‡

Abstract. We study here a matrix-free method for solving stiff systems of ordinary differential equations (ODE's). In the numerical time integration of stiff ODE initial value problems by BDF methods, the resulting nonlinear algebraic system is usually solved by a modified Newton method and an appropriate linear system algorithm. In place of that, we substitute Newton's method (unmodified) coupled with an iterative linear system method. The latter is a projection method called the Incomplete Orthogonalization Method (IOM), developed mainly by Y. Saad. A form of IOM, with scaling included to enhance robustness, is studied in the setting of Inexact Newton Methods. The implementation requires no Jacobian matrix storage whatever. Tests on several stiff problems, of sizes up to 16,000, show the method to be quite effective and much more economical, in both computational cost and storage, than standard solution methods, at least when the problem has a certain amount of clustering in its spectrum.

Key words. ordinary differential equations, stiff systems, initial value problems

AMS(MOS) subject classifications. Primary 65L; secondary 65F

1. Introduction. We consider here the numerical solution of the ODE initial value problem

$$\dot{y} = f(t, y), \qquad y(t_0) = y_0, \tag{1.1}$$

where the dot denotes d/dt and y is a vector of length N. The ODE in (1.1) is assumed to be stiff, meaning that one or more strongly damped modes are present, and we will use the popular BDF (Backward Differentiation Formula) methods to solve it. These methods have the general form

$$y_n = \sum_{j=1}^{q} \alpha_j y_{n-j} + h\beta_0 \dot{y}_n, \qquad \dot{y}_n = f(t_n, y_n), \tag{1.2}$$

where q is the method order. The BDF methods are implicit, and hence at each time step one must solve for y_n in an algebraic system

$$\begin{aligned} &y_n - h\beta_0 f(t_n, y_n) - a_n = 0, \\ &a_n \equiv \sum_{j=1}^{q} \alpha_j y_{n-j}, \qquad \beta_0 > 0. \end{aligned} \tag{1.3}$$

Actually, we will deal with an equivalent form of the problem (1.3) that is posed in terms of

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

The restated system is

$$F_n(x_n) \equiv x_n - hf(t_n, a_n + \beta_0 x_n) = 0. \tag{1.4}$$

Some form of Newton iteration is usually employed to solve (1.4) for x_n (or (1.3) for y_n), and this generates a sequence of linear systems to be solved at each time step. The coefficient matrix in each of these linear systems is some value of (or an approxima-

* Received by the editors May 24, 1984, and in revised form June 14, 1985. This work was performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under contract W-7405-Eng-48, and supported by the DOE Office of Energy Research, Applied Mathematical Sciences Research Program.

† Mathematics Department, University of Houston, Houston, Texas 77004.

‡ Computing and Mathematics Research Division, L-316, Lawrence Livermore National Laboratory, Livermore, California 94550.

tion to a value of)

$$F_n'(x) = I - h\beta_0 J(t_n, y) \qquad (y = a_n + \beta_0 x), \tag{1.5}$$

where $J(t, y)$ denotes $\partial f/\partial y$, the system Jacobian matrix. The solution of these linear systems often makes up most of the work involved in the integration of the ODE's. The most common approaches taken for the linear systems are direct methods, and for them most of the required core memory is for the storage of the Jacobian matrix and the decomposition factors of F_n'. Iterative methods have also long been used in BDF solvers [7], but until recently they have also required storage of the nonzero Jacobian elements.

We consider here what we feel is best called a *matrix-free* method for the solution of these linear systems. In [6], Gear and Saad proposed the use of a Krylov-subspace projection method known as the *Incomplete Orthogonalization Method*, or IOM. The IOM algorithm is an iterative method for the solution of a linear system, and as will be seen below, does not require the storage of the coefficient matrix in any form. Its usefulness relies heavily on the fact that the convergence of the sequence of approximations to the solution of the linear system is fastest in the dominant subspace (i.e., in those components corresponding to the eigenvalues in the outermost part of the spectrum of the coefficient matrix). This is a desirable feature in the context of solving stiff systems of ODE's by multistep or multivalue methods, since normally the residual associated with the predicted value of the solution in a time step (which is given by an explicit method) has its largest errors in the stiff components. These errors are then damped out in the first few Newton corrections since the IOM algorithm solves for these components the most accurately. See Saad [12], [13], and Gear and Saad [6] for a more detailed discussion of the convergence properties of the IOM algorithm.

In [6], Gear and Saad discuss the implementation of the IOM algorithm into the Newton iteration and present some preliminary results. They give several enlightening results concerning both the linear and nonlinear iterations, but no comprehensive analysis of the resulting Newton-like iteration scheme is presented. Here we view the combined Newton-IOM iteration as an *Inexact Newton Method*, a class of methods in which the linear system in the Newton iteration is solved only approximately. The advantage of this viewpoint is that it lends a theoretical base upon which to build the combined iteration, at the same time suggesting how that combination might be done. For a discussion of Inexact Newton Methods, see Dembo, Eisenstat, and Steihaug [4]. We prove a generalization of a theorem in [4], relaxing a condition there on the norm of the residuals in the approximate solutions of the linear systems, and a second theorem that concerns the iteration error. These results will provide a basis for the stopping condition in the IOM algorithm presented later.

In [3], Chan and Jackson consider the use of preconditioned Krylov subspace methods in ODE solvers. However, since all of the preconditioning techniques implemented there require the explicit use of the Jacobian matrix, those methods are not matrix-free. Also, their solver attempts to use the same Jacobian over several corrector iterations and several steps, and they view the modified Newton/Krylov combination as an *inexact chord-Newton method*. This differs from the Newton/IOM approach studied here. In a second paper [2], Chan and Jackson have investigated general matrix-free preconditioning techniques, and in particular one based on nonlinear SSOR, for general nonlinear algebraic systems, but we are aware of no testing of this idea in the stiff ODE setting as yet. In [10], Miranker and Chern have considered the use of the conjugate gradient algorithm in the solution of the model problem $dy/dt = Jy$ by BDF methods, where J is symmetric and the matrix $I - h\beta_0 J$ is positive definite.

The remainder of this paper is organized as follows: In § 2, we introduce the modified and inexact Newton methods, prove the above-mentioned results, and describe the scaling to be used in the problem. Then in § 3, we present the IOM algorithm, and a scaled version of it, denoted SIOM. The incorporation of the SIOM algorithm into the general purpose ODE solver LSODE [8], [9] is discussed in § 4, and test results on several problems are reported in § 5. Finally, in § 6 we conclude with a discussion of our results and some suggestions for future work.

2. Modified and inexact Newton methods. The nonlinear system (1.4) is often solved using a Newton-like iteration scheme. In this section we discuss two such schemes, namely modified Newton and the class of Inexact Newton methods. We then introduce a natural scaling into the above system.

Consider a general nonlinear system in R^N,

$$F(x)=0, \tag{2.1}$$

with solution x^*. If Newton's method is used to solve (2.1), one generates an initial guess $x(0)$ by some technique and then computes successive iterates by the scheme

$$\begin{aligned} Ps(m) &= -F(x(m)), \\ x(m+1) &= x(m)+s(m), \end{aligned} \tag{2.2}$$

where $P=F'(x(m))$ is the Newton matrix (F' denoting $\partial F/\partial x$).

When instead using *modified Newton* to solve (2.1), the matrix P in (2.2) is given initially as some approximation to $F'(x^*)$, but is then held fixed while computing the iterates $x(m)$. In the ODE context, one must solve a system of the form (2.1) at each time step, and so for efficiency the matrix P is usually also held fixed over several time steps. The system (2.2) is typically solved by performing an LU decomposition of P (at the time it is formed) and using that for all iterations (on all time steps) until a decision is made to reevaluate P.

The convergence analysis of modified Newton is well known (cf. Ortega and Rheinboldt [11]). Under appropriate conditions on P and F that guarantee the local convergence of the iterates $x(m)$ to x^*, when $P \neq F'(x^*)$, one has *linear* convergence in that as $m \to \infty$

$$\|x(m+1)-x^*\|/\|x(m)-x^*\| \to C,$$

where $0<C<1$ and $\|\cdot\|$ is any norm on R^N. If $P=F'(x^*)$, then one can show *superlinear* convergence, that is,

$$\|x(m+1)-x^*\|/\|x(m)-x^*\| \to 0.$$

In practice, it is appropriate to assume only linear convergence. Then, at the time $x(m+1)$ is computed, one can form an estimate

$$C_m = \|x(m+1)-x(m)\|/\|x(m)-x(m-1)\|$$

of the asymptotic rate constant C, and use these in subsequent convergence tests. Thus a stopping condition on $x(m)$ of the form

$$\|x(m)-x^*\| \leqq \varepsilon$$

will be satisfied approximately if the (verifiable) condition

$$\bar{C}\|x(m)-x(m-1)\| \leqq \varepsilon$$

holds, provided that $\bar{C}$ is a good approximation to $C/(1-C)$, or simply to C if C is

sufficiently small. LSODE uses this convergence acceleration idea (with suitable "fudge factors") in its stopping test for modified Newton iterations, and it is quite beneficial there in reducing the average number of iterations per step. We would also like to use the same idea in conjunction with Inexact Newton Method iterations, but this will require some further justification.

The particular class of ODE problems of interest here is such that most of the work required for the integration is in the linear algebra operations associated with (2.2). Furthermore, much (often most) of the core memory needed in the integration is used for the storage of the matrix P and its LU factorization. (The latter is usually overwritten on P.) For large problems in which the number of unknowns is on the order of several thousand or more, storage considerations may be prohibitive on many computers. Thus, for this class of problems, any method which can approximately solve the system in (2.2) at reasonable cost, and also reduce the core memory required, merits investigation. The *Incomplete Orthogonalization Method* (IOM) proposed by Gear and Saad [6] is such a method, which we will view here from a somewhat different perspective than that in [6].

The IOM algorithm itself is a method (described in more detail in the next section) for the approximate solution of a linear system of equations $Ax = b$ in R^N. The use of the IOM algorithm in the solution of the nonlinear system (2.1) by Newton's method gives rise to a method which is more properly viewed as an *Inexact Newton Method.* From Dembo, Eisenstat, and Steihaug [4], an Inexact Newton Method for (2.1) has the following general form:

Set $x(0) =$ an initial guess
For $m = 0, 1, 2, \cdots$ until convergence:
  Find (in some unspecified manner) a vector $s(m)$ satisfying

$$F'(x(m))s(m) = -F(x(m)) + r(m) \tag{2.3}$$

  Set $x(m+1) = x(m) + s(m)$.

The *residual* $r(m)$ represents the amount by which $s(m)$ fails to satisfy the Newton equation (2.2). It is not generally known in advance, being the result of some inner algorithm which produces only an approximate solution to (2.2). In order to guarantee convergence of the scheme, one must demand some auxiliary conditions on the residual $r(m)$. In [4], it is shown that if

$$\|r(m)\| \leqq \eta_m \|F(x(m))\|, \qquad m = 0, 1, 2, \cdots, \tag{2.4}$$

where $0 \leqq \eta_m \leqq \eta_{\max} < 1$, then the sequence of iterates $x(m)$ converges to a true solution of $F(x) = 0$ at least linearly, as long as the initial guess $x(0)$ is close enough. Here again, $\|\cdot\|$ is any norm on R^N.

For the present context, where actual convergence of the iterates is not necessary, and the cost of obtaining them is high, the condition (2.4) is overly restrictive. Thus it is of interest to find out how much one can relax (2.4) and still obtain enough accuracy in the approximate solution $x(m)$ to x^*. The following is a result along these lines.

