14.1 Variable Order Adams Method for Ordinary Differential Equations

A. Purpose

This collection of subroutines uses a variable order Adams method to solve the initial value problem

$$\left.\begin{aligned} \frac{dy_i}{dt} &= f_i(t, y_1, y_2, ..., y_{NEQ}) \\ y_i(t_0) &= \eta_i \end{aligned}\right\}, \; i = 1, 2, ..., NEQ, \quad (1)$$

or more generally

$$z_i^{(d_i)} = f_i(t, \mathbf{y}), \quad \mathbf{y}(t_0) = \eta_0, \quad i = 1, 2, ..., NEQ, \qquad (2)$$

where $\mathbf{y}$ is the vector $(z_1, z_1', ..., z_1^{(d_1-1)}, z_2, ..., z_{\text{NEQ}}^{(d_{\text{NEQ}}-1)})$, $z_i^{(k)}$ is the k^{th} derivative of z_i with respect to t, d_i is the order of the i^{th} differential equation, and η is a vector with the same dimension as $\mathbf{y}$.

If your derivatives are cheap to compute and you have a first order system, the Runge-Kutta routines in Chapter 14.2 will probably run faster. The RK routines will also provide a little more accuracy, and thus you may want to use them if the Adams method just barely fails to meet your accuracy needs.

B. Usage

Described below under B.1 through B.8 are:

B.1 Program Prototype, Double Precision

B.1.a The calling routine

The dimensioning parameters must satisfy the following constraints.

IFDIM $\geq k \times \text{NEQ} + 1$ (See the specification of IFDIM below for information on selecting k.)

IKDIM $\geq \text{NEQ} + 4$

ITDIM ≥ 4

IYDIM $\geq 2 \times L = 2 \times \sum_{i=1}^{\text{NEQ}} d_i = 2 \times$ (number of components in the $\mathbf{y}$ vector)

The subroutine arguments should be declared as follows:

EXTERNAL DIVAF, DIVAO

INTEGER NEQ, ITDIM, IYDIM, IFDIM, IKDIM, KORD(IKDIM), **IOPT**($\geq k$)
[k depends on options used (≥ 4).]

DOUBLE PRECISION TSPECS(ITDIM), **F**(IFDIM), **Y**(IYDIM)

Assign values to NEQ, TSPECS(), and Y(i), $1 \leq i \leq L$, where L = number of components in the $\mathbf{y}$ vector. Set KORD(1) = 0. Set the option vector, IOPT(), to specify an error tolerance. For example, to get the same absolute error tolerance on all equations:

IOPT(1) = 16
IOPT(2) = NEQ + 4
IOPT(3) = NEQ + 1
KORD(NEQ+4) = NEQ
F(NEQ + 1) = desired tolerance

If integrating a system of first order equations and no further options are desired, then terminate the option vector with

IOPT(4) = 0

If integrating a system of second order equations and no further options are desired, then set

IOPT(4) = 17
IOPT(5) = 2
IOPT(6) = 0

The subroutine DIVA may then be called as follows:

```
100 CALL DIVA (TSPECS, Y, F, KORD,
        NEQ, DIVAF, DIVAO, ITDIM,
        IYDIM, IFDIM, IKDIM, IOPT)
    if (KORD(1) .ne. 1) go to 100
```

(Here the computation is complete.)

B.1.b Program Prototype, DIVAF, Double Precision

```
SUBROUTINE DIVAF (TSPECS, Y, F, KORD)
DOUBLE PRECISION TSPECS(*), Y(*), F(*)
INTEGER KORD(*)
T = TSPECS(1)
Compute F(i) = f_i(T, Y), i = 1, 2, ..., NEQ.
RETURN
END
```

If **f** has complicated subexpressions which depend only weakly or not at all on **y**, such subexpressions need only be computed when KORD(1) = 1 and their values may be reused when KORD(1) = 2.

The use of options 7 or 18 described in Section B.6. requires additional action to be taken in DIVAF.

B.1.c Program Prototype, DIVAO, Double Precision

```
SUBROUTINE DIVAO (TSPECS, Y, F, KORD)
DOUBLE PRECISION TSPECS(*), Y(*), F(*)
INTEGER KORD(*)
Output results depending on the value of
  KORD(1), see Section B.5 for details.
If KORD(1) ≤ 2, the output increment, Δt,
  stored in TSPECS(3) can be changed.
If KORD(1) ≤ 3, or = 5, any output abscissas
  stored in TSPECS() can be changed.
RETURN
END
```

B.1.d Argument Definitions

TSPECS() [inout] An array used to store information about the independent variable. The first location available for options is TSPECS(5).

