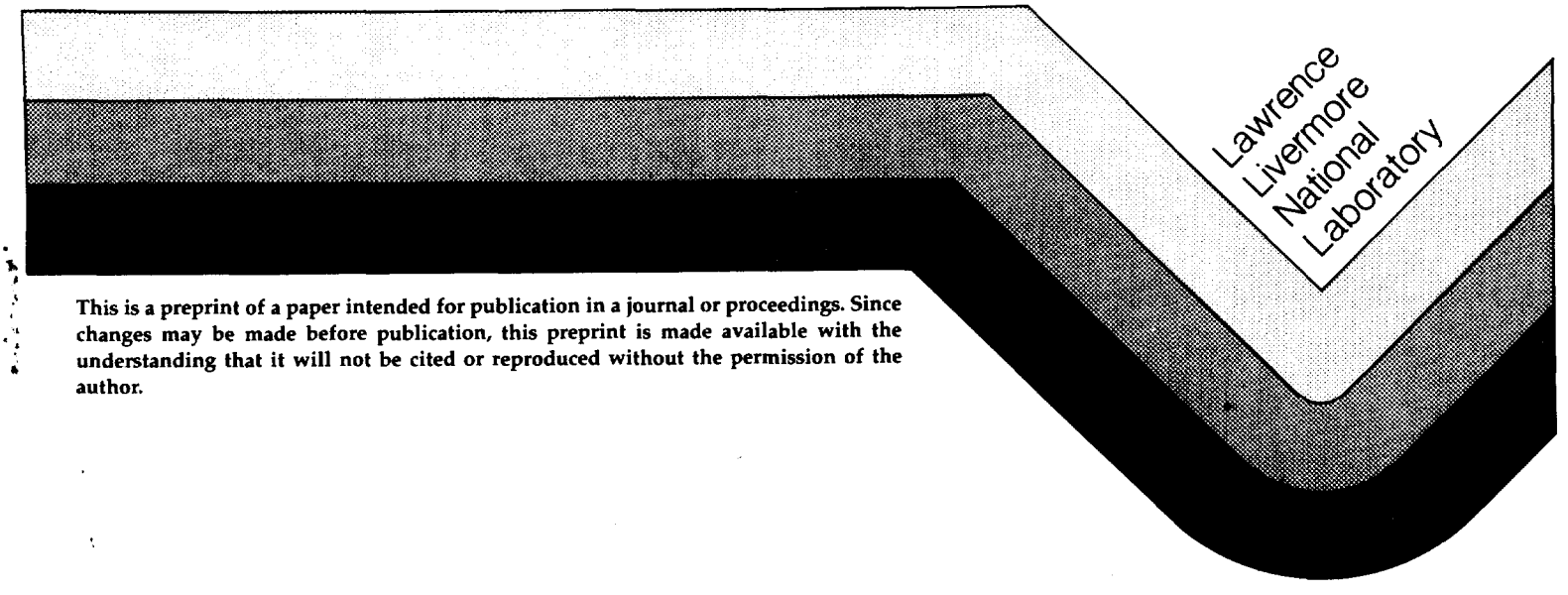
UCRL-98412
PREPRINT

VODE, A VARIABLE-COEFFICIENT ODE SOLVER

Peter N. Brown
George D. Byrne
Alan C. Hindmarsh

Prepared for submission to the SIAM Journal
on Scientific and Statistical Computing.

June 1988



VODE, A Variable-Coefficient ODE Solver*

Peter N. Brown
Computing & Mathematics Research Division, L-316
Lawrence Livermore National Laboratory
Livermore, CA 94550

George D. Byrne
Corporate Research
Exxon Research and Engineering Company
Annandale, NJ 08801

and

Alan C. Hindmarsh
Computing & Mathematics Research Division, L-316
Lawrence Livermore National Laboratory
Livermore, CA 94550

June 1, 1988

*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. The second author was supported by Exxon Research and Engineering Company.

Abstract

VODE is a new initial value ODE solver for stiff and nonstiff systems. It uses variable-coefficient Adams-Moulton and BDF methods in Nordsieck form, as taken from the older solvers EPISODE and EPISODEB, treating the Jacobian as full or banded. Unlike the older codes, VODE has a highly flexible user interface that is nearly identical to that of the ODEPACK solver LSODE.

In the process, several algorithmic improvements have been made in VODE, aside from the new user interface. First, a change in stepsize and/or order that is decided upon at the end of one successful step is not implemented until the start of the next step, so that interpolations performed between steps use the more correct data. Secondly, a new algorithm for setting the initial step size has been included, which iterates briefly to estimate the required second derivative vector. Efficiency is often greatly enhanced by an added algorithm for saving and reusing the Jacobian matrix J, as it occurs in the Newton matrix, under certain conditions. As an option, this Jacobian-saving feature can be suppressed if the required extra storage is prohibitive. Finally, the modified Newton iteration is relaxed by a scalar factor in the stiff case, as a partial correction for the fact that the scalar coefficient in the Newton matrix may be out of date.

Independently, we have studied the fixed-leading-coefficient form of the BDF methods, and have developed a version of VODE that incorporates it. This version does show better performance on some problems, but further tuning and testing is needed to make a final evaluation of it.

Like its predecessors, VODE demonstrates that multistep methods with fully variable stepsizes and coefficients can outperform fixed-step-interpolatory methods on problems with widely different active time scales. In one comparison test, on a 1-D diurnal kinetics-transport problem with a banded internal Jacobian, the run time for VODE was 36% lower than that of LSODE without the J-saving algorithm, and 49% lower with it. The fixed-leading-coefficient version ran slightly faster, by another 12% without J-saving and 5% with it.