THEOREM 2.1. *Let $F: R^N \to R^N$ be a mapping such that*

(2.5) *there exists x^* in R^N with $F(x^*) = 0$;*

(2.6) *F is continuously differentiable in a neighborhood of x^*; and*

(2.7) *$F'(x^*)$ is nonsingular.*

Let $x(m)$ be a sequence generated by an inexact Newton method: That is, $x(0)$ is an initial guess, and for $m = 0, 1, 2, \cdots$, we have

$$F'(x(m))s(m) = -F(x(m)) + r(m), \qquad x(m+1) = x(m) + s(m). \tag{2.8}$$

Then there are positive constants ε_0 and K (depending only on F and x^*) with the following property: For any positive numbers $\varepsilon \leqq \varepsilon_0$ and $\delta \leqq K\varepsilon$, whenever $\|x(0) - x^*\| \leqq \varepsilon$, and the residuals $r(m)$ in (2.8) satisfy $\|r(m)\| \leqq \delta$ for all m, then all the iterates $x(m)$ exist and satisfy $\|x(m) - x^*\| \leqq \varepsilon$, and

$$\limsup_{m \to \infty} \|x(m) - x^*\| \leqq \delta / K.$$

Proof. First define $\mu = \|F'(x^*)\|$ and $\lambda = \|F'(x^*)^{-1}\|$. Then

$$\lambda^{-1}\|x\| \leqq \|F'(x^*)x\| \leqq \mu\|x\| \quad \text{for all } x \text{ in } R^N. \tag{2.9}$$

Let $\gamma > 0$ be chosen such that

$$b \equiv 2\gamma\lambda(1 + \mu\gamma) < 1.$$

Next observe that by (2.5)-(2.7), there exists $\varepsilon_0 > 0$ so that $\|x - x^*\| \leqq \varepsilon_0$ implies

$$\|F'(x) - F'(x^*)\| \leqq \gamma, \tag{2.10}$$

$$\|F'(x)^{-1} - F'(x^*)^{-1}\| \leqq \gamma, \quad \text{and} \tag{2.11}$$

$$\|F(x) - F(x^*) - F'(x^*)(x - x^*)\| \leqq \gamma\|x - x^*\|. \tag{2.12}$$

Next, for any x with $\|x - x^*\| \leqq \varepsilon_0$, consider the vector given by

$$x^+ = x + s \quad \text{with } F'(x)s = -F(x) + r,$$

where nothing is specified as yet about the size of the residual r. ($F'(x)$ is nonsingular by (2.11).) We have the following identity:

$$\begin{aligned} F'(x^*)(x^+ - x^*) = \{I + F'(x^*)[F'(x)^{-1} - F'(x^*)^{-1}]\} \\ \cdot \{r + [F'(x) - F'(x^*)](x - x^*) - [F(x) - F(x^*) - F'(x^*)(x - x^*)]\}. \end{aligned}$$

Taking norms and using (2.10)-(2.12), we then obtain

$$\begin{aligned} \|F'(x^*)(x^+ - x^*)\| &\leqq [1 + \|F'(x^*)\|\,\|F'(x)^{-1} - F'(x^*)^{-1}\|] \\ &\quad \cdot [\|r\| + \|F'(x) - F'(x^*)\| \cdot \|x - x^*\| \\ &\quad + \|F(x) - F(x^*) - F'(x^*)(x - x^*)\|] \\ &\leqq (1 + \mu\gamma)(\|r\| + 2\gamma\|x - x^*\|), \end{aligned}$$

or using (2.9),

$$\|x^+ - x^*\| \leqq \lambda(1 + \mu\gamma)\|r\| + b\|x - x^*\|. \tag{2.13}$$

We now want to insure that when $\|x - x^*\| \leqq \varepsilon$ and $\|r\| \leqq K\varepsilon$ for some constant K, then $\|x^+ - x^*\| \leqq \varepsilon$ also. From (2.13), it is clear that we will achieve that end if

$$\lambda(1 + \mu\gamma)K\varepsilon + b\varepsilon \leqq \varepsilon,$$

and so we define K by

$$K = (1 - b)/[\lambda(1 + \mu\gamma)].$$

Now for any positive $\varepsilon \leqq \varepsilon_0$, and any $x(0)$ with $\|x(0) - x^*\| \leqq \varepsilon$, a sequence of inexact

Newton iterates $x(m)$ $(m=1,2,\cdots)$ satisfying (2.8) with $\|r(m)\| \leqq K\varepsilon$ is guaranteed to exist, and for each m, $\|x(m)-x^*\| \leqq \varepsilon$.

We can say more about the norms of the errors if we suppose further that $\|r(m)\| \leqq \delta$ for some constant δ with $\delta \leqq K\varepsilon$. Then, setting

$$a = \lambda(1+\mu\gamma)\delta,$$

(2.13) gives

$$\|x(m+1)-x^*\| \leqq a + b\|x(m)-x^*\|,$$

and by induction we obtain

$$\|x(m)-x^*\| \leqq a(1+b+b^2+\cdots b^{m-1}) + b^m\|x(0)-x^*\|.$$

Thus, since $b<1$,

$$\|x(m)-x^*\| \leqq a/(1-b) + b^m\varepsilon,$$

and finally

$$\limsup_{m\to\infty} \|x(m)-x^*\| \leqq a/(1-b) = \delta/K. \qquad \text{Q.E.D.}$$

In practice, it is of interest to know how large δ can be. If ε_1 is the tolerance for the final computed iterate $x(m)$, where $\varepsilon_1 \leqq \varepsilon_0$, then we must demand that $\delta/K \leqq \varepsilon_1$ in order to insure $\limsup \|x(m)-x^*\| \leqq \varepsilon_1$. This indicates we should choose

$$\delta \simeq K\varepsilon_1.$$

Therefore, we need to know the largest allowable value of K within the constraints of the theorem. From the proof,

$$K = K(\gamma) = (1-b)/[\lambda(1+\mu\gamma)] = 1/(\lambda+\gamma\lambda\mu) - 2\gamma,$$

where $\lambda = \|F'(x^*)^{-1}\|$ and $\mu = \|F'(x^*)\|$. Recall that γ is chosen so that $b<1$. Hence, $b = 2\gamma(\lambda+\gamma\lambda\mu) < 1$ when $0<\gamma<\gamma_0$, where γ_0 is the positive root of $2\gamma(\lambda+\gamma\lambda\mu)=1$. It is apparent that $K(\gamma)$ is monotone in γ for $0<\gamma<\gamma_0$, with

$$K \to 0 \text{ as } \gamma \to \gamma_0 \quad \text{and} \quad K \to 1/\lambda \text{ as } \gamma \to 0.$$

Thus we would expect that

$$\delta \simeq \varepsilon_1/\lambda = \varepsilon_1/\|F'(x^*)^{-1}\|$$

is an acceptable choice for δ.

In the context of a stiff ODE system, we have $F(x) = x - hf(t, a+\beta_0 x)$ (see (1.4)), so that

$$F'(x^*) = I - h\beta_0 J, \qquad J = f_y(t, a+\beta_0 x^*)$$

(where f_y denotes $\partial f/\partial y$). If we assume that the ODE is stable in the sense that all the eigenvalues of J have negative real part, then

$$\rho(F'(x^*)^{-1}) < 1$$

where $\rho(A)$ is the spectral radius of a matrix A. To the extent that $\rho(F'(x^*)^{-1})$ approximates $\|F'(x^*)^{-1}\|$, we have roughly that $\lambda<1$. Hence the choice of $\delta=\varepsilon_1$ seems reasonable.

It is somewhat harder to assure that we have a reliable but practical estimate of the error $\|x(m)-x^*\|$. In doing so, we want to make use of estimates of the convergence

rate constant (based on linear convergence), as is done for modified Newton. Here, however, it must be remembered that the iterates in Theorem 2.1 do not even converge in general. The next result says that under a stronger assumption on the residuals $r(m)$, not only do we have convergence, but the error can be reliably bounded in terms of iterate differences. Thus a stopping test based on linear convergence remains valid when convergence is superlinear. Following the proof, we argue heuristically that it is still valid under the weaker assumption $\|r(m)\| \leqq \delta$, provided that δ is chosen small enough.

THEOREM 2.2. *Let $F: R^N \to R^N$ be a mapping satisfying* (2.5), (2.7), *and*:

(2.14) *F' satisfies a Lipschitz condition in a neighborhood of x^*.*

Let $x(m)$ be a sequence generated by the inexact Newton method (2.8). *If for some $\eta < 1$ the residuals $r(m)$ satisfy*

$$\|r(m)\| \leqq \eta \|F(x(m))\|^2, \tag{2.15}$$

then there exists an $\varepsilon > 0$ such that whenever $\|x(0) - x^*\| \leqq \varepsilon$, the iterates $x(m)$ satisfy $x(m) \to x^*$ as $m \to \infty$. Furthermore, for any $\varepsilon' > 0$, we have, for all sufficiently large m,

$$\|x(m+1) - x^*\| \leqq C_m \|x(m+1) - x(m)\|(1+\varepsilon'),$$

where

$$C_m = \|x(m+1) - x(m)\| / \|x(m) - x(m-1)\|.$$

Proof. It follows from the results in Dembo et al. [4] or from a simple modification of the proof of Theorem 2.1 that the sequence of iterates $x(m)$ converges to x^* when $m \to \infty$. Dembo et al. [4] show that under these stronger conditions the sequence $x(m)$ actually converges to x^* quadratically.

Next, note that by the mean value theorem (see Ortega and Rheinboldt [11])

$$F(x(m)) = F(x(m)) - F(x^*) = A_m(x(m) - x^*), \tag{2.16}$$

where

$$A_m = \int_0^1 F'(x^* + t[x(m) - x^*])\, dt,$$

and

$$A_m - F'(x^*) = \int_0^1 [F'(x^* + t(x(m) - x^*)) - F'(x^*)]\, dt.$$

It follows from (2.14) that

$$\|A_m - F'(x^*)\| \leqq \alpha \|x(m) - x^*\| \tag{2.17}$$

for some constant α. This then implies that $A_m \to F'(x^*)$ as $m \to \infty$ and by the perturbation lemma ([11, p. 45]) A_m is nonsingular for all m large enough.

Denoting $F(x(m))$ by F_m and $F'(x(m))$ by F'_m, one has, from the definition of the inexact Newton method,

$$x(m+1) = x(m) - (F'_m)^{-1}[F_m - r(m)]. \tag{2.18}$$

Using (2.16) then gives

$$x(m+1) - x^* = [I - (F'_m)^{-1} A_m][x(m) - x^*] + (F'_m)^{-1} r(m). \tag{2.19}$$

Also, from (2.16) and (2.18) one has

$$x(m+1)-x(m)=-(F'_m)^{-1}[A_m(x(m)-x^*)-r(m)].$$

Multiplying this last equation by $B_m \equiv I - A_m^{-1}F'_m$ and using (2.19) gives

$$\begin{aligned} B_m[x(m+1)-x(m)] &= -B_m(F'_m)^{-1}[A_m(x(m)-x^*)-r(m)] \\ &= x(m+1)-x^*-A_m^{-1}r(m). \end{aligned}$$

Therefore,

$$x(m+1)-x^* = B_m[x(m+1)-x(m)]+A_m^{-1}r(m). \tag{2.20}$$

We next show that

$$\|B_m\| \leqq C_m \text{ for } m \text{ large enough.} \tag{2.21}$$

Consider

$$\begin{aligned} C_m - \|B_m\| &= \|x(m+1)-x(m)\|/\|x(m)-x(m-1)\| - \|B_m\| \\ &= \|(F'_m)^{-1}[F_m - r(m))]\|/\|x(m)-x(m-1)\| - \|B_m\| \\ &\geqq \|F'_m\|^{-1}\|F_m - r(m)\|/\|x(m)-x(m-1)\| - \|B_m\| \\ &= \frac{\|F_m - r(m)\|}{\|x(m)-x(m-1)\|}\left[\|F'_m\|^{-1} - \|x(m)-x(m-1)\|\frac{\|B_m\|}{\|F_m - r(m)\|}\right]. \end{aligned}$$

Now, if we can show that $\|B_m\|/\|F_m - r(m)\|$ remains bounded as $m \to \infty$, then since $\|F'_m\| \to \|F'(x^*)\| > 0$, (2.21) will follow. We have

$$\|F_m\| \leqq \|F_m - r(m)\| + \|r(m)\| \leqq \|F_m - r(m)\| + \eta\|F_m\|^2$$

by (2.15). Thus,

$$\begin{aligned} \|F_m - r(m)\| &\geqq (1-\eta\|F_m\|)\|F_m\| \\ &\geqq (1-\eta\|F_m\|)\|A_m^{-1}\|^{-1}\|x(m)-x^*\|, \end{aligned} \tag{2.22}$$

using (2.16). Also, for m large enough,

$$\|B_m\| \leqq \|A_m^{-1}\|\,\|A_m - F'_m\| \leqq \|A_m^{-1}\|\alpha\|x(m)-x^*\|, \tag{2.23}$$

by using the Lipschitz conditon in a way similar to that for (2.17). Using (2.22) and (2.23), we see that

$$\|B_m\|/\|F_m - r(m)\| \leqq \alpha\|A_m^{-1}\|^2/(1-\eta\|F_m\|),$$

and this is bounded as $m \to \infty$. This establishes (2.21).