TSPECS(1) = current value of t, the independent variable. Must be initialized by the user, and is updated by the integrator.

TSPECS(2) = current value of h, the integration stepsize. Must be initialized by the user, and is updated by the integrator. The initial value selected for h is not critical to the accuracy or the efficiency of the integration.

TSPECS(3) = current value of Δt, the output increment. The user subroutine DIVAO is called with KORD(1) = 2 at $t = t_k$, where $t_k = t_{k-1} + \Delta t$, t_0 is the initial value of t, and k = 1, 2, ... Different increments between these output points can be obtained by changing TSPECS(3) whenever DIVAO is called with KORD(1) = 2. If no output of this type is desired, give TSPECS(3) a large absolute value. You must have $h \times \Delta t > 0$ in all cases.

TSPECS(4) = current value of t_f, the "final" output point. The user subroutine DIVAO is called with KORD(1) = 3 at $t = t_f$. If TSPECS(4) is not changed at this time, control is returned to the program which called the integrator with KORD(1) = 1. If TSPECS(4) is changed, the integration is continued.

Y() [inout] Vector of dependent variables, **y**. For a system of first order equations Y(i) is the i^{th} dependent variable, and must be initialized by the user for i = 1, 2,..., NEQ. Y(NEQ+i), i = 1, ..., NEQ is used to store previous values of Y(i) required for the integration process.

For a system of second order equations Y($2i-1$) is the i^{th} dependent variable and Y($2i$) is the derivative of the i^{th} dependent variable. They must be initialized by the user for i = 1, 2,..., NEQ. Y($2 \times (\text{NEQ} + i) - 1$) and Y($2 \times (\text{NEQ} + i)$) are used to save previous values of Y required for the integration.

For a system with mixed orders, values are stored in the order given for the vector **y** just below Eq. (2). This part of Y must be initialized by the user. Following these values is space to save previous values of **y** for use by the integration process.

F() [inout] Derivative value. The user should compute $f_i(t, \mathbf{y})$ and store it in F(i), i = 1, 2,..., NEQ, whenever DIVAF is called with KORD(1) = 1 or 2. Additional storage is required in F() for options and difference tables required for the integration. We recommend using the space starting at F(NEQ+1) for options. (The difference tables will be stored in the largest contiguous unused space in F().)

KORD() [inout] Vector used for flags to communicate with the user and to store integration orders. The user must set KORD(1) = 0 before the first call to DIVA for a new problem. Thereafter, DIVA sets KORD(1) to indicate what is to be done or what has happened. See "B.5 Additional Detail for the Argument, KORD" for further details. This section should be read before using the options that require examination of KORD. The first location available for options is KORD(NEQ+4).

NEQ [in] The number of differential equations in the system being solved.

DIVAF [in] Name of the subroutine supplied by the user. The name need not be 'DIVAF'. See "B.1.b Program Prototype, DIVAF, Single Precision" for the specifications of this subroutine.

DIVAO [in] Name of the subroutine supplied by the user. The name need not be 'DIVAO'. See "B.1.c Program Prototype, DIVAO, Single Precision" for the specifications of this subroutine.

ITDIM [in] Dimension of the TSPECS array. ITDIM ≥ 4 unless extra output points are specified by Option 5, in which case ITDIM must be > 4.

IYDIM [in] Dimension of the Y array. This should be twice the dimension of the dependent variable vector $\mathbf{y}$. *E.g.*, for a system of NEQ first order equations, IYDIM $= 2\times$NEQ; for a system of NEQ second order equations, IYDIM $= 4 \times$ NEQ.

IFDIM [in] Declared dimension of F(). The first NEQ locations of F() contain the current value of the derivative vector $\mathbf{f}$. Other locations in F() are used for options, as specified by the user. The largest contiguous block left in F() is used for difference tables.

At least one location must be used to provide the error tolerance associated with Option 16, which is required. We suggest location F(NEQ+1) be used for this purpose.

The suggested amount of space for difference tables is $(k-1)\times$ NEQ, where $k = \lfloor 7.50 - .5 \log_{10} \epsilon \rfloor$, and ϵ is the machine precision. For IEEE Arithmetic, we suggest $k = 11$ for single precision and $k = 16$ for double precision.