1 Introduction

In the early 1970s, one of the more heavily used ODE initial value solvers was the GEAR package [9], which uses (in the stiff case) a fixed-coefficient Backward Differentiation Formula (BDF) method. But it was found that GEAR was unable to cope with certain chemical kinetics problems that have sharp and frequent time variations in the coefficient functions and solutions. This motivated a project in which a variable-coefficient form of that BDF method was developed. The result was a solver called EPISODE [5,14,13], which was nearly identical to GEAR in external appearance, but quite different internally in the areas of coefficient evaluation and the estimation and control of local error. (In analogy with the nonstiff Adams method in GEAR, EPISODE was augmented with a variable-coefficient Adams method, so that nonstiff systems could be handled with a simple change of an input method flag.) EPISODE was found to be more reliable on these difficult kinetics problems, but less efficient on smooth stiff problems, because of the frequent changing of the scalar coefficient in the Newton matrix [4]. Two variants were written soon afterward, mainly for the method-of-lines solution of PDEs with nonsmooth variations of the type EPSIODE was designed to accommodate. These variants are EPISODEB, for systems with banded Jacobian matrices [6], and EPISODEIB, for problems in linearly implicit form with banded Jacobians, both in analogy to the GEAR variants GEARB and GEARIB.

In an independent development, the considerable experience and user feedback from the GEAR family of solvers motivated a joint effort among people in the U.S. Department of Energy Laboratories to redesign and improve some of the ODE initial value solvers being used heavily. The goal of the project was a systematized collection of IVP solvers, to be called ODEPACK. The initial phase was the development of a standard user interface for the solvers. This was accomplished [10], and the GEAR and GEARB packages were then rewritten as a single solver, LSODE [11]. While LSODE represents only modest improvements in the internal methods and algorithms of GEAR and GEARB, it reflects a much greater flexibility with respect to user controls and options, yet

it is generally easier to use, it is much more portable, and it is far easier to install in a library environment. These properties of LSODE, and now also of the ODEPACK collection [12], have received virtually unanimous positive feedback from the user community.

VODE combines these two developments. That is, the EPISODE and EPISODEB solvers, with their fully variable-step methods, have been combined and rewritten as an initial value ODE solver called VODE, with a user interface that conforms to the standard developed for ODEPACK. Thus it has a highly flexible user interface nearly identical to that of LSODE, but it contains variable-coefficient methods which are more efficient on problems which require frequent and wide changes in step sizes. In the process, several algorithmic improvements have also been made, as described below. In addition, we have studied an alternative approach, based on the fixed-leading-coefficient form of BDF methods [15], and have implemented these in a variant of VODE, described below.

2 Basic Methods

A detailed mathematical description of the methods used in EPISODE, and hence in VODE, is given in [5], and will not be repeated here. But for the sake of completeness, a brief summary is given below.

We write the initial value problem as

$$\dot{y} = f(t, y), \quad y(t_0) = y_0, \quad y \in \mathbf{R}^N.$$

The basic linear multistep formulas for both the stiff and nonstiff cases have the form

$$\sum_{i=0}^{K_1} \alpha_{n,i} y_{n-i} + h_n \sum_{i=0}^{K_2} \beta_{n,i} \dot{y}_{n-i} = 0.$$

For use on nonstiff problems, the Adams formula is characterized by $K_1 = 1$ and $K_2 = q - 1$, and the order q varies between 1 and 12. For stiff problems, the BDF formula has $K_1 = q$ and $K_2 = 0$, and the order q varies between 1 and 5. The coefficients $\alpha_{n,i}, \beta_{n,i}$ are computed as functions of the current and past step sizes $h_j = t_j - t_{j-1} (j = n - q + 1, ..., n)$. The past history is represented by

the N by $q+1$ Nordsieck array,

$$z_n = [y_n, h_n\dot{y}_n, ..., h_n^q y_n^{(q)}/q!], \tag{2.1}$$

the scaled derivatives being those of the corresponding interpolating polynomial associated with the history data involved in the formula.

As with any implicit method, some iterative scheme must be applied at each step to solve a nonlinear system

$$G(y_n) \equiv y_n + h_n(\beta_{n,0}/\alpha_{n,0})f(t_n, y_n) + a_n = 0$$

for the advanced value y_n (a_n is a vector involving past values of y_k and $\dot{y}_k$). VODE offers a choice between functional iteration (where no matrices are involved) and a modified Newton iteration in which the Jacobian matrix $J = \partial f/\partial y$ is treated as either full or banded, and as either supplied by the user or approximated internally by difference quotients. In the Nordsieck representation, the predicted value of z_n is

$$z_{n(0)} = z_{n-1}A, \tag{2.2}$$

where A is the order q Pascal triangle matrix. The first two columns of this array are the predicted values of y_n and $h_n\dot{y}_n$, denoted $y_{n(0)}$ and $h_n\dot{y}_{n(0)}$, respectively. Then the nonlinear system takes the form

$$G(y_n) = y_n - y_{n(0)} - (h_n/\ell_1)[f(t_n, y_n) - \dot{y}_{n(0)}] \tag{2.3}$$

and the final correction to the Nordsieck array is

$$z_n = z_{n(0)} + [y_n - y_{n(0)}]\ell \tag{2.4}$$

in terms of a vector of coefficients $\ell = [\ell_0, \ell_1, ..., \ell_q](\ell_0 = 1)$.

Following a successful corrector iteration for y_n, the local error is estimated and tested. Regardless of its outcome, a change in step size is considered, either for the current step or the next one, depending on the error test. Periodically, a change in the order q is also considered, based on

estimated local errors at orders $q-1$ and $q+1$. The details of these parts of the algorithm are given in [5].