From (2.20) and (2.21), we have that for m large,

$$\|x(m+1)-x^*\| \leqq C_m\|x(m+1)-x(m)\|\left[1+\frac{\|A_m^{-1}r(m)\|\cdot\|x(m)-x(m-1)\|}{\|x(m+1)-x(m)\|^2}\right]. \tag{2.24}$$

We now show that $\|A_m^{-1}r(m)\|/\|x(m+1)-x(m)\|^2$ remains bounded as $m \to \infty$. From (2.18) and (2.22),

$$\|x(m+1)-x(m)\| \geqq \|F'_m\|^{-1}\|F_m - r(m)\| \geqq \|F'_m\|^{-1}(1-\eta\|F_m\|)\|F_m\|. \tag{2.25}$$

Thus, by (2.15) and (2.25),

$$\|A_m^{-1}r(m)\|/\|x(m+1)-x(m)\|^2 \leqq \|A_m^{-1}\|\eta\|F'_m\|^2(1-\eta\|F_m\|)^{-2},$$

and this remains bounded as $m \to \infty$. Therefore, given any $\varepsilon' > 0$ we can choose m large enough so that in (2.24)

$$\|x(m+1) - x^*\| \leqq C_m \|x(m+1) - x(m)\|(1+\varepsilon'). \qquad \text{Q.E.D.}$$

Since we are only requiring that $\|r(m)\| \leqq \delta$ in practice, it would be helpful to know if a result similar to Theorem 2.2 holds in this case. But since we do not have actual convergence of the iterates $x(m)$ in Theorem 2.1, the best we can do is argue heuristically. If instead of (2.15), we only assume $\|r(m)\| \leqq \delta$, then in (2.22)

$$\|F_m - r(m)\| \geqq \|F_m\| - \delta,$$

and if $\delta < \|F_m\|$, then

$$1/\|F_m - r(m)\| \leqq 1/(\|F_m\| - \delta) \leqq 1/(\|A_m^{-1}\|^{-1} e_m - \delta), \tag{2.26}$$

where $e_m = \|x(m) - x^*\|$. Then

$$C_m - \|B_m\| \geqq \frac{\|F_m - r(m)\|}{\|x(m) - x(m-1)\|} \left[\|F'_m\|^{-1} - \frac{\|x(m) - x(m-1)\| \, \|B_m\|}{\|A_m^{-1}\|^{-1} e_m - \delta} \right].$$

Now, recalling that $\lambda = \|F'(x^*)^{-1}\|$, we have

$$\frac{\|B_m\| \, \|x(m) - x(m-1)\|}{\|A_m^{-1}\|^{-1} e_m - \delta} \leqq \frac{\|A_m^{-1}\| \alpha e_m (e_m + e_{m-1})}{\|A_m^{-1}\|^{-1} e_m - \delta} \simeq \frac{\lambda \alpha e_m (e_m + e_{m-1})}{\lambda^{-1} e_m - \delta}.$$

Since $\|F'_m\|^{-1} \simeq \mu^{-1}$, we need δ chosen small enough that

$$\mu^{-1} - [\lambda \alpha e_m (e_m + e_{m-1})]/[\lambda^{-1} e_m - \delta] \geqq 0.$$

Hence, to get $\|B_m\| \leqq C_m$, δ must be chosen so that

$$\delta \leqq e_m [1/\lambda - \mu \lambda \alpha (e_m + e_{m-1})].$$

Note that e_m/λ is an approximate lower bound for $\|F_m\|$, and so the above inequality implies that $\delta < \|F_m\|$ (approximately). Now, if ε_1 is the desired tolerance for $\|x(m) - x^*\|$ and ε is chosen as in Theorem 2.1, then we can expect to have $\varepsilon_1 \leqq e_m \leqq \varepsilon$ for almost all m that actually occur. Thus δ should be chosen so that

$$\delta \leqq \varepsilon_1 (1/\lambda - 2\mu \lambda \alpha \varepsilon),$$

and ε needs to be small enough so that $1/\lambda - 2\mu\lambda\alpha\varepsilon > 0$.

Next, from the first part of (2.25) and from (2.26), one obtains

$$1/\|x(m+1) - x(m)\| \leqq \|F'_m\|/(\|A_m^{-1}\|^{-1} e_m - \delta).$$

Hence

$$\frac{\|A_m^{-1} r(m)\|}{\|x(m+1) - x(m)\|^2} \|x(m) - x(m-1)\| \leqq \frac{\|F'_m\|^2 \|A_m^{-1}\| \delta (e_m + e_{m-1})}{(\|A_m^{-1}\|^{-1} e_m - \delta)^2}$$

$$\simeq \frac{\mu^2 \lambda \delta (e_m + e_{m-1})}{(e_m/\lambda - \delta)^2} \leqq \frac{2\varepsilon \mu^2 \lambda \delta}{(\varepsilon_1/\lambda - \delta)^2}.$$

For this last quantity to be less than a given ε', δ must satisfy

$$\delta^2 - 2\delta(\varepsilon_1/\lambda + \mu^2 \lambda \varepsilon/\varepsilon') + \varepsilon_1^2/\lambda^2 \geqq 0.$$

Again, this will be true for δ small enough. Therefore, when δ is chosen small enough

in comparison to ε_1 we have, approximately,

$$\|x(m+1)-x^*\| \leqq C_m \|x(m+1)-x(m)\|.$$

The sizes of ε_1 and δ used will be seen below and are related to the norm used and the ODE context. However, it is very likely the case that δ will need to be much smaller than ε_1/λ.

We close this section with a discussion of *scaling* in the problem (2.1), which will at the same time specify the norm to be used. Suppose that a diagonal matrix D is given, with positive diagonal elements, and that the vector $D^{-1}x$ is to be considered well-scaled whenever x approximates x^*. That is, we will give the components of $D^{-1}x$ equal weight when measuring the size of x (or errors in x, etc.). This means that the L_2 (Euclidean) norm $\| \ \|_2$ on $D^{-1}x$, or any multiple of this, is an appropriate choice of norm. In the LSODE solver these scale factors are available in a natural way in terms of the user-defined absolute and relative tolerances ATOL and RTOL. The weight associated with component y^i of y during the nth time step is

$$w_i = \mathrm{RTOL}_i \, |y^i_{n-1}| + \mathrm{ATOL}_i$$

in terms of the last accepted numerical solution value y_{n-1}. However, in order not to introduce a bias when dealing with similar ODE systems of differing sizes, we use the root-mean-square (RMS) norm instead of the L_2 norm, the two being related by

$$\|x\|_{RMS} = \|x\|_2/\sqrt{N}.$$

Thus, for a vector y approximating y_n, or a vector x approximating $h\dot{y}_n$, we will use the *weighted RMS norm*,

$$\|x\|_{\mathrm{WRMS}} = \|D^{-1}x\|_2, \tag{2.27}$$

where

$$D = \mathrm{diag}\,(d_1, \cdots, d_N), \qquad d_i = w_i\sqrt{N}.$$

The given problem $F(x)=0$ can now be restated in a scaled form in which the appropriate norm is Euclidean. For any given x we define $\tilde{x} = D^{-1}x$, and define a new function

$$\tilde{F}(\tilde{x}) = D^{-1}F(x) = D^{-1}F(D\tilde{x}). \tag{2.28}$$

If an Inexact Newton Method is applied to $\tilde{F}(\tilde{x})=0$, the iterates $\tilde{x}(m)$ satisfy

$$\tilde{F}'(\tilde{x}(m))\tilde{s}(m) = -\tilde{F}(\tilde{x}(m)) + \tilde{r}(m), \qquad \tilde{x}(m+1) = \tilde{x}(m) + \tilde{s}(m), \tag{2.29}$$

where $\tilde{F}'(x) = D^{-1}F'(x)D$. If one defines $s(m) = D\tilde{s}(m)$ and $r(m) = D\tilde{r}(m)$, then clearly (2.29) is equivalent to (2.3). We do not explicitly rescale the vectors in the nonlinear iteration, but will perform the scaling implicitly. However, this is not the case for the associated linear algebraic system problem, as will be seen in the next section, where the IOM method is given in an explicitly scaled form, called SIOM.

3. The incomplete orthogonalization method. The Incomplete Orthogonalization Method (IOM) given in [13] is an algorithm for the approximate solution of the linear system

$$Ax = b \tag{3.1}$$

where A is an $N \times N$ matrix and x and b are N-vectors. Here we are interested in IOM only as a means of performing the general step of an Inexact Newton Method

for a nonlinear problem $F(x)=0$. The vector b represents $-F(x(m))$, A represents $F'(x(m))$, and the solution vector x represents the increment $s(m)=x(m+1)-x(m)$ giving the next Newton iterate $x(m+1)$. (The letter x is used to denote solutions of both nonlinear and linear systems; the particular meaning should be clear to the reader from the context, however.) We give here a brief development of IOM in the original (Arnoldi) form, the IOM form, and the scaled IOM form. Details on the Arnoldi and (unscaled) IOM algorithms and related methods can be found in [13].

By a *projection method* on the subspace $K_l=\text{span}\{V_l\}$, where $V_l=[v_1,\cdots,v_l]$ is an orthonormal system in R^N, we mean a method which finds an approximate solution x_l of (3.1), using an initial guess x_0, by requiring that

$$x_l-x_0\in K_l, \qquad (Ax_l-b)\perp v_j (j=1,2,\cdots,l). \tag{3.2}$$

Here, orthogonality is meant in the ordinary Euclidean sense. Different choices of the subspace give rise to different projection methods, as do other choices of orthogonality conditions. If we let r_0 be the initial residual $r_0=b-Ax_0$, and set K_l equal to the *Krylov subspace*

$$K_l=\text{span}\{r_0, Ar_0,\cdots,A^{l-1}r_0\},$$

then we have a *Krylov subspace projection method.* Writing $x=x_0+z$, the equivalent equation for z is

$$Az=r_0, \tag{3.3}$$

and the method finds an approximate solution z_l of (3.3) by requiring that

$$z_l\in K_l, \qquad (Az_l-r_0)\perp v_j \qquad (j=1,\cdots,l),$$

where $V_l=[v_1,\cdots,v_l]$ is an orthonormal basis of K_l. Letting V_l also denote the $N\times l$ matrix with columns v_i, and letting $z_l=V_ly_l$ with $y_l\in R^l$, we see immediately that y_l must be the solution of the linear system

$$V_l^TAV_ly_l-V_l^Tr_0=0,$$

and so $x_l=x_0+z_l$ becomes

$$x_l=x_0+V_l(V_l^TAV_l)^{-1}V_l^Tr_0. \tag{3.4}$$

It is assumed throughout that the vectors $r_0, Ar_0,\cdots,A^{l-1}r_0$ are linearly independent so that the dimension of K_l is l.

We next present an algorithm given by Saad [13], which is an adaptation of an earlier one due to Arnoldi [1]. It constructs an orthonormal basis $V_l=[v_1,\cdots,v_l]$ of K_l such that $V_l^TAV_l$ has Hessenberg form:

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

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

$$h_{j+1,j}=\|w_{j+1}\|_2,$$

$$v_{j+1}=w_{j+1}/h_{j+1,j}.$$

Here $(\cdot,\cdot)$ is the Euclidean inner product and $\|\cdot\|_2$ the Euclidean norm. Saad (cf. [13]) has shown that $[v_1,\cdots,v_l]$ is an orthonormal basis for K_l and that the matrix $V_l^TAV_l$ is the upper Hessenberg matrix H_l whose nonzero elements are the h_{ij} defined

in the above algorithm. It then follows that the vector $V_l^T r_0$ in (3.4) is equal to $\beta V_l^T v_1 = \beta e_1$, where $\beta = \|r_0\|_2$, and $e_1 = (1, 0, \cdots, 0)^T \in R^l$. Therefore, the approximation x_l is given by

$$x_l = x_0 + \beta V_l H_l^{-1} e_1.$$

A very important practical consideration is the choice of l, which amounts to a stopping criterion. A very useful identity for this is the following equation for the residual norm:

$$\|b - Ax_l\|_2 = h_{l+1,l} |e_l^T y_l|. \tag{3.5}$$

The relation (3.5) follows from the relation

$$AV_l = V_l H_l + h_{l+1,l} v_{l+1} e_l^T,$$

which can be derived from the algorithm. An interesting feature of the relation (3.5) is that one does not have to form x_l or y_l in order to compute $\|b - Ax_l\|_2$. If we perform an LU factorization of H_l, writing $H_l = LU$, and assume that no pivoting was necessary, then it can be shown [13] that

$$h_{l+1,l} |e_l^T y_l| = h_{l+1,l} \beta \left| \left(\prod_{i=1}^{l-1} l_i \right) \Big/ u_{ll} \right|,$$

where the l_i $(i = 1, \cdots, l-1)$ are the successive multipliers (subdiagonal elements of L). In general, one can show that when *no* pivoting has been necessary for $i \in I$, where $I \subset \{1, \cdots, l-1\}$, then

$$h_{l+1,l} |e_l^T y_l| = h_{l+1,l} \beta \left| \left(\prod_{i \in I} l_i \right) \Big/ u_{ll} \right|. \tag{3.6}$$

See [13] for more details.