A storage size of $(k-1)\times$NEQ permits the subroutine to use differences of f up to $k-2$ in the predictor formula and $k-1$ in the corrector. Use of higher order differences, up to a limit depending on the machine precision, generally permits a problem to be solved in fewer integration steps. The subroutine requires difference table space of at least $5 \times$ NEQ and will not use more than $16 \times$ NEQ in single precision or $20 \times$ NEQ in double precision.

IKDIM [in] Declared dimension of KORD. This depends on the options used. With no options one must have IKDIM $\geq$ NEQ + 4.

IOPT() [in] A vector providing access to optional features of the integrator. Option 16 must be set by the user to specify an error tolerance. Defaults are such that you need not read about any of the other options. See Section B.6 for details and the following table for a quick overview.

Option	**Brief description**
0	No more options.
1	Skip initial point output.
-2	Resets all options to their nominal values.
2	Define a special output point while integrating.
3	Select interpolation, extrapolation or integration to final output point.
4	Increment for output based on step number.
5	Additional output points.
6	Interpolating G-stops and/or output every step.
7	Extrapolating G-stops.
8	Output when stepsize is changed.
9	Save information to reconstruct global solution.
10	Diagnostic output control.
11	Use when calling divadb to avoid diagnostics on uninitialized memory.
12	Stepsize control.
13	Reverse communication for $\mathbf{f}$.
14	Reverse communication for output.
15	Return after initialization.
16	Define error control (must be specified).
17	Direct integration of higher order equations.
18	Grouping of equations for derivative evaluation.
19	Integration order control.
20	Save estimated local error.
21	Set tolerance on G-stops.

B.2 Usage of G-Stops

The "G-stop" feature provides a means for the integration package to monitor user defined functions, $g_i(t,\mathbf{y})$, and return control to the user's code for special actions whenever one of these functions attains a zero value. This feature is activated by Options 6 and/or 7, and is supported by the subroutine DIVAG.

A distinction is made between *interpolating* and *extrapolating* G-stop functions. If the differential equations remain well behaved beyond a zero of a function $g(t,\mathbf{y})$ then g should be an interpolating G-stop function. Its zero can be detected by noting a sign change at some step of the solution algorithm and then using iteration and interpolation to locate the zero precisely. When $\mathbf{f}(t,\mathbf{y})$ has a discontinuity it is usually the case that there is a natural extension of $\mathbf{f}$ which is smooth and thus will allow the use of interpolatory G-stops. One can distinguish how $\mathbf{f}$ is to be computed in DIVAF via a flag that is set to one value before the G-stop, and to another value after the stop. Thus if $y' = -|y| + \sin t,\ y_0 = 1$, one could start with some flag $Y_{mul} = -1.$, and when getting a G stop for a change in sign on y, set $Y_{mul} = -Y_{mul}$ and restart the integration. One then computes $y' = Y_{mul} * y + \sin t$. Note that unlike the first form for y', the second form has a discontinuity only upon having the G-stop flagged at which time the integration is restarted.

If there is no convenient way to compute $\mathbf{f}$ (or a smooth extension of $\mathbf{f}$) beyond the G-stop, the function $g(t,\mathbf{y})$ must be tested more frequently and its zero must be found by searching from one side using iteration and extrapolation. This latter case requires more computation time, is slightly less accurate, and slightly less reliable.

Using Option 6 the user may specify that there are K6 interpolating G-stop functions, say $g_i^{(6)}(t, \mathbf{y})$, $i = 1, ...,$ K6, and using Option 7 the user may specify that there are K7 extrapolating G-stop functions, say $g_i^{(7)}(t, \mathbf{y})$, $i = 1, ...,$ K7. Code defining the interpolating (or extrapolating) G-stop functions must be in the user's DIVAO (or DIVAF) subroutine or in the equivalent part of the main program if reverse communication (Section B.4) is used.

The code is designed to find a zero crossing only if the sign change persists for at least an integration step. (This is to insure that computational noise does not result in multiple stops.) It will miss sign changes which persist for less than the current integration step, TSPECS(2). If it is necessary to locate such sign changes, the following procedure should be followed. Let d denote the lowest order total derivative of $\mathbf{g}$ with respect to t for which one knows that $(d^d/dt^d)\mathbf{g}$ does not change sign more than once in a given integration step. For some k, let $g_{k-j}^{(6)} = (d^j/dt^j)g$, $j = 0$, 1,..., d. Just above the statement with label 100 below, set some variable TT = TSPECS(1) (=value of TSPECS(1) at the end of the last complete integration step). When computing G6(i), if TSPECS(1) > TT (or when TSPECS(2) < 0, if TSPECS(1) < TT), and $k - d <$ NSTOP $\leq k$ then

1. If multiple G-Stops due to computational noise are not a concern, set G6(NSTOP) = 0, else
2. Set KG = 0 when defining TT. If for $k-d \leq$ NSTOP $< k$, $g_{NSTOP}^{(6)}$ has a different sign from the corresponding GT6 (see argument definitions below), or $k - d <$ NSTOP $\leq k$ and KG= NSTOP − 1, then set G6(NSTOP) = 0 and KG = NSTOP.