3 Algorithmic Improvements

In the course of rewriting EPISODE and EPISODEB as VODE, we took the opportunity to correct some of the known deficiencies in the algorithm, and also to introduce some improvements that have been inspired by the work of others. Below is a summary of the most significant of these changes, as far as they have been completed. More modifications of this type are in progress or are planned, and will be reported later.

3.1 Step/Order Resetting

At the conclusion of a successful step, VODE chooses a step size h' and an order q' for the next step, step $n+1$. Either or both of these may be the same as the values h and q used for step n. In the original EPISODE family and in LSODE, if either h or q is to be changed, the Nordsieck array z_n is adjusted accordingly (rescaled, and shortened or augmented) immediately following the decision to make the change. If interpolated values of $y(t)$, with t in (t_{n-1}, t_n), are called for and are then computed before proceeding, these may be based on the wrong interpolating polynomial and so be slightly in error, as was pointed out by Berzins [1]. On close inspection, one finds that EPISODE commits no such error if $q' = q+1$, because the augmented column in z_n is the zero vector. But in the case $q' = q-1$, the interpolant is incorrect by an amount proportional to the discarded last column $h^q y^{(q)}/q!$. In particular, in the case of the Adams method of any order or the BDF method with $q = 2$ ($q' = 1$), the interpolant fails to be continuous at t_{n-1}.

This error is fairly easy to correct, and it has been corrected in VODE, by postponing the adjustment of z_n until step $n+1$ is about to be taken, after any interpolations have already been done. This entails the use of some extra flags in the internal state data, and careful logic to handle all the combinations of cases. These include step size and/or order changes for step $n+1$ forced

by the user's stopping conditions or a change in method parameters.

With this change, the interpolated values are guaranteed to be continuous (within roundoff error), even in the presence of iteration error in the solution for y_n. This can be seen by writing

$$z_n = z_{n-1}A + e_n\ell$$

and noting that by construction the vector ℓ satisfies

$$\sum_{j=0}^{q} \ell_j(-1)^j = 0.$$

The value of the interpolant $y(t_{n-1})$ obtained from $z_n = [z_n^0, ..., z_n^q]$ is

$$y(t_{n-1}) = \sum_{j=0}^{q} z_n^j(-1)^j.$$

It follows that this value is the same as that gotten from $z_{n-1}A$, and this is the same as that from z_{n-1}, namely y_{n-1}. The presence of iteration error is reflected in the vector $e_n = y_n - y_{n(0)}$ and has no effect.

In general, the interpolant will not be C^1, and we have not made an attempt to make it C^1, as was done in [1]. Interestingly, in the case of an Adams method of order $q \geq 2$, the interpolant is always C^1, even with iteration error, by virtue of the identity

$$\sum_{j=1}^{q} j\ell_j(-1)^j = 0.$$

3.2 Initial Stepsize

In the initial version of VODE, an algorithm for selecting the initial stepsize was taken directly from the LSODE solver. It involves only the initial conditions, and the initial slope vector $\dot{y}_0 = f(t_0, y_0)$. Inspired by some of the work on this problem by Watts [19] and by Shampine [18], we have adopted a new algorithm which uses somewhat more data. Both algorithms start from the following premise: The desired initial stepsize h should satisfy

$$\|h^2\ddot{y}/2\|_{WRMS} = 1, \tag{3.1}$$

where WRMS denotes the weighted root-mean-square norm,

$$\|v\|_{WRMS} = \sqrt{(1/N)\sum_{i=1}^{N}(v^i/w^i)^2}$$

with weights w^i defined in terms of the relative and absolute tolerances supplied by the user, and where the second derivative is evaluated at the initial point. The basis for this premise is that this equation is identical to the local error test to be made at the end of the first step, except that the second derivative is estimated by a difference expression. The difficulty with the initial step selection is that the initial second derivative is not readily available.

The algorithm in LSODE (and the other ODEPACK solvers) approximates the components of weighted first-order principal error function, $(1/2)\ddot{y}^i/w^i$, by the squares of the components of the weighted zero-th order principal error function, $\dot{y}^i/w^i$. An adjustment has to be made to protect against the case where the initial $\dot{y}$ vanishes.

The algorithm in VODE, like those in [19] and [18], makes a more genuine attempt to approximate the initial second derivative vector. This involves an iteration, as follows: If a guessed value $\bar{h}$ is available, then the initial second derivative is approximated as

$$\ddot{y} = [f(t_0 + \bar{h}, y_0 + \bar{h}\dot{y}_0) - \dot{y}_0]/\bar{h}.$$

This value is inserted into equation (3.1) from the error test and that equation is solved for h. This is the next guess.

To get an initial guess for the iteration, we form some simple lower and upper bounds. First, in terms of the machine unit roundoff u and the initial output time $t_{\rm out}$, a reasonable lower bound is 100 times the roundoff level in the initial time t, or

$$h_L = 100u \max\{|t_0|, |t_{\rm out}|\}.$$

For an upper bound, we first take a fraction of the first output interval,

$$h_U = 0.1\,|t_{\rm out} - t_0|.$$

Then we reduce this value, if necessary, to guarantee that for every $i = 1, ..., N$ the corresponding change in y^i is not very large in either relative or absolute terms. Specifically we guarantee that h_U satisfies

$$h_U |\dot{y}_0^i| \leq 0.1 \, |y_0^i| + ATOL^i,$$

where ATOL is the input absolute tolerance array (or scalar). The initial guess for the iteration is taken as the geometric mean

$$\bar{h} = (h_L h_U)^{1/2}.$$

The convergence test on the iteration is quite loose, since we only need a crude approximation to the largest stepsize permissible on the first step. Thus we stop iterating if the new and previous values of h differ by less than a factor of 2. A maximum of 4 iterations is allowed. In the tests, no more that 2 have ever been observed. A few special situations are handled seperately (such as when $h_L > h_U$ or the norm of $\ddot{y}$ is very small), and the proper sign is attached to h at the end of the algorithm.