The use of (3.5) to estimate the error $\|b - Ax_l\|_2$ then leads to the following algorithm, in which $l_{\max}$ and δ are given parameters.

ALGORITHM 3.1 (Arnoldi's Algorithm).

1. Compute $r_0 = b - Ax_0$ and set $v_1 = r_0 / \|r_0\|_2$.
2. For $l = 1, 2, \cdots, l_{\max}$ do:
   (a) $w_{l+1} = Av_l - \sum_{i=1}^{l} h_{il} v_i$, $h_{il} = (Av_l, v_i)$,
   $h_{l+1,l} = \|w_{l+1}\|_2$,
   $v_{l+1} = w_{l+1} / h_{l+1,l}$.
   (b) Update the LU factorization of H_l.
   (c) Use (3.6) to compute $\rho_l = h_{l+1,l} |e_l^T y_l| = \|b - Ax_l\|_2$.
   (d) If $\rho_l < \delta$, go to Step 3. Otherwise, go to (a).
3. Compute $x_l = x_0 + \|r_0\|_2 V_l H_l^{-1} e_1$ and stop.

In the above algorithm, if the test on ρ_l fails, and if $l = l_{\max}$ iterations have been performed, then one has the option of either accepting the final approximation x_l, or setting $x_0 = x_l$ and then going back to step 1 of the algorithm. This last procedure has the effect of "restarting" the algorithm. We also note that due to the upper Hessenberg form of H_l, there is a convenient way to perform an LU factorizztion of H_l by using the LU factors of H_{l-1} $(l > 1)$.

In Algorithm 3.1, as l gets large, a considerable amount of the work involved is in making the vector v_{l+1} orthogonal to all the previous vectors $v_1, \cdots, v_l$. Saad [13] has proposed a modification of Algorithm 3.1 in which the vector v_{l+1} is only required to be orthogonal to the previous p vectors, $v_{l-p+1}, \cdots, v_l$. Saad [13] has shown that equations (3.5) and (3.6) still hold in this case.[1] This leads to an algorithm called the *Incomplete Orthogonalization Method*, denoted by IOM. It differs from Algorithm 3.1 only in that the sum in Step 2(a) begins at $i = i_0$ instead of at $i = 1$, where $i_0 = \max(1, l-p+1)$. The remarks made after Algorithm 3.1 are also applicable to IOM. In [13], Saad compares the two algorithms on several test problems, and reports that IOM is sometimes preferred, based on total work required and run times.

When A is symmetric, the inner products $h_{i,l}$ theoretically vanish for $i < l-1$, so that one can take $p = 2$. If A is also positive definite, then IOM with $p = 2$ is equivalent to the conjugate gradient method [13, § 3.3.1, Remark 4]. Thus a value of p less than $l_{\max}$ might be expected to be cost-effective when A is nearly symmetric.

For our purposes, the use of the Euclidean norm is overly restrictive, and we use *scaled* versions of these algorithms, corresponding to the scaled form of the problem given in § 2. Suppose that instead of the linear system $Ax = b$, we want to solve an equivalent problem that is better scaled, namely

$$\tilde{A}\tilde{x} = \tilde{b}, \quad \text{where } \tilde{A} = D^{-1}AD, \quad \tilde{x} = D^{-1}x, \quad \tilde{b} = D^{-1}b,$$

and where D is a diagonal scaling matrix. Recall from (2.27) that the weighted *RMS* norm of interest is given by $\|x\|_{\text{WRMS}} = \|D^{-1}x\|_2 = \|\tilde{x}\|_2$, by virtue of the choice of D in terms of tolerance parameters. Applying IOM to this scaled problem then gives the following algorithm, called *Scaled* IOM, or SIOM:

ALGORITHM 3.2 (Scaled IOM).

1. Compute $r_0 = b - Ax_0$, set $\tilde{r}_0 = D^{-1}r_0$, and compute $\|\tilde{r}_0\|_2$. Then set $\tilde{v}_1 = \tilde{r}_0 / \|\tilde{r}_0\|_2$.
2. For $l = 1, 2, \cdots, l_{\max}$ do:
   (a) Compute $\tilde{A}\tilde{v}_l = D^{-1}(A(D\tilde{v}_l))$.
   (b) $\tilde{w}_{l+1} = \tilde{A}\tilde{v}_l - \sum_{i=i_0}^{l} \tilde{h}_{il}\tilde{v}_i$ where $i_0 = \max(1, l-p+1)$ and $\tilde{h}_{il} = (\tilde{A}\tilde{v}_l, \tilde{v}_i)$.
   (c) Set $\tilde{h}_{l+1,l} = \|\tilde{w}_{l+1}\|_2$, $\tilde{v}_{l+1} = \tilde{w}_{l+1} / \tilde{h}_{l+1,l}$.
   (d) Update the LU factorization of $\tilde{H}_l = (\tilde{h}_{ij})$.
   (e) Use (3.6) to compute $\rho_l = \|\tilde{b} - \tilde{A}\tilde{x}_l\|_2 = \|b - Ax_l\|_{\text{WRMS}}$.
   (f) If $\rho_l < \delta$, go to Step 3. Otherwise, go to (a).
3. Compute $\tilde{z}_l = \|\tilde{r}_0\|_2 \tilde{V}_l \tilde{H}_l^{-1} e_1$, set $x_l = x_0 + D\tilde{z}_l$, and stop.

This algorithm basically performs an explicit scaling of the problem $Ax = b$ without actually scaling the matrix A. Alternatively, this can be viewed as the IOM algorithm with a more general inner product, which in this case is given by $(u, v)_{\text{WRMS}} = (D^{-1}u, D^{-1}v)$. In step 2(a), the parentheses indicate that the product $\tilde{A}\tilde{v}_l$ is computed by first computing $D\tilde{v}_l$, then multiplying by A, and finally scaling by D^{-1}. In the present context, the matrix A is never formed explicitly; only the action of A times a vector v is needed. Hence an explicit scaling of A has no meaning here. The matrix $\tilde{H}_l$ is again upper Hessenberg, with its nonzero elements given by the $\tilde{h}_{ij}$, and the matrix $\tilde{V}_l$ is given by $\tilde{V}_l = [\tilde{v}_1, \cdots, \tilde{v}_l]$. As indicated earlier, the vector $\tilde{H}_l^{-1}e_1$ is computed by generating an LU decomposition of $\tilde{H}_l$ (by successive updating as l varies), followed by back-substitution. Here $\tilde{H}_l$ is treated as a *general* Hessenberg matrix, even though

[1] We note that in [13] there is a typographical error in the line following equation (3.15), where $\hat{y}_m$ should be replaced by $\hat{y}_m$.

it is actually banded (with an upper half-bandwidth of $p-1$), because its size is too small to gain any efficiency advantage from the band structure.

One could alternatively implement an implicitly scaled version of IOM, which, as noted above, could be viewed as the IOM algorithm with a more general inner product. The $\tilde{h}_{il}$ can then be defined by $\tilde{h}_{il} = (D^{-1}Av_l, D^{-1}v_i)$ and $\tilde{h}_{l+1,l} = \|D^{-1}w_{l+1}\|_2 = \|w_{l+1}\|_{\text{WRMS}}$. At first glance, this algorithm may appear to be more costly due to the additional vector scalings required. However, if D^{-2} is formed and saved before the IOM iteration, and $D^{-2}Av_l$ is formed for each l, then the $\tilde{h}_{il}$ can be computed by $\tilde{h}_{il} = (D^{-2}Av_l, v_i)$ and the norms $\|\cdot\|_{\text{WRMS}}$ computed similarly, and the scalings by D^{-1} are not necessary. A careful operation count (assuming $D^{-2}Av_l$ is saved in an available work array, and D and D^2 share the same storage) indicates that this implictly scaled IOM is just as efficient as Algorithm 3.2 above, except for the cost of recovering D from D^2. This last operation is necessary in our setting, since D is needed for use in the ODE solver after the call to IOM. A third choice, namely not scaling at all in IOM but using a stopping test on the WRMS norm of the residual, is impractical because it would require the direct calculation of the residual and its norm.

In the setting of a stiff ODE system, the Newton iteration begins with an explicit prediction $y_n(0)$, and a corresponding prediction $x(0) = (y_n(0) - a_n)/\beta_0$ of $h\dot{y}_n$. Thus, the first linear system to be solved on the nth time step is $Ax = b$ with

$$b = -F(x(0)) = hf(t_n, y_n(0))) - x(0),$$
$$A = F'(x(0)) = I - h\beta_0 J(t_n, y_n(0)).$$

The stiffness of the problem can be expected to make b largest in the stiff components (i.e., in the subspace corresponding to the stiff eigenvalues). It is crucial that these components be damped out in the corrector iteration. Results of Gear and Saad [6], [12] suggest (but in general do not prove) that the convergence of the Arnoldi iterates is fastest in the subspace corresponding to the outermost parts of the spectrum, which would include the stiff components. To the extent that this is true, it not only helps explain the success of the method, but also suggests that a fairly small value of $l_{\max}$ may suffice. However, it is also important for IOM to have reasonably good convergence properties over the rest of the spectrum as well, and our tests suggest that it does, within some limitations.

4. Algorithmic implementation. The LSODE solver [8], [9] has been modified to solve the nonlinear system (1.4) by an Inexact Newton Method in which the scaled IOM algorithm is used to solve the resulting linear systems (2.2). We denote the iterates approximating $x_n = h\dot{y}_n$ by $x_n(m)$, and the corresponding approximations to y_n by $y_n(m)$. These two vectors are related by

$$y_n(m) = a_n + \beta_0 x_n(m) \tag{4.1}$$

for every $m = 0, 1, \cdots$. From (1.4), recall that the function whose zero we seek is

$$F_n(x_n) = x_n - hf(t_n, a_n + \beta_0 x_n) = x_n - hf(t_n, y_n), \tag{4.2}$$

and the equations for the (true) Newton correction are

$$F_n'(x_n(m))s_n(m) = -F(x_n(m)),$$

or

$$\begin{aligned} &[I - h\beta_0 J(t_n, y_n(m))]s_n(m) = hf(t_n, y_n(m)) - x_n(m), \\ &x_n(m+1) = x_n(m) + s_n(m). \end{aligned} \tag{4.3}$$

In order to describe precisely the combined algorithm used, we must first outline the structure and overall algorithm of LSODE, to the extent that this is relevant here.

4.1. The unmodified algorithm. Aside from several auxiliary routines of secondary importance, the structure of LSODE (unmodified) is shown in Fig. 1, with the dashed line connections ignored. Subroutine LSODE is a driver, and subroutine STODE performs a single step and associated error control. STODE calls PREPJ to evaluate and do an LU factorization of the matrix P which approximates $I - h\beta_0 J$ (see (4.3)), and subsequently calls SOLSY to solve the linear system (2.2). Both of these routines call LINPACK routines [5] to do the matrix operations.