B.2.a Code for G-stops in subroutine DIVAO

Let M6 denote the maximum value K6 will have during a run. DIVAO must have the additional declarations:

```
INTEGER  NSTOP
DOUBLE PRECISION  G6(M6), GT6(M6)
SAVE  G6, GT6
```

The code in DIVAO may have the form:

```
    IF((KORD(1).EQ.6).OR.(KORD(1).EQ.7)) THEN
100    {Compute G6(i) = g_i^(6)(TSPECS(1), Y)
          for i = 1, ..., K6.}

     CALL DIVAG (TSPECS, Y, F, KORD,
        IFLAG, NSTOP, G6, GT6)

       IF (IFLAG .EQ. 1) THEN
{Take any other action you may want to take at the
   end of every step}
          RETURN
       ELSE IF (IFLAG .NE. 2) THEN
          IF (IFLAG .EQ. 4) GO TO 100
          IF ((IFLAG .EQ. 3) .OR. (IFLAG .EQ. 8))
   1         RETURN
{A zero of a G-stop function has been found for the
   current values of TSPECS(1) and Y(). If
   NSTOP > 0 it is a zero of g_NSTOP^(6), otherwise
   it is a zero of g_-NSTOP^(7).}
{Take whatever action is desired. If the action taken
   produces a discontinuous change in some f_i, then set
   KORD(1) = 0 to cause a restart of the integration.}
          RETURN
       END IF
   END IF
{Test KORD(1) for values other than 6 and 7 and do
   the selected output actions. See Section B.5}
   RETURN
```

B.2.b Code for G-stops in subroutine DIVAF

Let M7 denote the maximum value K7 will have during a run. DIVAF must have the additional declarations:

```
INTEGER  NSTOP
DOUBLE PRECISION  G7(M7), GT7(M7)
SAVE  G7, GT7
```

The code in DIVAF may have the form:

```
100  {Compute G7(i) = g_i^(7)(TSPECS(1), Y)
        for i = 1, ..., K7.}

     CALL DIVAG (TSPECS, Y, F, KORD,
        IFLAG, NSTOP, G7, GT7)

     IF (IFLAG .EQ. 4) GO TO 100
     IF (IFLAG .LE. 2) THEN
        {Compute F(i) = f_i(TSPECS(1), Y) for
           i = 1, ..., NEQ.}
     END IF
     RETURN
```

B.2.c Argument Definitions, G-stops

TSPECS(), Y(), F() [inout] Same as in B.1 above.

IFLAG [out] A flag telling the user what to do after a call to DIVAG.

= 1 Continue as if DIVAG were not called.
= 2 Check KORD(1) as one would do at the beginning of the subroutine if no G-stops were present (only has any real effect in DIVAO).

= 3 Return to the integrator. (If in DIVAF and NSTOP $\neq 0$, a G-stop was found, but further checks must be made to see if there is preceding output.)

= 4 Compute values of G6() if in DIVAO or of G7() if in DIVAF; then call DIVAG again.

= 5 A G-stop has been found, and NSTOP gives its index. (If NSTOP < 0, an extrapolating G-stop with index $-$NSTOP has been found.)

= 6 Same as 5, but requested accuracy not met.

= 7 Same as 5, but there is probable error in computing G.

= 8 Fatal error of some type. An error message has been printed, and the program will be stopped if a return is made to the integrator.

NSTOP [out] Index of the G-stop. See IFLAG = 5, 6, and 7 above. After the call to DIVAG, if IFLAG = 4, and NSTOP $\neq 0$, only the G with index NSTOP need be computed.

G6() [in] An array containing current values of the functions whose zeros are to be found using interpolatory G-stops. The user must compute G6 in DIVAO as indicated above.

G7() [in] As for G6, except for extrapolatory G-stops. G7 is computed in DIVAF.

GT6() [inout] An array containing previous values of G6. The user should not change its values, except for the following.