3.3 Jacobian Saving

We have added to VODE a device that has been used by several other authors of ODE solvers, which greatly improves its efficiency on many, if not most, stiff problems. This is an algorithm for saving and reusing the Jacobian matrix J, as it occurs in the Newton matrix $P = I - \gamma J$ (I is the identity matrix and γ is the scalar h_n/ℓ_1). In the form used in VODE, this algorithm is essentially the same as what is used in LSODES, the sparse Jacobian variant of LSODE. There the matrix P and its LU factorization are necessarily stored separately, and so it is natural to reuse an old value of P rather than recalculate it, if circumstances warrant this. A very similar algorithm is described in [8] and [17].

In both LSODES and VODE, a decision is made at the start of the step as to whether or not to update and refactor P. A decision to update may result from the step count (first step, or 20 steps

taken since the last update), the change in γ (by 30% or more), or by a failure of the corrector iteration to converge on a previous attempt at the current step. If the decision is made to update, then a saved value of J is used instead of recalculating it, if either

(a) no convergence failure occurred on this step and J is less than 50 steps old, or

(b) a convergence failure occurred with an old J and the relative change in γ since the last update to P exceeds 0.2.

In both cases, the idea is to isolate the situations in which P requires updating because of changes in γ and not changes in J.

This feature in VODE differs from the corresponding feature of LSODES in two ways. First, the VODE version is simpler, in that it saves J directly, whereas LSODES saves P and must recover a new value $\hat{P} = I - \hat{\gamma}J$ from an old value $P = I - \gamma J$ (with careful consideration of roundoff effects). Secondly, VODE requires considerable extra storage for this feature, for the saved copy of J as well as the factors of P, whereas LSODES must separately store the two sparse matrix arrays anyway. A user with a large problem (even with an ordering that gives a minimal bandwith), may well be unable to afford that extra storage without expensive overhead. For this reason, VODE provides an option to suppress the J-saving feature, i.e. to force an evaluation of J whenever the decision is made to update P, and to overwrite P on J (and the factors of P on P).

3.4 Linear System Relaxation For Stiff Systems

When the ODE problem is stiff, the nonlinear system (2.3) typically must be solved using a modified Newton iteration. In this subsection, we discuss a relaxation idea due independently to Petzold [16] and Burrage et al. [3] for speeding up the convergence of the nonlinear iteration. From (2.3), the nonlinear system has the general form

$$G(y) = y + a - \gamma f(t, y), \tag{3.2}$$

where $\gamma = h_n/\ell_1$, $a = y_{n(0)} - \gamma \dot{y}_{n(0)}$ and $t = t_n$. The Newton iteration with relaxation has the form

$$\begin{array}{l} \text{Solve } \hat{P}s(m) = -cG(y_{n(m)}), \\ \text{Set } y_{n(m+1)} = y_{n(m)} + s(m), \end{array} \quad m = 0, 1, 2, \cdots \tag{3.3}$$

where c is a scalar to be defined below, and $\hat{P}$ is an approximation to the Newton matrix

$$P = I - \gamma J(t, y_{n(0)}), \text{ with } J(t, y) = \frac{\partial f}{\partial y}(t, y).$$

Typically, $\hat{P} = I - \hat{\gamma}\hat{J}$, where $\hat{\gamma}$ and $\hat{J}$ are close to γ and J, respectively.

The iteration (3.3) will converge (under suitable conditions on G) if

$$\rho(I - c\hat{P}^{-1}P) < 1,$$

and the initial guess $y_{n(0)}$ is close enough to a root of (3.2). Here $\rho(A)$ denotes the spectral radius of a matrix A. The modified Newton iteration (3.3) normally converges at a linear rate, with $\rho(I - c\hat{P}^{-1}P)$ an estimate of the rate constant for the iteration. Thus, one would like to choose the scalar c so that it solves the minimization problem

$$\min_{c \in \mathbf{R}} \rho(I - c\hat{P}^{-1}P).$$

Since this is impractical in the general setting, we choose c to solve the following more tractable problem. Suppose $f(t, y) = Ay$ for a matrix A whose eigenvalues all have negative real part. In this case, $P = I - \gamma A$ and $\hat{P} = I - \hat{\gamma}A$. If λ is an eigenvalue of A, then

$$\bar{\lambda} = 1 - d\frac{\beta - \lambda}{\hat{\beta} - \lambda}$$

is an eigenvalue of $I - c\hat{P}^{-1}P$, letting $d = c(\gamma/\hat{\gamma})$, $\beta = 1/\gamma$ and $\hat{\beta} = 1/\hat{\gamma}$. In general, we do not know the eigenvalues of A, so we choose d to minimize

$$\max_{Re(\lambda)<0} \left| 1 - d\frac{\beta - \lambda}{\hat{\beta} - \lambda} \right|.$$

The solution of this problem is given by

$$d = d^* \equiv 2/(1 + \frac{\beta}{\hat{\beta}}),$$

which then gives the optimal value for c as

$$c = c^* = \frac{\hat{\gamma}}{\gamma} d^* = \frac{2\hat{\gamma}}{\gamma + \hat{\gamma}}. \tag{3.4}$$

Note that when $\gamma = \hat{\gamma}$ we have $c = 1$. Thus, c attempts to speed up the convergence only when the iteration is using an old Newton matrix approximation from a previous step.

4 A Fixed-Leading-Coefficient Variant

In this section, we discuss the fixed-leading-coefficient (FLC) form of the BDF methods and the relevant details of implementing them in the VODE solver. We have based our FLC solver on the work of Jackson and Sacks-Davis [15]. To be consistent with the fully variable-coefficient (VC) implementation already in VODE, we will use a Nordsieck form of the FLC methods. We first describe the FLC method formulas, and then discuss the necessary modifications to VODE for the FLC variant.