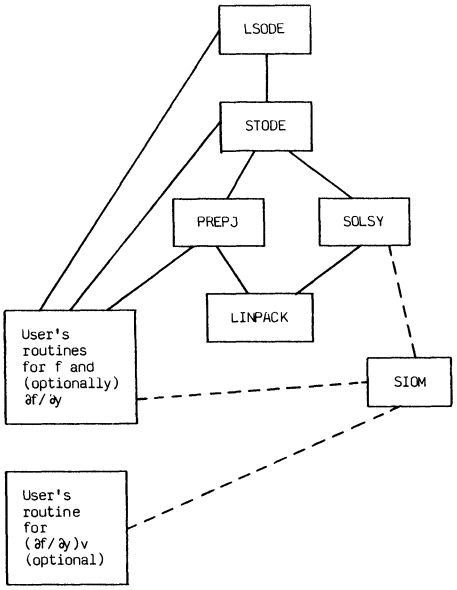

FIG. 1. *Simplified overall structure of the* LSODE *package.*

Within STODE, the basic algorithm for time step n, in its unmodified form, is as follows:

(1) Set flag showing whether to reevaluate J.
(2) Predict y_n as $y_n(0)$, and $h\dot{y}_n$ as $x_n(0)$.
(3) Compute $f(t_n, y_n(0))$; set $m = 0$.
(4) Call PREPJ if flag is on.
(5) Form $F_n(x_n(m))$.
(6) Call SOLSY and correct to get $x_n(m+1)$ and $y_n(m+1)$.
(7) Update estimate of convergence rate constant C, if $m \geqq 1$.
(8) Test for convergence.
(9) If convergence test failed:
  (a) Set $m \leftarrow m+1$.
  (b) If $m < 3$, compute $f(t_n, y_n(m))$ and go to Step (5).
  (c) If $m = 3$ and J is current, set $h \leftarrow h/4$ and go to Step (1) (redo time step).
  (d) If $m = 3$ and J is not current, set flag to reevaluate J and go to Step (3) (redo time step).
(10) If the convergence test passed, update history, do error test, etc.

In algorithm step (1) above, the decision is made to reevaluate J (and redo the LU factorization of $P = I - h\beta_0 J$) if either

(a) 20 steps have been taken since the last evaluation of J, or

(b) the value of $h\beta_0$ has changed by more than 30% since J was last evaluated.

In algorithm step (7), the iterate difference $s_n(m) = x_n(m+1) - x_n(m)$ is used, together with $s_n(m-1)$ if $m \geqq 1$, to form the ratio $C_m = \|s_n(m)\| / \|s_n(m-1)\|$, and C is updated to be the larger of $.2C$ and C_m. Whenever J is evaluated, C is reset to 0.7. The norm here is the weighted RMS norm given in § 2. The convergence test in step (8) requires the product $\|s_n(m)\| \min(1, 1.5C)$ to be less than a constant which depends only on q (the order of the BDF method). This is based on linear convergence, as discussed in § 2. Algorithm step (10) includes step and order selection for the next step (if the error test passed) or for redoing the current step (if it failed), but the details of that are not relevant here.

4.2. The modified algorithm. In the modified algorithm, subroutine SOLSY calls a module denoted SIOM (consisting of subroutine SIOM and auxiliary routines), which performs the solution of the linear system (4.3) using the scaled IOM algorithm given in Algorithm 3.2. Subroutine SIOM then calls user-supplied routines (mainly for evaluating f), as indicated by the dashed lines in Fig. 1. It also performs the looping and convergence test for the SIOM iterations. We describe below the essential features of our implementation of the SIOM algorithm in the LSODE solver.

We have chosen to implement Algorithm 3.2 without the restart feature proposed in [6]. This was done primarily for reasons of simplicity. Thus, if the convergence test $\rho_l < \delta$ in SIOM has not yet passed for $l = l_{\max}$, no further attempt is made to solve the current linear system. However, the Newton iteration is not necessarily considered hopeless, depending on the final value of ρ_l ($l = l_{\max}$) and the number m of Newton iterations already performed. If on the first Newton correction ($m = 0$) the final ρ_l is either $\leqq 1$ or $\leqq \|\tilde{b}\|_2 = \|F_n(x_n(0))\|_{\text{WRMS}}$, or if $m \geqq 1$ and the final ρ_l is $\leqq 1$, then the solution vector x_l is computed and returned, and the Newton iteration is continued (if possible). Otherwise, a flag is passed back to STODE causing it to declare a Newton corrector convergence failure, reduce the step size h, and attempt the step again. As noted by Gear and Saad [6, § 2.1], the use of repeated Newton iterations makes it unnecessary to do restarting within the IOM algorithm. In fact, when $f(t, y)$ is linear in y and $x_0 = 0$, the combined Newton-IOM iteration is identical to IOM with restarts.

We note also that a convergence failure in SIOM is likely to occur when the dominant subspace in the linear system (4.3) has too large a dimension for the size of $l_{\max}$. Reducing h then effectively reduces the size of this subspace until SIOM can achieve convergence.

We have tailored the SIOM algorithm to the ODE context by taking advantage of the relation $A = I - h\beta_0 J$. Thus in Algorithm 3.2, A is replaced by J and then the required quantities are constructed. Thus in step 2(a) we compute $\tilde{J}\tilde{v}_l = D^{-1}(J(D\tilde{v}_l))$, and in steps 2(b) and 2(c) we compute quantities

$$g_{il} = (\tilde{J}\tilde{v}_l, \tilde{v}_i),$$

$$u_{l+1} = \tilde{J}\tilde{v}_l - \sum_{i=i_0}^{l} g_{il}\tilde{v}_i,$$

$$g_{l+1,l} = \|u_{l+1}\|_2,$$

$$\tilde{v}_{l+1} = u_{l+1} / g_{l+1,l},$$

$$\tilde{h}_{il} = \delta_{il} - h\beta_0 g_{il} \qquad (i_0 \leqq i \leqq l+1),$$

where δ_{ij} is the Kronecker delta. The only other change in the equations is to use $|\tilde{h}_{l+1,l}| = |h\beta_0 g_{l+1,l}|$ in place of $\tilde{h}_{l+1,l}$ in computing ρ_l. Note that the two forms of the algorithm give the same Krylov subspace in exact arithmetic. Also, the subspace based on J appears to be more accurate in the presence of rounding error when h is small, because of the absence of the cancellation $(\tilde{v}_l - \tilde{v}_l)$ in computing u_{l+1}. Furthermore, since we are not explicitly forming A, computing Jv instead of Av saves N multiply/subtract operations.

As in other situations where Gram-Schmidt orthogonalization is done, there is a potential loss of accuracy of the vectors obtained in SIOM, due to numerical cancellation errors. In order to avoid this loss, we insert a correction after the calculation of u_{l+1}. In SIOM we compute the inner products

$$\varepsilon_i = (u_{l+1}, \tilde{v}_i)$$

(which would vanish in the absence of roundoff error). Then for each i $(i_0 \leqq i \leqq l)$, if

$$|h\beta_0 \varepsilon_i| > 10^3 \text{ (unit roundoff) } |\tilde{h}_{il}|,$$

then we correct $\tilde{h}_{il}$ by $-\varepsilon_i h\beta_0$ and correct u_{l+1} by $-\varepsilon_i \tilde{v}_i$. This enhances the numerical orthonormality of the $\tilde{v}_i$ if it was lacking to a significant degree.

The values of $l_{\max}$ (the maximum Krylov subspace dimension) and p (the number of vectors to which $\tilde{v}_{l+1}$ is made orthogonal) are optionally set by the user of the modified solver. The default values we have set are

$$l_{\max} = 5 \quad \text{and} \quad p = l_{\max}.$$

This means that, for the default values, we are actually doing a scaled Arnoldi iteration rather than a properly incomplete IOM. However, in cases where the problem Jacobian is known to be symmetric or nearly symmetric, the use of $p = 2$ (or other value $< l_{\max}$) should prove valuable.

The fairly small value of $l_{\max}$ is based on the desire to keep the storage requirements of the method to a minimum. The dominant part of the storage for SIOM is $(l_{\max}+2)N + (l_{\max})^2$ words. Hence to assure that the method will be competitive in storage with more traditional methods for the kinds of problems anticipated, where sparsity would be strongly exploited, we wish to limit $l_{\max}$ severely.

Next, we need to consider the choice of the starting vector x_0 in Algorithm 3.2. In Gear and Saad [6], a nonzero value of x_0 is discussed. However, for reasons of simplicity and efficiency, we take $x_0 = 0$. This means that the convergence test $\|b - Ax_l\|_{\text{WRMS}} \leqq \delta$ is easily applied also for $l = 0$. This test is made in step 1 of Algorithm 3.2, and if it passes, $x_0 = 0$ is accepted as the approximate solution.

In [3], Chan and Jackson used $x_0 = b (= -F_n(x_n(m)))$ as an alternative initial guess for the iterative solution of (4.3), and also suggested that using $x_0 = 0$ may have a deleterious effect upon the stepsize and order selection strategies of the ODE solver. However, we found no evidence of such an effect in our testing. In addition, while for small h using $x_0 = b$ is probably a better initial guess than $x_0 = 0$, it imposes the added cost of computing $r_0 = b - Ax_0$, and it is not at all apparent that either value is better than the other when h is large. Miranker and Chern [10] discuss more general choices for x_0 when solving linear initial value problems. Further study of choices for x_0 and their effects seems to be needed.

An observation of crucial importance, that was also made and utilized by Gear and Saad in [6], is that in Algorithm 3.2 the matrix A is not needed explicitly—only the action of A times a vector v is necessary. By using the relation $A = I - h\beta_0 J$, this means we require values of the product Jv, where $J = J(t_n, y_n(m))$. We can approximate

Jv by using the difference quotient

$$Jv \simeq w = [f(t_n, y+\sigma v) - f(t_n, y)]/\sigma, \tag{4.4}$$

where y denotes $y_n(m)$, for a suitably chosen scalar σ. Note that if $f(t_n, y)$ has been saved, then this only requires one additional f evaluation.

The choice of σ is limited by roundoff error if σ is too small, in that the two values f in (4.4) may be numerically equal in some components while the true components of Jv may not be zero. Also, σ is limited by truncation error if σ is too large, in that f may be nonlinear between y and $y+\sigma v$, and the difference quotient (4.4) may be inaccurate as a result. In Algorithm 3.2, Jv is needed for vectors $v = D\tilde{v}$ which are normalized to 1 in the weighted RMS norm: $\|v\|_{\text{WRMS}} = \|\tilde{v}\|_2 = 1$. On the other hand, the test on local error in LSODE requires that the estimated local error vector e satisfy $\|e\|_{\text{WRMS}} \leqq 1$, and we can expect that $\|e\|_{\text{WRMS}} \simeq 1$ for the step sizes selected. As a local error estimate, e can be regarded as a small correction to y, whose size is about at the user's tolerance level. So for any vector d having the same WRMS norm as e, it is likely that f is reasonably close to being linear between y and $y+d$, but unlikely that d is so small as to make f hard to resolve due to roundoff between y and $y+d$. Therefore, a reasonable criterion on σ is to make σv and e have the same norm (WRMS norm). This leads to the choice

$$\sigma = 1, \tag{4.5}$$

which we have adopted in the algorithm. There has been no clear evidence that this choice is ever a bad one, but this issue deserves some further study.

As an alternative to using (4.4), it is easy to give the user the option of supplying his own routine for the computation of Jv. This has been done in the experimental version of LSODE containing SIOM.

It remains to set the test constant δ. The discussion in § 2 concludes that δ should be much smaller than the tolerance level in the solution vector $x_n = h\dot{y}_n$ of the nonlinear system (1.4) being solved. We denote this tolerance level (in terms of the WRMS norm) by ε_1. The value of ε_1 is determined by the nature of the local error estimation in LSODE, independently of the means used to solve (1.4). A vector e of estimated local errors is formed from the difference between the prediction $x_n(0)$ and the final corrected value x_n:

$$e = [x_n - x_n(0)]/\tau,$$

τ being a coefficient depending only on the current order q ($\tau = \text{TESCO}(2, NQ)$ in subroutine STODE). The local error control is designed to keep $\|e\|_{\text{WRMS}} \simeq 1$, and hence the tolerance level (in WRMS norm) for the error in e (due to inaccuracy in x_n) is set to a heuristic constant $\eta_1 < 1$, given by $\eta_1 = 1/2(q+2)$ (= CONIT in STODE). Therefore, the tolerance level for errors in x_n is $\varepsilon_1 = \tau\eta_1$. Finally, we choose another heuristic constant $\delta_1 < 1$ and set $\delta = \delta_1\varepsilon_1$. The value we have adopted for the present is $\delta_1 = 0.05$, as this seems to work well in our tests. However, it is an optional input to the modified solver, and so other values can be easily used.

One further modification has been made in the Newton iteration strategy: When SIOM fails or the Newton convergence test fails after 3 iterations, the step size h is cut by a factor of $\frac{1}{2}$, not $\frac{1}{4}$ as in LSODE, because we expect the convergence region to be larger for the Newton method than for modified Newton (even though the former is an inexact Newton method).