1. To turn off testing for G6(k). This can be done by setting G6(k) = GT6(k) = 0.
2. When redefining how G6(k) is computed. In this case, it is permitted to store in GT6(k) any value that has the same sign as would be expected for the new G function it it were computed slightly before the current time. (GT6(k) = 0 just after an indication of a G-stop for G6(k). While GT6(k) is 0 no G-stop is indicated, but GT6(k) is replaced by G6(k) after the g's are computed.)

GT7() [inout] As for GT6, except used with G7.

KORD() [inout] Vector used for flags to communicate with the user. See Section B.5 for details.

B.3 Saving the Solution

If Option 9 has been specified, DIVAO is called with KORD(1) = 9 whenever information necessary to reconstruct the solution should be saved. Besides making this call when required to maintain sufficient accuracy in the saved solution, this call is made if, before a return to the integrator from DIVAO, the user sets KORD(1) ≤ 0. (A restart of the integration results if KORD(1) = 0, and a return to the program which called the integrator results if KORD(1) < 0. See below, "B.7, Other Options.")

The contents of the common block DIVASC (SIVASC in the case of single precision), the base values for Y(), the difference tables, and the integration orders (which are stored as indicated below) must be saved in order to reconstruct the solution. The common block contains the following, in the order given.

TN The base value of the independent variable.

XI() XI(k) = $t_n - t_{n-k}$, where t_n = TN, and t_{n-k} was the value of TN k steps back. XI has dimension KDIMDT, where KDIMDT=16 for single precision and=20 for double precision.

IOPST Reserved for use in case of stiff equations.

KORDI If all differential equations have the same order, KORDI is that order, otherwise it is not used.

KQMAXD Maximum integration order used for stiff equations (when implemented).

KQMAXI Maximum integration order used for equations which are not stiff.

LDT Flag giving current state of difference tables. Whenever DIVAO is called with KORD(1)=9, LDT=1 indicating that the differences are updated to the proper values for the current base point TN.

MAXDIF Maximum order derivative of F to be computed. Ordinarily this will equal 0.

MAXINT Maximum number of integrations to be performed. Ordinarily this will be the same as the order of the differential equation with the largest order.

NKDKO If all differential equations are of the same order, NKDKO=0, otherwise orders of the differential equations are stored starting in KORD(NKDKO).

NTE Total number of differential equations being integrated.

NYNY The base values for Y() are stored in Y(NYNY), Y(NYNY + 1),...,Y(NYNY+k−1), where k is the total order of the system. (k = NTE$\times$ KORDI if NKDKO=0.) NYNY=k+1.

NDTF Difference tables are stored starting in F(NDTF)

NUMDT Number of differences for each equation.

Base values for Y() are stored as indicated just above. Difference tables are stored in F(NDTF), F(NDTF+1),...F(NDTF+NEQ$\times$NUMDT − 1), and the integration orders are stored in KORD(4), KORD(5), ..., KORD(NTE+3). TN and XI() have the same type as F. Note that TN, XI(), KQMAXI, the integration orders, the base values for Y(), and the difference tables are the only variables that change from one time to the next. Thus the other variables need only be saved once.

If one requests integration restarts when saving the solution (because of discontinuities in some f_i, for example), then the value of TSPECS(1) should be stored in addition to the variables indicated above. In describing how to reconstruct the solution we assume the values of TSPECS(1) have been saved. If not, simply use TN for TSPECS(1).

To reconstruct the solution at a given point t, find that value of j such that $t_{j-1} < t \leq t_j$ (or $t_{j-1} > t \geq t_j$ if the stepsize is negative), where t_j is the value of TSPECS(1) at the j^{th} time the solution was stored. Store the saved values from the common block, the base Y(), the difference tables, and the integration orders into the locations they occupied originally. Set TSPECS(1) = t, TSPECS(2) = $t_j - t_{j-1}$, and

```
CALL DIVAIN (TSPECS, Y, F, KORD)
```

Y() and F() will then contain the interpolated values of the solution and derivative at the specified value of t.

If the integrator is being used while reconstructing a solution from another integration, then one should introduce a new subroutine which is the same as DIVAIN (or SIVAIN) except for the subroutine name and the common block name DIVASC (or SIVASC), and call it in place of the above call. Names used for retrieved values should, of course, be different from names being used in the current integration.

If one is interested in interpolating only for the variables associated with some of the equations, one need store only the corresponding base values of Y, difference tables, and integration orders.