In the FLC methods, the prediction stage is based on the polynomial $\omega_n^P(t)$ of degree q or less that satisfies

$$\omega_n^P(t_{n-i}) = y_{n-i}, \ (i = 1, \cdots, q)$$
$$\dot{\omega}_n^P(t_{n-1}) = f_{n-1},$$

where $y_{n-i} \in \mathbf{R}^N$ for each i and $f_{n-1} = f(t_{n-1}, y_{n-1})$. Note that $\omega_n^P(t)$ satisfies the same interpolatory conditions as that for the VC methods. However, the interpolatory conditions for the FLC corrector polynomial differ from those satisfied by the VC corrector. The FLC corrector $\omega_n^C(t)$ of degree q satisfies the $q+2$ conditions

$$\omega_n^C(t_n) = y_n$$
$$\dot{\omega}_n^C(t_n) = f_n$$
$$\omega_n^C(t_n - ih_n) = \omega_n^P(t_n - ih_n), \ (i = 1, \cdots, q),$$

where $h_n = t_n - t_{n-1}$. Thus, the FLC corrector $\omega_n^C(t)$ interpolates $\omega_n^P(t)$ at evenly spaced past points rather than interpolating the computed values $\{y_{n-i}\}$ at the grid points $\{t_{n-i}\}$ as the VC corrector does.

The predicted values are given by

$$y_{n(0)} \equiv \omega_n^P(t_n) \text{ and } \dot{y}_{n(0)} = f_{n(0)} \equiv \dot{\omega}_n^P(t_n).$$

Using the above formulas, one can derive the relationship

$$\alpha_0(y_n - y_{n(0)}) + h_n(f_n - f_{n(0)}) = 0, \tag{4.1}$$

where $\alpha_0 = -\sum_{j=1}^{q} 1/j$. Equation (4.1) can be further reduced to the form

$$\alpha_0 y_n + \sum_{j=1}^{q} \alpha_{n,j}^* y_{n-j} + h_n f_n = 0.$$

This last equation differs from the analogous relationship for the VC methods in that the leading coefficient α_0 depends only on the order q, and not upon the previous grid points t_{n-i}.

The Nordsieck history array z_n is the same as that given in (2.1), with the prediction found using (2.2). To obtain z_n from $z_{n(0)}$, one uses the analogous formula for the VC methods, namely (2.4). However, the components of the row vector ℓ in the FLC version of (2.4) are different than those in the VC methods. In particular, $\ell_1 = -\alpha_0$ for the FLC methods, and depends only on the method order and not upon the previous grid points t_{n-i} as in the VC methods. As a result, the Jacobian matrix $P = I - \gamma J$ in the Newton iteration should change less with stepsize changes than in the VC formulas. Thus, the FLC methods have the potential to be more efficient in terms of Jacobian evaluations and factorizations than the VC methods on problems for which the stepsize varies smoothly, i.e. that class of problems for which the solver EPISODE is less efficient than GEAR.

When the order of the method is changed, the history array z_n must be adjusted. A decrease in order is handled in exactly the same way as for the VC methods. The array z_n must be modified

to become an array z_n^* which is based on the polynomial $\omega_{n+1}^*(t)$ of degree $q-1$ which is defined by

$$\omega_{n+1}^*(t_{n-i}) = y_{n-i},\ (i = 0, \cdots, q-2)$$
$$\dot{\omega}_{n+1}^*(t_n) = f_n.$$

This amounts to adding scalar multiples of the last column of z_n to itself, and can be written in the form

$$z_n^* = \bar{z}_n + \frac{h_n^q}{q!} y_n^{(q)} \cdot d,$$

where d is a row vector of the form $d = [d_0, d_1, d_2, \cdots, d_{q-1}]$, with $d_0 = d_1 = 0$ and $\bar{z}_n$ is the z_n array minus its last column.

An order increase for the FLC methods also requires a modification of the z_n array, unlike that for the VC methods. In this case, z_n must be modified to become an array z_n^* based on the polynomial $\omega_{n+1}^*(t)$ of degree $q+1$ defined by

$$\omega_{n+1}^*(t_{n-i}) = y_{n-i},\ (i = 0, \cdots, q)$$
$$\dot{\omega}_{n+1}^*(t_n) = f_n.$$

This can be accomplished in a way similar to the order decrease, and is given by

$$z_n^* = \tilde{z}_n + (y_n - y_{n(0)}) \cdot c,$$

where $\tilde{z}_n = [z_n, 0]$ and the row vector c is of the form $c = [c_0, c_1, c_2, \cdots, c_{q+1}]$, with $c_0 = c_1 = 0$. See [15] for more details.

5 Numerical Tests

The variable-coefficient solver VODE as described above was tested on several problems, of which two are given here. As mentioned in the Introduction, VODE was written according to the standard ODEPACK [10] interface, and its overall structure is similar to that of LSODE [11]. So we will

forego a detailed presentation of the solver and its usage. We have also implemented a second version of the VODE solver which uses the fixed-leading-coefficient form of the BDF methods. This version will be referred to as VODE-FLC.

Since both the linear system scaling and new initial h modifications were overall beneficial in improving the efficiency of the solver, we have elected not to specifically present test results regarding these enhancements. However, since the J-saving strategy is a user controllable option, the test results presented below will compare its effect on the solution of the test problems. For both problems considered below, we compare statistics for LSODE, VODE and VODE-FLC (with and without the J-saving strategy).