The modified version of the basic algorithm in STODE is then as follows:

(1) Predict $y_n(0)$ and $x_n(0)$.
(2) Compute $f(t_n, y_n(0))$; set $m=0$, and reset the convergence rate constant C to 0.7.
(3) Form $F_n(x_n(m))$.
(4) Call SOLSY
  (a) If SIOM failed to converge and $m=0$, and the final $\rho_l > \max[1, \|F_n(x_n(0))\|_{\text{WRMS}}]$, set $h \leftarrow h/2$ and go to step (1) (redo time step).
  (b) If SIOM failed to converge and $m>0$, and the final $\rho_l > 1$, set $h \leftarrow h/2$ and go to step (1) (redo time step).
  (c) Otherwise, correct to get $x_n(m+1)$.
(5) Update estimate of convergence rate constant C, if $m \geqq 1$.
(6) Test for convergence.
(7) If the convergence test failed:
  (a) Set $m \leftarrow m+1$.
  (b) If $m<3$, compute $f(t_n, y_n(m))$ and go to Step (3).
  (c) If $m=3$, set $h \leftarrow h/2$ and go to Step (1) (redo time step).
(8) If convergence test passed, update history, do error test, etc.

The comments regarding steps (7) and (10) after the unmodified algorithm are still relevant for steps (5) and (8) here, respectively.

5. Numerical tests. The SIOM algorithm described above, and implemented in a modified version of the LSODE solver, has been tested on various ODE test problems. In this section we give, for each of three test problems, a description of the problem, numerical results obtained, and some discussion. All three problems are based on time-dependent multidimensional partial differential equation (PDE) systems, solved by the method of lines. In addition, we ran the test problems given in [6], and one of those in [3], with our solver. All of the tests were done on a Cray-1 computer with the CFT compiler.

The algorithms tested are (a) the unaltered LSODE package (as discussed in § 4.1), and (b) a version denoted here by LSODP (P for projection method), modified to use the SIOM algorithm (the scaled form of the IOM algorithm, as described in § 4.2) to solve the linear systems (4.3). The BDF method was selected in both cases, with banded Jacobian treatment in LSODE. In the LSODP tests, except where noted, we used the parameter values $p = l_{\max} = 5$ and $\delta_1 = 0.05$, in the notation of § 4.2. In most cases, the J evaluations in LSODE were done by a user-supplied subroutine, while the products Jv in LSODP were generated using the difference quotient (4.4).

For each of the test cases, various statistics were printed after the completion of the run. Those of interest include the following:

R.T. = run time (CPU sec),
NST = number of time steps,
NFE = number of f evaluations,
NJE = number of evaluations of J (in LSODE) or of Jv (in LSODP) (= the number of LU decompositions for LSODE),
NSIOM = number of calls to routine SIOM (= the number of corrector iterations in LSODP),
AVDIM = NJE/NSIOM = average dimension l of the Krylov subspace in the SIOM iterations.

Of course the counters NSIOM and AVDIM are relevant only to LSODP. When the

difference quotient form (4.4) of Jv is used in LSODP, NJE is given by NJE = NFE − NSIOM − 1. The number AVDIM is significant in that it indicates on the average how hard the SIOM algorithm must work to achieve convergence. If on a particular problem AVDIM is very close to the value of the parameter $l_{\max}$, then it may be wise to increase $l_{\max}$ to improve the overall accuracy and efficiency for that problem. We will also tabulate the work space, which is the total length in words of the real and integer work arrays required. For the options involved here, this length is $42+(11+2ML+MU)N$ for LSODE (where ML and MU are the half-bandwidths), and $107+16N$ for LSODP.

5.1. Test problem 1. This problem is based on a pair of PDE's in two dimensions, representing a simple model of ozone production in the stratosphere with diurnal kinetics. There are two dependent variables c^i, representing concentraions of O_1 and O_3 (ozone) in moles/cm^3, which vary with altitude z and horizontal position x, both in km, with $0 \leqq x \leqq 20$, $30 \leqq z \leqq 50$, and with time t in sec, $0 \leqq t \leqq 86400$ (one day). These obey a pair of coupled reaction-transport equations, with horizontal diffusion and advection and nonuniform vertical diffusion:

$$\frac{\partial c^i}{\partial t} = K_h \frac{\partial^2 c^i}{\partial x^2} + \frac{\partial}{\partial z}\left(K_v(z)\frac{\partial c^i}{\partial z}\right) + V\,\partial c^i/\partial x + R^i(c^1, c^2, t) \qquad (i = 1, 2),$$

$$K_h = 4 \cdot 10^{-6}, \quad K_v(z) = 10^{-8} e^{z/5}, \quad V = .01,$$

$$R^1(c^1, c^2, t) = -k_1 c^1 - k_2 c^1 c^2 + k_3(t) \cdot 7.4 \cdot 10^{16} + k_4(t) c^2,$$

$$R^2(c^1, c^2, t) = k_1 c^1 - k_2 c^1 c^2 - k_4(t) c^2,$$

$$k_1 = 6.031, \qquad k_2 = 4.66 \cdot 10^{-16},$$

$$k_3(t) = \begin{cases} \exp\left[-22.62/\sin\left(\pi t/43200\right)\right] & \text{for } t < 43200, \\ 0 \quad \text{otherwise,} \end{cases}$$

$$k_4(t) = \begin{cases} \exp\left[-7.601/\sin\left(\pi t/43200\right)\right] & \text{for } t < 43200, \\ 0 \quad \text{otherwise.} \end{cases}$$

Homogeneous Neumann boundary conditions are posed:

$$\frac{\partial c^i}{\partial x} = 0 \quad \text{on } x = 0 \text{ and } x = 20, \qquad \frac{\partial c^i}{\partial z} = 0 \quad \text{on } z = 30 \text{ and } z = 50.$$

The initial condition functions are polynomials chosen to be slightly peaked in the center and consistent with the boundary conditions:

$$c^1(x, z, 0) = 10^6 \alpha(x)\beta(z), \qquad c^2(x, z, 0) = 10^{12}\alpha(x)\beta(z),$$

$$\alpha(x) \equiv 1 - (.1x - 1)^2 + (.1x - 1)^4/2,$$

$$\beta(z) \equiv 1 - (.1z - 4)^2 + (.1z - 4)^4/2.$$

The PDE's are treated by central differencing, on a rectangular grid with uniform spacings, $\Delta x = 20/(J-1)$ and $\Delta z = 20/(K-1)$. If c^i_{jk} denotes the approximation to $c^i(x_j, z_k, t)$, where $x_j = (j-1)\Delta x$, $z_k = 30 + (k-1)\Delta z$, $1 \leqq j \leqq J$, $1 \leqq k \leqq K$, then we obtain the following ODE's:

$$\begin{aligned} \dot{c}^i_{jk} = {} & R^i(c^1_{jk}, c^2_{jk}, t) + (K_h/\Delta x^2)(c^i_{j+1,k} - 2c^i_{jk} + c^i_{j-1,k}) \\ & + (1/\Delta z^2)[K_v(z_{k+1/2})(c^i_{j,k+1} - c^i_{jk}) - K_v(z_{k-1/2})(c^i_{jk} - c^i_{j,k-1})] \\ & + (V/2\Delta x)(c^i_{j+1,k} - c^i_{j-1,k}). \end{aligned}$$

The boundary conditions are simulated by taking

$$c^i_{0,k} = c^i_{2,k},\ c^i_{J+1,k} = c^i_{J-1,k} \qquad (\text{all } k)$$

and

$$c^i_{j,0} = c^i_{j,2},\ c^i_{j,K+1} = c^i_{j,K-1} \qquad (\text{all } j).$$

The size of the ODE system is $N = 2JK$. The variables are indexed first by species, then by x position, and finally by z position. Thus in $\dot{y} = f(t, y)$, we have $c^i_{jk} = y_m$ with $m = i + 2(j-1) + 2J(k-1)$.

For these tests, we chose $J = K = 20$ ($N = 800$). The problem is stiff because of the kinetics, and the Jacobian has half-bandwidths $ML = MU = 2J = 40$. A mixed relative/absolute error tolerance was chosen, with RTOL $= 10^{-5}$ and ATOL $= 10^{-3}$. For both LSODE and LSODP, both the user-supplied Jacobian (or Jv) routine and the internal difference quotient option were tested. These are denoted by USJ and DQJ, respectively, in the tabulated results. In the DQJ case, LSODE does $ML + MU + 1 = 81$ evaluations of f for each value of the (banded) Jacobian.

For this problem, we also give the solution by a third solver, GEARBI [7]. It uses the BDF method and block-SOR on the linear systems (4.3) (within a modified Newton iteration). Since GEARBI was specifically designed to solve ODE systems arising from 2-D kinetics-transport problems, and takes full advantage of the Jacobian sparsity structure, a comparison of it with LSODP is quite significant. Here, GEARBI was altered so as to use a mixed relative/absolute error weighting identical to that used in the LSODE and LSODP tests. For GEARBI, the counter NJE is the number of evaluations and LU decompositions of the block-diagonal part of $I - h\beta_0 J$ (the blocks being 2×2 here), as needed for the block-SOR algorithm.

We solve first the problem with no advection ($V = 0$). (See also [9] for comparison tests on this problem.) The results of testing the three solvers on this problem are shown in Table 1a. LSODP shows a dramatic improvement over LSODE, trading 90 or 97 Jacobian evaluations (and the same number of banded LU decompositions) for 651 SIOM calls and 727 to 731 evaluations of Jv, resulting in a savings of roughly 14 sec. (66%) of the CPU time in the USJ case, and a savings of 42 sec. (86%) in the DQJ case. Another significant result is that, while the use of a difference quotient Jacobian imposes a severe cost penalty for LSODE (81 f evaluations per J value here), for LSODP the closed form computation of Jv is actually slower than the difference quotient approximation (1 f evaluation per Jv value), and the latter shows no signs of being less accurate. Of course there is much less user effort with the DQJ option in setting up the problem for the solver. We note also that the work space required was reduced by approximately 88%, and the value of AVDIM = 1.1 indicates that SIOM is not having any difficulty at all achieving convergence here. The GEARBI

TABLE 1a
Test results for Problem 1, $V = 0$.

Solver	R.T	NST	NFE	NJE	NSIOM	Work space	AVDIM
LSODE (USJ)	20.6	462	659	90	—	104,842	—
LSODE (DQJ)	48.6	490	8,574	97	—	104,842	—
GEARBI	17.1	425	660	48	—	12,004	—
LSODP (USJ)	7.1	340	652	727	651	12,907	1.12
LSODP (DQJ)	6.8	339	1,383	731	651	12,907	1.12

solution is only slightly faster than that of LSODE (USJ), and 2.5 times slower than the LSODP (DQJ) solution. (Not shown in the table is the fact that GEARBI performed an average of 1.34 block-SOR iterations per modified Newton iteration.) GEARBI is competitive with LSODP in storage, but this does not remain true as the number of species grows (because its storage goes as the square of the number of species), or if the transport coupling is more complicated.

For this test, the success of LSODP can be predicted from spectral information about the problem. Using the RG driver in EISPACK [15], we computed the spectrum of the problem Jacobian $J = \partial f/\partial y$ for a 6×6 mesh at each of the 12 equally spaced output points, and we can assume that analogous results hold for a 20×20 mesh. The spectrum is real and has contributions from the kinetics and the transport, but the stiffness is a result only of the kinetics, which (after the initial transient) produces a tight cluster of eigenvalues (one per mesh point) about a point $\lambda \simeq -6$ (roughly the value of $-k_1$), with all the others much smaller in size. Thus the SIOM algorithm might be expected to behave as if there were only one stiff eigenvalue, and in fact it does. This would not be the case if the diffusion coefficients were much larger. However, the values of those were dictated by an actual atmospheric model, and so we did not alter them.

Next we solve the problem in its stated form, with horizontal advection velocity $V = .01$. Here the presence of advection limits the step sizes for any solver, as well as making the linear system problem harder. The Jacobian now has highly nonsymmetric contributons from both the transport and the kinetics. The computed spectrum (for a 10×10 mesh) still has a tight cluster at about $-k_1$, although it is mostly nonreal and the spread is slightly larger. The test results are shown in Table 1b. Again, LSODP is the fastest solver by a wide margin, even though AVDIM = 2.2 shows that SIOM is working harder than before. GEARBI is faster than LSODE (DQJ) but slower than LSODE (USJ), presumably because of the advection contribution to the Jacobian (making it now do 2.4 block-SOR iterations per modified Newton iteration).