B.4 Program Prototype, Reverse Communication, Double Precision

Declare the subroutine parameters and initialize their values as described in Section B.1.a with the following exceptions:

(a) To use reverse communication for derivative computation set Option 13. Then DIVAF need not be declared EXTERNAL but some name must occupy the sixth position in the CALL to DIVA.

(b) To use reverse communication for output set Option 14. Then DIVAO need not be declared EXTERNAL but some name must occupy the seventh position in the CALL to DIVA.

(c) The array IOPT() must be dimensioned sufficiently large to handle the selected options.

Remark: In a Fortran system that checks argument typing carefully, it may be necessary that the sixth and seventh argument in the CALL to DIVA always be names of actual subprograms that are accessible at run time.

```
    CALL DIVA(TSPECS, Y, F, KORD,
       NEQ, DIVAF, DIVAO, ITDIM,
       IYDIM, IFDIM, IKDIM, IOPT)
100 CONTINUE
    CALL DIVAA (TSPECS, Y, F,
       KORD, DIVAF, DIVAO)
```

```
   IF (KORD(2) .LT. 0) THEN
      If (KORD(1) .NE. 1) GO TO 100
   ELSE IF(KORD(2) .EQ. 0) THEN
{This point can only be reached if Option 13 has been
   selected. Do what was specified in Section B.1.b
   for subroutine DIVAF.}
      GO TO 100
   ELSE IF(KORD(2) .GT. 0) THEN
{This point can only be reached if Option 14 has been
   selected. Do what was specified in Section B.1.c.
   for subroutine DIVAO.}
      GO TO 100
   END IF
   (Here the computation is complete.)
```

B.5 Additional detail for the argument KORD

KORD(1) and KORD(2) are used for communication between the user's code and the integration package. KORD(3) is used with Option 18. KORD(4), ..., KORD(NEQ+3) are used to store integration orders. Other locations in KORD() may be used with various options.

The integrator transfers control to the user by: returning, calling DIVAF, or calling DIVAO. The user may use the same name for DIVAF and DIVAO, and there are options which permit the user to get a return in place of a call to DIVAF and/or DIVAO. The value of KORD(2) indicates which of the above cases should apply if there is a chance for confusion. (If DIVAF and DIVAO are different and the optional return feature is not used, then KORD(2) need never be examined.) KORD(1) gives the specific information on what is to be done or what has happened as indicated below. Also see Other Options on page 10 for values of KORD that can be set by the user's code.

RETURN (KORD(2) = −1)

KORD(1) = 1 TSPECS(1) = TSPECS(4), presumably the integration is finished. If the integrator is called with a new value for

TSPECS(4), the integration continues; if called with TSPECS(4) unchanged, a diagnostic results.

KORD(1) > 1 A diagnostic message with values of key variables has been printed. Except for special situations we recommended simply calling the integrator. If KORD(1) > 10 this will result in the program being stopped. For details see Section E.

DIVAF (KORD(2) $= 0$)

KORD(1) $= 1$ Y() has been predicted. Compute $\mathbf{f}(t, \mathbf{y})$ and store in F().

KORD(1) $= 2$ Y() has been corrected. Compute $\mathbf{f}(t, \mathbf{y})$ and store in F(). The fact that $\mathbf{f}$ was last computed for the same value of t and a nearby value of $\mathbf{y}$ can frequently be used to reduce the work in computing $\mathbf{f}$.

KORD(1) > 2 This occurs only if the option for extra equations or stiff equations is used, see IOPT below.

DIVAO (KORD(2) $= 1$)

KORD(1) $= 1$ Initial point, output results (if desired) and return.

KORD(1) $= 2$ Output results. See description of TSPECS(3).

KORD(1) $= 3$ Output results. See description of TSPECS(4).

KORD(1) > 3 This occurs only if some special output option is specified; see IOPT below. In brief: $= 4$, step number; $= 5$, extra output point; $= 6$, end of step (or G-stop); $= 7$, G-stop; $= 8$, step change; and $= 9$, solution dump.

B.6 Setting options using the array IOPT()

Any number of options can be stored in the array IOPT() in any order. Each option consists of an identifying code plus 0, 1, or 2 integer arguments. Option codes are summarized in the table on page 14.1–3. An option is selected by placing its code in the array IOPT() with its required arguments, if any, in the immediately following array locations. Thus if an option has two arguments, its code and its arguments occupy three locations in IOPT().