One important aspect of the implementation of the VC and FLC methods in the two versions of VODE is algorithmic tuning. For this preliminary study, we have elected to use the tuning recommended in [14] for VODE, and that used by Jackson and Sacks-Davis in [15] for VODE-FLC. Further testing may indicate modifications of these recommendations.

In the tables below, the following performance statistics will be helpful in comparing the individual solvers:

- NST = number of time steps
- NFE = number of f evaluations
- NJE = number of J evaulations
- NLU = number of LU factorizations of $P = I - \gamma J$
- R.T. = run time (in seconds) on a Cray-1.

5.1 Test Problem 1

Our first test problem is one that was used in [7]. It is a system derived from a 1-D diurnal kinetics-diffusion PDE system with two species. The PDEs have the form

$$\frac{\partial c^i}{\partial t} = \frac{\partial}{\partial z}\left[K(z)\frac{\partial c^i}{\partial z}\right] + R^i(c^1, c^2, t) \ (i = 1, 2),$$

$$K(z) = 10^{-8} e^{z/5},$$

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

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

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

$$k_3(t) = \begin{cases} \exp[-22.62/\sin(\omega t)], & \text{for } \sin(\omega t) > 0, \\ 0, & \text{for } \sin(\omega t) \le 0, \end{cases}$$

$$k_4(t) = \begin{cases} \exp[-7.601/\sin(\omega t)], & \text{for } \sin(\omega t) > 0, \\ 0, & \text{for } \sin(\omega t) \le 0, \end{cases}$$

$$\omega = \pi/43200,$$

with $30 \le z \le 50$ km, $0 \le t \le 432000$ seconds (5 days), and are discretized by finite differencing on a mesh of size 50. Homogeneous Neumann boundary conditions are used with polynomial initial conditions. See [7] for complete details on the problem definition and a discussion of the physical aspects.

The results of solving this problem using all three solvers are given in Table 5.1. The method flag MF in the table is an input for all three solvers. It indicates which ODE method is used, if the Jacobian matrix is treated as a dense or banded matrix, and whether or not the Jacobian is obtained by finite differences or analytically. The value |MF| = 25 means that the solver uses a stiff method, the Jacobian is banded, and it is obtained by finite differencing. A positive MF value means that the J-saving strategy is used, while a negative value means it is not. The RTOL value is the user-specified relative error tolerance. The absolute error tolerance ATOL is then 100*RTOL.

From the table, it is immediately apparent that the J-saving strategy is very beneficial on this

Solver	RTOL	MF	NST	NFE	NJE	NLU	R.T.
VODE	10^{-3}	-25	644	2322	236	236	2.27
VODE	10^{-3}	25	696	1406	19	275	1.95
VODE	10^{-5}	-25	1126	3552	298	298	3.59
VODE	10^{-5}	25	1122	2207	26	298	2.90
VODE-FLC	10^{-3}	-25	731	2131	207	207	2.13
VODE-FLC	10^{-3}	25	835	1377	25	224	1.90
VODE-FLC	10^{-5}	-25	1211	3093	271	271	3.17
VODE-FLC	10^{-5}	25	1252	2007	32	286	2.76
LSODE	10^{-3}	25	622	1875	185	185	1.88
LSODE	10^{-5}	25	2050	5511	470	470	5.64

Table 5.1: Results for Test Problem 1

problem in reducing the number of J evaluations, at the cost of a somewhat higher number of LU factorizations. Both versions of VODE are more efficient in terms of run time than LSODE at the tighter tolerance, while remaining competitive for the looser tolerance. The VODE-FLC is the overall winner with a 51% decrease in run time over LSODE for the tight tolerance, although in general VODE-FLC takes slightly more steps per run.

5.2 Test Problem 2

This second test problem is a 2-D version of the first one, and was also used in [7]. The PDEs have the form

$$\frac{\partial c^i}{\partial t} = K_h \frac{\partial^2 c^i}{\partial x^2} + \frac{\partial}{\partial z}\left[K_v(z)\frac{\partial c^i}{\partial z}\right] + R^i(c^1, c^2, t) \ (i = 1, 2),$$
$$K_h = 4 \times 10^{-6}, \text{ and } K_v(z) = 10^{-8} e^{z/5},$$

with $0 \le x \le 20$, $30 \le z \le 50$ km, $0 \le t \le 86400$ seconds (1 day), and the other terms defined as in the first problem. We discretize the PDEs on a uniform mesh of size 20×20, and use homogeneous Neumann boundary conditions and polynomial initial conditions. Again, see [7] for further details.

The results for this problem are given in Table 5.2. Here, the additional MF values of ± 24 correspond to a banded Jacobian obtained analytically. The RTOL and ATOL values are the same