TABLE 1b
Test results for Problem 1, $V = .01$.

Solver	R.T.	NST	NFE	NJE	NSIOM	Work space	AVDIM
LSODE (USJ)	113	3,137	4,165	415	—	104,842	—
LSODE (DQJ)	235	3,132	38,363	422	—	104,842	—
GEARBI	163	2,976	4,084	345	—	12,004	—
LSODP (USJ)	74	2,374	4,595	10,296	4,594	12,907	2.24
LSODP (DQJ)	71	2,447	15,198	10,468	4,729	12,907	2.21

Both cases of this problem were also run with nondefault values of p and δ_1 in LSODP (DQJ). The runs with $p < l_{\max}$ (and the default δ_1) showed no improvement in speed over the runs with the default ($p = 5$). In the case $V = 0$, the results were nearly identical to the default results (counts and run time) for $p = 3$ and 4, while for $p = 2$ the run time was slightly higher (6.9 sec vs. 6.8 sec). In the latter run the total number of SIOM iterations was 793 (vs. 731 before), offsetting the reduced cost per iteration. In the case $V = .01$, the run times for $p = 4$, 3, and 2 were progressively longer (73 to 104 sec). The value of AVDIM was larger than for $p = 5$ in all cases, and for $p = 2$ and 3 the number of steps was also. These results are in accord with the highly nonsymmetric nature of the Jacobian resulting from the kinetics terms. Runs made with $p = l_{\max} = 5$ but with δ_1 larger than .05 (the default), namely .1, .2, and .5, gave

mixed results. For $V = 0$, the results were slightly worse (run times of 6.8 sec to 7.5 sec vs. 6.8 with the default), while for $V = .01$, the results were better (run times of 64.6 to 65.7, vs. 70.6 sec with the default).

5.2. Test problem 2. This problem is based on a reaction-diffusion system arising from a Lotka–Volterra predator-prey model, with diffusion effects in two space dimensions included. There are two species variables, $c^1(x, y, t)$ and $c^2(x, y, t)$, representing (respectively) the prey and predator species densities over the spatial habitat $\Omega = \{(x, y): 0 \leqq x \leqq 1,\ 0 \leqq y \leqq 1\}$ and time t in sec, $0 \leqq t \leqq 3$. The equations are

$$\frac{\partial c^i}{\partial t} = d_i \left(\frac{\partial^2 c^i}{\partial x^2} + \frac{\partial^2 c^i}{\partial y^2} \right) + f^i(c^1, c^2) \qquad (i = 1, 2),$$

$$f^1(c^1, c^2) = c^1(b_1 - a_{12}c^2), \qquad f^2(c^1, c^2) = c^2(b_2 - a_{21}c^1),$$

$$d_1 = .05, \quad d_2 = 1.0, \quad b_1 = 1, \quad b_2 = -1000, \quad a_{12} = .1, \quad a_{21} = -100.$$

Homogeneous Neumann boundary conditions are imposed:

$$\partial c_i / \partial x = 0 \quad \text{on } x = 0 \text{ and } x = 1; \qquad \partial c^i / \partial y = 0 \quad \text{on } y = 0 \text{ and } y = 1.$$

The initial conditions involve products of cosines and are chosen to be consistent with the boundary conditions:

$$c^1(x, y, 0) = 10 - 5 \cos(\pi x) \cos(10\pi y),$$

$$c^2(x, y, 0) = 17 + 5 \cos(10\pi x) \cos(\pi y).$$

As $t \to \infty$, the solution becomes spatially homogeneous and tends to a time-periodic solution of the Lotka-Volterra ODE system $dc^i/dt = f^i$ $(i = 1, 2)$. This ODE system is alternately stiff and nonstiff depending on the position of the solution in (c^1, c^2) phase space.

The two PDE's are again treated by central differencing on a rectangular grid with uniform spacings, $\Delta x = 1/(J-1)$ and $\Delta y = 1/(K-1)$, with boundary conditions discretized as before. The system is a stiff one of size $N = 2JK$, and the Jacobian has half-bandwidths $ML = MU = 2J$. A mixed relative/absolute error tolerance was chosen, with $\text{RTOL} = 10^{-6}$ and $\text{ATOL} = 10^{-4}$. We tested 5 cases with $J = K$ varying from 10 ($N = 200$) to 50 ($N = 5000$).

We also looked at the spectrum of the Jacobian for this problem (for a 10×10 mesh) and found that after an initial transient period the dominant eigenvalues λ are those from the discrete diffusion operator. Thus the problem can be predicted to be nonstiff for crude meshes and stiff for fine meshes, and experiments show that this is true, with the nonstiff method becoming less efficient than a stiff one when J and K are 20 or more. We also see that the stiff part of the spectrum is well spread out, rather than clustered as in the first problem. More precisely, the extreme eigenvalue from the discrete diffusion terms is roughly $-8(J-1)^2$ for $J = K$, while the step sizes h found to be appropriate for the accurate resolution of the oscillatory steady state are about .002. Thus, in the 50×50 case, the values of $h\lambda$ cover an interval from 0 to about -40 in a nonclustered manner. Nonreal eigenvalues also occur on parts of the limit cycle.

For this (and the next) problem, we tested only the cheaper Jacobian option for each solver, namely the user-supplied Jacobian with LSODE, but the difference quotient Jv option in LSODP. A comparison involving the DQJ options for both solvers would be more fair and more realistic, but the expense of testing LSODE (DQJ) makes this prohibitive.

The test results on this problem are given in Table 2. Here, in all cases, LSODP gives superior run times, with a 51% savings in run time and a 95% savings in required work space for the 50×50 mesh problem. However, note the steady rise in AVDIM with grid size, as predicted from the spectral results, and indicating that SIOM might require a larger value of $l_{\max}$ if the grid were further refined. For LSODP the run time and storage costs here are nearly proportional to N, but for LSODE they grow more rapidly than that.

TABLE 2
Test results for Problem 2.

Solver	Mesh	R.T.	NST	NFE	NJE	NSIOM	Work space	AVDIM
LSODE	10×10	6.7	1,248	1,635	129	—	14,242	—
LSODP	10×10	5.8	1,230	4,891	2,380	2,510	3,307	.95
LSODE	20×20	52.0	1,346	1,795	197	—	104,842	—
LSODP	20×20	26.3	1,085	6,211	3,935	2,275	12,907	1.73
LSODE	30×30	131.3	1,197	1,569	144	—	343,242	—
LSODP	30×30	76.1	1,176	8,288	5,954	2,333	28,907	2.55
LSODE	40×40	377.0	1,231	1,565	173	—	803,242	—
LSODP	40×40	164.3	1,140	10,337	8,044	2,292	51,307	3.51
LSODE	50×50	661.4	1,145	1,493	139	—	1,555,042	—
LSODP	50×50	322.8	1,261	12,608	10,149	2,458	80,107	4.13

Two experiments were done with nondefault values. For the 20×20 case, values of $\delta_1 = .1$, .2, and .5 were used. The results in all cases were inferior to those with the default ($\delta_1 = .05$), with run times varying from 27.3 sec to 29.2 (vs. 26.3 for the default). For the 50×50 case, we also ran LSODP with $p = 2$. The various counts were nearly identical to the default results, and the run time was reduced to 309.8 sec. This confirms the nearly symmetric nature of the Jacobian, as dominated by the diffusion contributions, and the reduced cost per iteration with $p = 2$.

5.3. Test problem 3. Like Problem 2, this problem is based on a reaction-diffusion system comprising a Lotka-Volterra competition model, but in this case in 3 space dimensions. The two variables, c^1 and c^2, represent the species densities, and vary over the unit cube, $0 \leqq x, y, z \leqq 1$, and $0 \leqq t \leqq 10$. The equations are analogous to those in Problem 2, but have different coefficients:

$$\frac{\partial c^i}{\partial t} = d_i \Delta c^i + f^i(c^1, c^2) \qquad (i = 1, 2),$$

$$d_1 = .05, \qquad d_2 = 1.0,$$

$$f^1(c^1, c^2) = c^1(b_1 - a_{11}c^1 - a_{12}c^2),$$

$$f^2(c^1, c^2) = c^2(b_2 - a_{21}c^1 - a_{22}c^2),$$

$$a_{11} = 10^6, \quad a_{12} = 1, \; a_{21} = 10^6 - 1, \quad a_{22} = 10^6,$$

$$b_1 = b_2 = (1 + \alpha xyz)(10^6 - 1 + 10^{-6}).$$

As before, homogeneous Neumann boundary conditions are posed. The initial conditions are

$$c^1(x, y, z, 0) = 500 + 250 \cos(\pi x) \cos(3\pi y) \cos(10\pi z),$$

$$c^2(x, y, z, 0) = 200 + 150 \cos(10\pi x) \cos(\pi y) \cos(3\pi z).$$

The coefficients above have been chosen so that, as $t \to \infty$, the solution of the system approaches a steady state which is (intentionally) not flat in space. This steady state is given roughly by the asymptotic solution of the problem without diffusion, namely:

$$c^1 = (1-10^{-6})(1+\alpha xyz), \qquad c^2 = 10^{-6}(1+\alpha xyz).$$

The two PDE's are discretized on a regular J by K by L grid in a manner completely analogous to the formulation in Problems 1 and 2, giving an ODE system of size $N = 2JKL$. We consider $\alpha = 0$ and $\alpha = .2$, and vary the mesh width $J = K = L$ from 6 ($N = 432$) to 20 ($N = 16{,}000$). Tolerance parameters are taken to be RTOL $= 10^{-6}$ and ATOL $= 10^{-8}$.

For this problem, the use of LSODE is inappropriate because of the amount of sparsity structure present. However, in addition to testing LSODP, we have solved the problem with GEARBI, in a suitably modified form so that it handles the unequal diffusion coefficients. In LSODP, we used only the difference quotient Jv option.

Beginning with the easier case $\alpha = 0$, we show results for five cases in Table 3a. Because of the high expense and predictable results, the GEARBI solution was not done for the last two (finest) meshes. The growth of the run time and storage requirement is nearly proportional to N for both LSODP and GEARBI. GEARBI has a slight advantage in storage here, because only the diagonal blocks of the Jacobian are stored. However, LSODP is the faster solver, by a factor of about 2 or more, uniformly.

TABLE 3a
Test results for Problem 3, $\alpha = 0$.

Solver	Mesh	R.T.	NST	NFE	NJE	NSIOM	Work space	AVDIM
LSODP	6×6×6	6.1	554	2,218	1,189	1,028	7,019	1.16
GEARBI	6×6×6	11.8	553	592	67	—	5,618	—
LSODP	10×10×10	33.3	603	2,785	1,671	1,113	32,107	1.50
GEARBI	10×10×10	68.5	580	624	79	—	26,002	—
LSODP	14×14×14	91.9	599	2,840	1,730	1,109	87,915	1.56
GEARBI	14×14×14	232.8	618	676	102	—	71,346	—
LSODP	18×18×18	204.3	615	2,995	1,871	1,123	186,731	1.67
LSODP	20×20×20	325.0	659	3,528	2,315	1,212	256,107	1.91

Next, we show the results for the case $\alpha = .2$, where the steady state is truly space-dependent. Here, for the meshes considered, one finds that near the steady state solution, the interaction terms dominate the Jacobian, which is nearly symmetric, and from the nondiffusive system the dominant part of the spectrum consists of JKL points widely spread over the interval from -10^6 to $-10^6(1+\alpha)$. Thus SIOM can be expected to have more difficulty with the linear systems in this case. The results for three meshes are shown in Table 3b. Note that LSODP was not able to complete the $14\times14\times14$ case with default values for all inputs, but was with a larger value of $l_{\max}$. The overall average AVDIM is deceptive for the LSODP runs; in fact the average l for the interval $1 \leqq t \leqq 10$ is equal to $l_{\max}$ exactly in the LSODP runs with $l_{\max} = 5$, and 9.96 in the run with $l_{\max} = 10$. Thus SIOM *is* having more difficulty converging, but a detailed explanation for this would require a considerably greater diagnostic effort. At any rate, while this case is more costly for LSODP, the GEARBI solution cost is much the same as for $\alpha = 0$, as expected. The success of the run with $l_{\max} = 10$ and $p = 2$ agrees with the spectral information. It was also observed that the numerical solution is sensitive to

errors in the smaller component (c^2), and that tightening the absolute tolerance from 10^{-8} to 10^{-10} also allowed LSODP (with $l_{max} = p = 5$) to complete the solution for the $14 \times 14 \times 14$ mesh, but at a run time of 344 sec.