The option codes, with space for their arguments as required, must be stored as a contiguous sequence beginning in IOPT(1) and ending with the option code 0. If the same nonzero option code appears more than once, the last occurrence will be effective. If the option code 1 or codes 3–20 are set negative, it has the effect of setting values for that option to their default values. Space for the option arguments must be set aside even though the arguments aren't referenced for negative values of the option.

Depending on the option, an argument may be an integer datum or it may be a pointer, *i.e.* an index, into the array F(), TSPECS(), or KORD(). Any option that uses an argument as a pointer to another array requires that the storage referred to be distinct from the storage pointed to for any other option, and from the storage already indicated as required for that array in the preceding declarations. The dimension of the array which is pointed to must be increased to the size required by the option.

The option, 16, is required to be selected. It is used to specify an error tolerance.

Option Code	**Description**
0	(No argument) No more options. The list of requested options must always be terminated by this zero option.
1	(No argument) Skip initial point output. The call to DIVAO with KORD(1) $= 1$ is not to be made.
−2	(No argument) Sets all options to their nominal values. However, if other nonzero option codes follow this one in the array IOPT(), they will be effective, as usual.
2	(Args: K2,L2) Available only when calling DIVAOP during an integration, see Section B.7. This call will give the next output at TSPECS(1) $=$ FOPT(L2) with KORD(1) $=$ K2. This is useful when one wants to get the output from TSPECS(3) based at a certain point. If K2 is 3, then one should set TSPECS(4) to the output point you want, set L2 to 4, and pass TSPECS to DIVAOP for the argument described as FOPT. In this case the next output point is only changed if TSPECS(4) precedes other output points.
3	(Arg: K3) Select interpolation, extrapolation or integration to the final output point. When DIVAO is called with KORD(1) $= 3$, results at TSPECS(4) are obtained as follows. ($\hat{t}$ $=$ value of TSPECS(1) at the last point at which $\mathbf{f}$ was computed, t_f $=$ TSPECS(4), h $=$ TSPECS(2).) Reverse all inequalities if $h < 0$. K3 $= 0$ Interpolation $\hat{t} - h < t_f \le \hat{t}$; $= 1$ Integration $\hat{t} = t_f$; $= -1$ Extrapolation $\hat{t} < t_f$. Interpolation is the default and should ordinarily be used unless there is a problem in computing derivatives beyond the end point.
4	(Arg: K4) Increment for output based on step number. DIVAO is called with KORD(1) $= 4$ every K4 steps. The default is K4 $= 500000$.

5 (Arg: K5) Additional output points. If K5 $>$ 4, then TSPECS(5), ..., TSPECS(K5) are the additional output points. Whenever the integration passes one of these points, DIVAO is called with KORD(1) = 5, KORD(3) = index in TSPECS() to which the output corresponds, and all other variables set to their appropriate values, *i.e.*, TSPECS(1)= TSPECS(KORD(3)), and Y and F are interpolated at TSPECS(1).
If K5 $<$ 0, let NK5 = $-$K5. Then starting in KORD(NK5) one should have stored a vector of the form i_1, j_1, k_1, i_2, j_2, k_2,..., where the end of the vector is flagged by setting $i_n = 0$. The triple (i_n, j_n, k_n) indicates output points are contained in TSPECS(i_n),TSPECS($i_n + 1$), ..., TSPECS(j_n) and the way results are computed depends on k_n in the same way as K3 is used for option 3. The user can indicate that output for the n^{th} block is to be turned off by setting $i_n = -i_n$. One must always have $j_{n-1} < i_n \leq j_n$ (except for the case $i_n = 0$), and $i_1 > 4$. Output is indicated in the same way as for the case K5 $>$ 0.
If K5 = 0, this option is turned off.
When several output points coincide, the one with KORD(1) = 2 is given first, then that for KORD(1) = 5 with smaller values of KORD(3) given first, and finally that for KORD(1) = 3.

6 (Arg: K6) Interpolating G-stops and/or output every step. K6 = -1 gives a call to DIVAO with KORD(1) = 6 at the conclusion of every step. K6 = 0 turns this option off. K6 $>$ 0 indicates that the user has K6 interpolatory G-stop functions. See Section B.2. K6 $\neq$ 0 calls DIVAO at the initial point also.

7 (Arg: K7) Extrapolating G-stops. K7 $>$ 0 indicates that the user has K7 extrapolating G-stop functions. See Section B.2. K7 = 0 turns this option off.

8 (No argument) DIVAO is called with KORD(1) = 8 whenever the stepsize is changed.