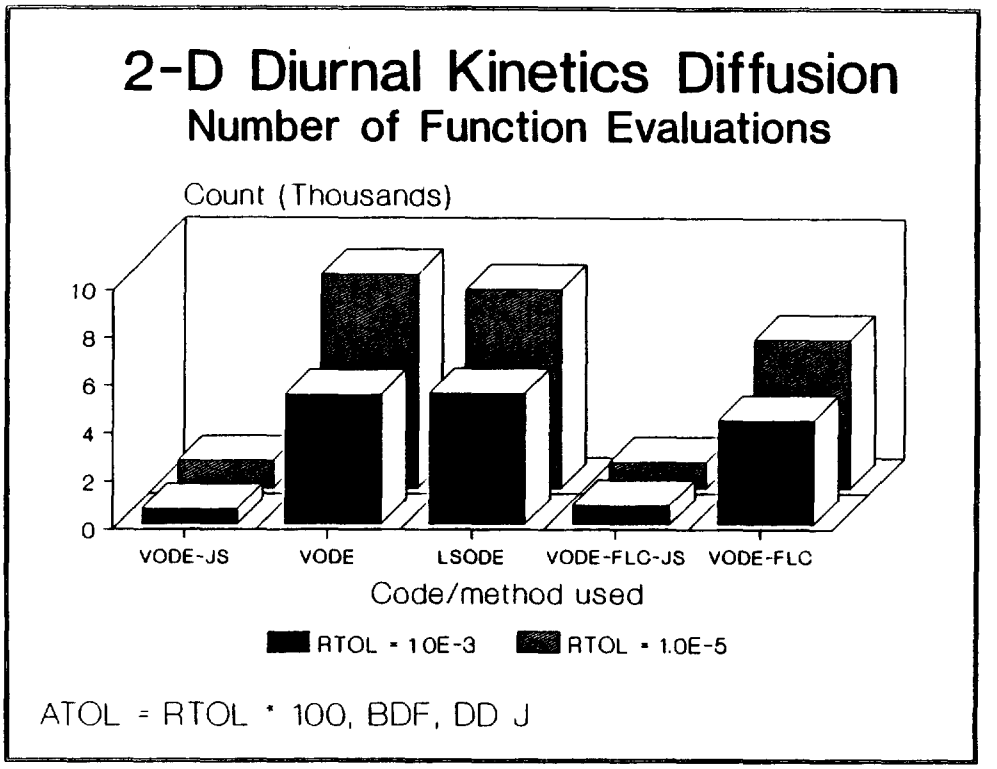
Solver	RTOL	MF	NST	NFE	NJE	NLU	R.T.
VODE	10^{-3}	-24	183	320	63	63	11.23
VODE	10^{-3}	24	185	323	4	63	11.19
VODE	10^{-3}	-25	183	5423	63	63	25.11
VODE	10^{-3}	25	185	647	4	63	12.07
VODE	10^{-5}	-24	331	589	103	103	19.04
VODE	10^{-5}	24	320	578	7	108	19.40
VODE	10^{-5}	-25	331	8932	103	103	41.86
VODE	10^{-5}	25	320	1145	7	108	21.08
VODE-FLC	10^{-3}	-24	230	307	50	50	9.43
VODE-FLC	10^{-3}	24	232	326	6	53	10.04
VODE-FLC	10^{-3}	-25	230	4357	50	50	20.56
VODE-FLC	10^{-3}	25	232	812	6	53	11.38
VODE-FLC	10^{-5}	-24	339	461	70	70	13.69
VODE-FLC	10^{-5}	24	342	469	7	73	14.14
VODE-FLC	10^{-5}	-25	353	6226	71	71	29.69
VODE-FLC	10^{-5}	25	375	1096	7	80	17.25
LSODE	10^{-3}	24	223	325	51	51	10.40
LSODE	10^{-3}	25	248	5480	63	63	29.64
LSODE	10^{-5}	24	482	708	98	98	20.99
LSODE	10^{-5}	25	481	8323	94	94	46.04

Table 5.2: Results for Test Problem 2

as used in the first problem. The values of NFE and R.T. in the case MF = ± 25 are also shown in the bar charts in Figure 5.1. It is clear that the J-saving strategy is also beneficial in reducing J evaluations and overall work for this problem. In addition, VODE and VODE-FLC are both more efficient than LSODE on this problem, with the biggest reductions in run time occuring when MF = 25 for LSODE and VODE-FLC. At the tighter tolerance, the run time is reduced by 63%!.

6 Conclusion

We believe that VODE accomplishes the objective of incorporating the ODE methods used in EPISODE and EPISODEB in a flexible solver package that conforms to the ODEPACK structure. More than that, it improves on the older codes in several ways, and includes some features that



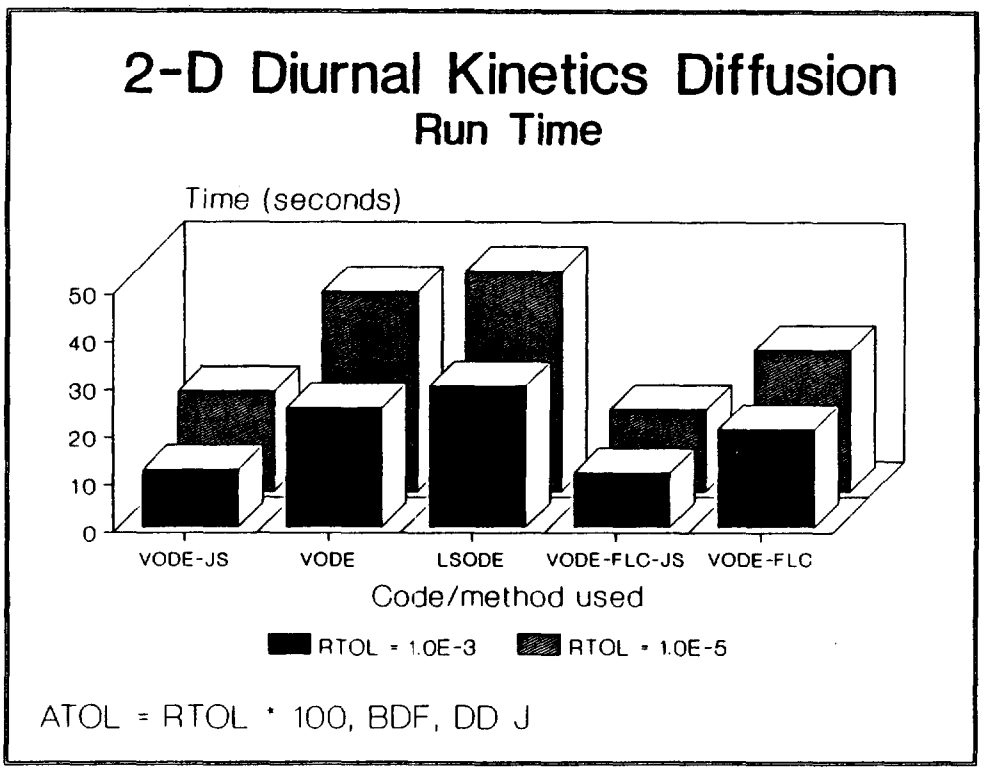


Figure 5.1: NFE and R.T. for Test Problem 2 (Difference-Quotient J)

enhance its efficiency, in some cases by a considerable amount. The relative speedup resulting from Jacobian-saving can increase dramatically as the cost of evaluating f and J increases, especially when J is obtained by difference quotients. The new initial step algorithm and the linear system relaxation each contribute a modest additional gain in speed.