TABLE 3b
Test results for Problem 3, $\alpha = .2$.

Solver	Mesh	R.T.	NST	NFE	NJE	NSIOM	Work space	AVDIM
LSODP	$6\times6\times6$	7.9	591	2,973	1,873	1,099	7,019	1.70
GEARBI	$6\times6\times6$	11.5	553	590	65	—	5,618	—
LSODP	$10\times10\times10$	50.6	736	4,381	3,054	1,326	32,107	2.30
GEARBI	$10\times10\times10$	72.8	596	647	90	—	26,002	—
LSODP	$14\times14\times14$	(run stopped at $t = 5.3$ with repeated convergence failures)						
LSODP*	$14\times14\times14$	131.6	601	4,235	3,119	1,115	115,510	2.80
GEARBI	$14\times14\times14$	233.1	609	665	98	—	71,346	—

* Run with $l_{max} = 10$ and $p = 2$.

5.4. Other tests. For the sake of completeness, we mention here three other test problems, namely those used by Gear and Saad in [6], and one of those used by Chan and Jackson in [3].

The first test problem in [6] is an adaptation of one due to Krogh. Our test results with LSODE (USJ and DQJ) match those in [6] exactly (except for run times), even though run on a different computer. However, our results using IOM are quite different from theirs (for both cases of their algorithm). For the test case $N = 50$, we get NST = 151 (vs. 184 or 180 in [6]) and NFE = 669 (vs. 720 or 643), and the LSODP run time is less than those for LSODE by a factor of .48 to .55 (vs. .94 to 1.16). For the case $N = 80$, we get NST = 149 (vs. 186 or 172) and NFE = 660 (vs. 733 or 640), and the LSODP run time is less by a factor of .29 to .32 (vs. .54 to .65).

The second problem in [6] is a discrete form of the heat equation on a square.[2] Our test results for LSODE (DQJ) match those in [6], but the results using IOM differ greatly. For either $p = 5$ (our default) or $p = 2$ (utilizing the symmetry of A), we get NST = 132 (vs. 183) and NFE = 601 (vs. 724). Our two LSODP run times are nearly the same, and less than the LSODE time by a factor of .87 (vs. 2.4).

For both of these problems, it appears that in our runs the step sizes are hardly affected at all by the use of SIOM in place of a direct linear system solver. In contrast, the runs in [6] with IOM take about 20% to 40% more steps, presumably forced by convergence failures of the IOM algorithm used there (which differs in many respects from ours).

It is our understanding that the algorithm in [6] involves external storage for some of the vectors v_i, and associated I/O operations. As a result, direct comparisons of work space and run times are not entirely meaningful. However, in the context of a large computer, our algorithm (with no external data storage or I/O) definitely appears to be preferable.

In [3], Chan and Jackson test another form of the 2-D heat equation test problem, which differs in having nonuniform initial conditions, given by $u = 16xy(1-x)(1-y)$, and a looser tolerance ATOL = 10^{-3}. We also ran this problem, using an $M \times M$ mesh with $M = 10$, 20, and 30. The results of tests with LSODE and LSODP are given in

[2] There appears to be an error in the problem statement in [6], in that A is missing a factor of $(n+1)^2$.

TABLE 4
Test results for 2-D heat equation.

Solver	Mesh	R.T.	NST	NFE	NJE	NSIOM	AVDIM
LSODE	10×10	.20	37	275	11	—	—
LSODP	10×10	.13	38	208	150	57	2.63
LSODE	20×20	1.34	38	496	11	—	—
LSODP	20×20	1.16	61	541	439	101	4.35
LSODE	30×30	4.34	40	718	11	—	—
LSODP	30×30	5.04	113	1,074	891	182	4.90

Table 4. Both solvers were run with difference quotient evaluations of J (or Jv). Since the Jacobian is symmetric here, we also ran the problem with $p=2$ in LSODP, and verified that in all cases the results were the same (or very nearly so), while the run times were reduced by about 4% or less.

We note that LSODP is faster than LSODE in the first two cases, but not for the 30×30 mesh. The higher relative cost of LSODP there can be attributed to the impact of the (nonclustered) spectrum on the step sizes and on AVDIM. The LSODP results can also be compared with those from the conjugate residual method (without preconditioning) in [3], where the number of iterations (analogous to NJE above) is 273 for $M=10$, 417 for $M=20$, and 652 for $M=30$. Thus for this problem and this method the iteration count also rises with mesh size, but not as rapidly as in our SIOM results. This may be due to inherent differences between the conjugate residual method and IOM (which becomes the conjugate gradient method here), and may also be due in part to a higher maximum iteration count allowed in [3].

6. Conclusions and future work. We have presented here an essentially matrix-free Newton-like iteration scheme and its implementation into the LSODE package for the solution of stiff systems of ODE's. The focus has been on those problems for which the cost of performing the linear algebra associated with solving the system (2.2) by traditional matrix methods far outweighs that of several function evaluations. Thus one can hope that the use of a matrix-free method such as the IOM algorithm to perform the Newton iterations involved in the integration could succeed, at a greatly reduced cost in run time, overall work, and storage. The inclusion of scaling associated with error tolerances, in the SIOM form of the algorithm, is crucial for robustness, we feel. Although our testing has been somewhat limited, initial results with our experimental code LSODP seem very promising. In cases with tight spectral clustering, the method is quite successful, and understandably so. In the cases with many widely spread stiff eigenvalues, LSODP is sometimes unpredictably successful, and sometimes unsuccessful.

There are several aspects of the method which need further investigation. For example, a deeper understanding of the convergence of the IOM algorithm may indicate more clearly a class of problems on which IOM will work very well, while at the same time eliminating other classes. Currently, it is very difficult to predict how IOM will perform on realistic problems.

The corrector loop strategy given in the modified algorithm of § 4.2 may show improvement by employing some of the techniques from Dembo, Eisenstat, and Steihaug [4] for Inexact Newton Methods. In [6, § 2.2], Gear and Saad have suggested ways of reusing the vectors $v_1, \cdots, v_l$ and the matrix H_l (or their scaled equivalents). Earlier, we did some testing of these ideas, but did not find them beneficial. In any

case, much more development and testing needs to be done to make a final decision as to their usefulness.

In the test results of § 5, we have used the parameter values $l_{\max} = p = 5$ almost exclusively. This effectively reduces Algorithm 3.2 to a scaled version of Arnoldi's algorithm, instead of the IOM algorithm. Some of the test cases were also run with p less than $l_{\max}$, and some of the resulting run times were lower, but not by very much. This is probably because both p and $l_{\max}$ are relatively small, and the cost of evaluating f is much larger than that of an inner product. In the course of further work, it may be possible to set $l_{\max}$ and p in a dynamic manner, thus making IOM more robust, and then these two numbers would probably have larger and unequal values much of the time. For now, we are encouraged by the fact that the algorithm performs as well as it does with such small values of p and $l_{\max}$, by comparison with the values used in [6], where $l_{\max}$ was usually taken to be 15 to 30.

For our algorithm as presently implemented, some condition on the problem Jacobian, such as spectral clustering, seems to be necessary for success. But a much wider problem class may be reachable if some preconditioning is included in the linear iteration. We intend to explore this idea with various natural preconditioning matrices, although this detracts from the matrix-free nature of the method. We also intend to search for matrix-free preconditionings which have potential effectiveness in the stiff ODE setting.

While this paper addresses only explicitly given ODE systems $\dot{y} = f$, there is also much interest in implicit systems, especially linearly implicit ODE systems $A(t, y)\ \dot{y} = g(t, y)$ (A = a square matrix). The algebraic system arising from the latter problem type, when solved by a BDF method (or any other implicit linear multistep method), is somewhat more complicated than that studied here [9]. But it is not hard to extend our Newton/IOM algorithm to that problem. In addition to the residual routine that computes $r = g - A\dot{y}$, the user would have to supply a routine to compute the product $A(t, y)v$ for any given v. The algorithm would also need values of $(\partial r/\partial y)v$, and could obtain them cheaply by a difference quotient, or else from an optional user-supplied routine. The result would again be a matrix-free method, with the same basic features as that described here for $\dot{y} = f$.

Even more generally, it is likely that algorithms of the type studied here may be useful for arbitrary nonlinear algebraic systems $F(x) = 0$. When the system is poorly conditioned, Newton-like methods are generally preferred, but when the system size is large, such methods are very expensive (in time and storage), as currently used. For such cases, a combination of the Inexact Newton Method and IOM (or scaled IOM) may be quite competitive with present algorithms. This idea, and variations of it, are discussed by Chan and Jackson [2].

Finally, we remark that the IOM algorithm is only one of several Krylov subspace methods whch have the potential for reducing the overall storage and work in solving large stiff ODE systems. Rather than test an array of methods on one or two test problems, we elected to test essentially one promising method on an array of problems. In choosing IOM, we were influenced by the preliminary results of Gear and Saad [6]. From the more recent work of Chan and Jackson [3], it seems clear that other Krylov methods are also promising candidates for use in stiff ODE solvers, and additionally that preconditioning is essential if the methods are to be robust. We will certainly continue to investigate these areas.

Acknowledgments. We are grateful to two anonymous referees for clarifying a number of features in our algorithm and testing, and for suggesting improvements.

We also thank Don Wuebbles (LLNL Atmospheric Sciences Div.) for performing experiments with, and giving feedback on, an early version of the LSODP solver.

REFERENCES

[1] W. E. ARNOLDI, *The principle of minimized iterations in the solution of the matrix eigenvalue problem*, Quart. J. Appl. Math., 9 (1951), pp. 17-29.

[2] T. F. CHAN AND K. R. JACKSON, *Nonlinearly preconditioned Krylov subspace methods for discrete Newton algorithms*, SIAM J. Sci. Stat. Comp., 5 (1984), pp. 535-542.

[3] ——— *The use of iterative linear equation solvers in codes for large systems of stiff IVPs for ODEs*, Tech. Report 170/84, Dept. Computer Science, Univ. of Toronto, March 1984.

[4] R. S. DEMBO, S. C. EISENSTAT AND T. STEIHAUG, *Inexact Newton methods*, this Journal, 19 (1982), pp. 400-408.

[5] J. J. DONGARRA, J. R. BUNCH, C. B. MOLER AND G. W. STEWART, LINPACK *Users' Guide*, Society for Industrial and Applied Mathematics, Philadelphia, 1979.

[6] C. W. GEAR AND Y. SAAD, *Iterative solution of linear equations in ODE codes*, SIAM J. Sci. Stat. Comp., 4 (1983), pp. 583-601.

[7] A. C. HINDMARSH, *Preliminary documentation of* GEARBI: *solution of ODE systems with block-iterative treatment of the Jacobian*, Lawrence Livermore National Laboratory Report UCID-30149, December 1976.

[8] ——— LSODE *and* LSODI, *Two new initial value ordinary differential equation solvers*, in ACM Newsletter, 15, 4 (December 1980), pp. 10-11.

[9] ——— ODEPACK, *a systematized collection of ODE solvers*, in Scientific Computing, R. S. Stepleman et al., eds., North-Holland, Amsterdam, 1983, pp. 55-64.

[10] W. L. MIRANKER AND I-L. CHERN, *Dichotomy and conjugate gradients in the stiff inital value problem*, Lin. Alg. Appl., 36 (1981), pp. 57-77.

[11] J. M. ORTEGA AND W. C. RHEINBOLDT, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970.

[12] Y. SAAD, *Variations on Arnoldi's method for computing eigenelements of large unsymmetric matrices*, Lin. Alg. Appl., 34 (1980), pp. 269-295.

[13] ——— *Krylov subspace methods for solving large unsymmetric linear systems*, Math. Comp., 37 (1981), pp. 105-126.

[14] ——— *Practical use of some Krylov subspace methods for solving indefinite and nonsymmetric linear systems*, SIAM J. Sci. Stat. Comp., 5 (1984), pp. 203-228.

[15] B. T. SMITH et al., *Matrix Eigensystem Routines*—EISPACK *Guide*, Lecture Notes in Computer Science 6, Springer-Verlag, New York, 1976.