The relative merits of LSODE, VODE, and the FLC variant of VODE are not entirely clear at this time. It seems to remain true (as for EPISODE vs GEAR) that on smooth problems, LSODE may be more efficient than VODE (either version). This can be attributed mainly to the impact of variable coefficients on the Newton matrix. On the other hand, the greater efficiency of VODE on many nonsmooth problems gives justification to the variable-coefficient methods. Furthermore, the FLC version of VODE seems to reduce the gap between VODE and LSODE on the smooth problems, while retaining the advantage for nonsmooth problems. More confident overall statements of this sort will have to await further refinement and broader testing.

Several further developments are planned for VODE. One is to improve the order selection to make it more reliable in the presence of large imaginary parts in the problem spectrum. Another is to study various heuristic (or tuning) issues and consider changes to these. Another is the development of a variant that includes preconditioned Krylov iteration methods for the algebraic system problem, in analogy with the LSODPK variant of LSODE [2].

References

[1] M. Berzins, *A C^1 Interpolant for Codes Based on Backward Differentiation Formulae,* Appl. Num. Math. 2 (1986), 109-118.

[2] Peter N. Brown and Alan C. Hindmarsh, *Reduced Storage Matrix Methods in Stiff ODE Systems,* Lawrence Livermore National Laboratory Report UCRL-95088, Rev. 1 (June 1987); submitted to J. Appl. Math. & Comp.

[3] K. Burrage, J. C. Butcher and F. H. Chipman, *An Implementation of Singly-Implicit Runge-Kutta Methods,* BIT 20 (1980), pp. 326-340.

[4] G. D. Byrne, A. C. Hindmarsh, K. R. Jackson, and H. G. Brown, *A Comparison of Two ODE Codes: GEAR and EPISODE,* Comp. & Chem. Eng. 1 (1977), 133-147.

[5] G. D. Byrne and A. C. Hindmarsh, *A Polyalgorithm for the Numerical Solution of Ordinary Differential Equations,* ACM Trans. Math. Softw. 1 (1975), 71-96.

[6] G. D. Byrne and A. C. Hindmarsh, *EPISODEB: An Experimental Package for the Integration of Systems of Ordinary Differential Equations with Banded Jacobians,* Lawrence Livermore National Laboratory Report UCID-30132 (April 1976).

[7] G. D. Byrne and A. C. Hindmarsh, *Stiff ODE Solvers: A Review of Current and Coming Attractions,* Journal of Computational Physics, Vol. 70, No. 1, May 1987.

[8] A. R. Curtis, *Solution of Large, Stiff Initial Value Problems – The State of the Art,* Numerical Software – Needs and Availability, edited by D. A. H. Jacobs (Academic Press, London, 1978).

[9] A. C. Hindmarsh, *GEAR: Ordinary Differential Equation System Solver,* Lawrence Livermore National Laboratory Report UCID-30001, Rev. 3 (December 1974).

[10] A. C. Hindmarsh, *A Tentative User Interface Standard for ODEPACK,* Lawrence Livermore National Laboratory Report UCID-17954 (October 1987).

[11] A. C. Hindmarsh, *LSODE and LSODI, Two New Initial Value Ordinary Differential Equations Solvers,* ACM SIGNUM Newsletter, Vol. 15, No. 4 (1980), 10-11.

[12] A. C. Hindmarsh, *ODEPACK, A Systematized Collection of ODE Solvers,* in Scientific Computing, R. S. Stepleman et al., Eds., North-Holland, Amsterdam, 1983, pp. 55-64.

[13] A. C. Hindmarsh and G. D. Byrne, *Applications of EPISODE: An Experimental Package for the Integration of Systems of Ordinary Differential Equations,* in Numerical Methods for Differential Systems – Recent Developments in Algorithms, Software, and Applications, L. Lapidus and W. E. Schiesser, Eds., Academic Press, New York, 1976, pp. 147-166.

[14] A. C. Hindmarsh and G. D. Byrne, *EPISODE: An Effective Package for the Integration of Systems of Ordinary Differential Equations,* Lawrence Livermore National Laboratory Report UCID-30112, Rev. 1 (April 1977).

[15] K. R. Jackson and R. Sacks-Davis, *An Alternative Implementation of Variable Step-Size Multistep Formulas for Stiff ODEs,* ACM Trans. Math. Softw., 6 (1980), 295-318.

[16] L. R. Petzold, *A Description of DASSL: A Differential/Algebraic System Solver,* Sandia National Laboratories Report SAND82-8637 (September 1982).

[17] D. E. Salane, *Improving the Performance of a Code for Solving Stiff Systems of ODEs,* Sandia National Laboratories Report SAND86-2159 (November 1986); to appear in Appl. Num. Math.

[18] L. F. Shampine, *Starting Variable-Order Adams and BDF Codes,* Appl. Num. Math. 3 (1987), 331-337.

[19] H. A. Watts, *HSTART – An Improved Initial Step Size Routine for ODE Codes,* Sandia National Laboratories Report SAND86-2633 (November 1986).