9 (Arg: K9) Save global solution. DIVAO is called with KORD(1) = 9 whenever it is time to save information necessary to reconstruct the solution globally. See "B.3 Saving the Solution," above. With K9 = 1 if error estimates are sufficiently small, then the difference tables at a saved point may not span all the way back to the previously saved point. With K9 = -1 difference tables will span back to the previously saved point, with an exception possible if the error estimate is zero. K9 = 1, is recommended if the error control is scaled properly for all equations.

10 (Args: K10,L10) Diagnostic output control.
K10 = 0 (nominal value) no diagnostic output.
K10 $>$ 0 prints (as error messages) storage usage in TSPECS, Y, F, and KORD, and then gives internal output useful for tracing problems for K10 passes through the logic for estimating errors, selecting integration orders, etc. (Typically this will be slightly less than K10 integration steps.)
K10 = -1 just gives the storage usage part of this print.
L10 = 0 gives the internal output for all equations.
L10 $>$ 0 means a vector i_1, i_2, ..., i_n specifying equations for which output is desired is stored starting in KORD(L10). If $i_k > 0$, output is given only for equation i_k, if $i_k < 0$, output is given for equations $|i_{k-1}| + 1, ..., |i_k| (i_0$ assumed $= 0)$. One must have $0 < |i_1| < |i_2| < ... < |i_n|$ where n is the first index for which $|i_n| \geq$ NEQ. One specifies the end of this option list by setting $i_n \geq$ NEQ. (Use $>$ if you don't want the output for the last equation.) For example, if you you have 500 equations and want output for the first 4 and equations 10–15, you could use -4, 10, -15, 5001.

11 (No argument) When calling divadb using this option will initialize some variabless that are printed that may otherwise not be set.

12 (Arg: K12) Stepsize control. If K12 $>$ 0, then parameters which affect the choice of stepsize are defined by the user. (Replace F with FOPT below if DIVAOP is used.)
F(K12) = HINC = nominal factor for increasing the stepsize. After the integration is started, the stepsize when increased is ordinarily increased by a factor of HINC. $9/8 \leq$HINC ≤ 4, and nominally HINC = 2. Choosing HINC (or HDEC below) closer to one will generally result in fewer function evaluations to get a given accuracy, but will increase the internal overhead of the integrator. The best choice is problem dependent and depends primarily on the expense of computing derivatives. Values close to one also tend to make the global error a more regular function of the local error tolerance.
F(K12+1) = HDEC = nominal factor for decreasing the stepsize. After the integration is started, the stepsize when decreased is ordinarily decreased by a factor of HDEC. $1/4 \leq$ HDEC $\leq 7/8$, and nominally HDEC = $\frac{1}{2}$. Ordinarily one will probably want HINC $\times$ HDEC ≈ 1.
F(K12+2) = HMIN = the absolute value of the minimum stepsize permitted after the integration is started. The integrator will give a diagnostic if it cannot maintain the requested error with a stepsize of HMIN. The nominal value for HMIN is very close to zero.
F(K12+3) = HMAX = the absolute value of the max-

imum stepsize. The integrator will not take a step greater than HMAX. The nominal value for HMAX is a very large number.

A zero value for any of these parameters will result in the current value being left unchanged.

If K12 < 0, then DIVAO will be called at the end of every step with KORD(1) $= 8$, at which time the user can change TSPECS(2) (the integration stepsize). When K12 < 0, the integrator does not check the integration error, and does not alter the user's choice of stepsize.

13 (No argument) Use reverse communication for the computation of $\mathbf{f}$. The integrator will return to the user instead of calling DIVAF when it is time to evaluate $\mathbf{f}$. See "B.4 Program Prototype, Reverse Communication, Double Precision," above.

14 (No argument) Use reverse communication for output. The integrator will return to the user instead of calling DIVAO whenever DIVAO would otherwise be called. See "B.4 Program Prototype, Reverse Communication, Double Precision," above.

15 (No argument) Return after initialization. After the call to DIVA, control is returned to the user as is done for the two above options, and DIVAA should be called to do the actual integration. Reverse communication need not be used.

16 (Args: K16,M16) Define error control (must be specified). Let

$L_k = \text{KORD}(\text{K16}+k-1)$, and $\varepsilon_k = \text{F}(\text{M16}+k-1)$. K16 thus points to the location in KORD containing L_1, where the L's are used to define groups and types of error control; M16 points to the location in F containing ε_1, where the ε's define the magnitude of the local error tolerance. Note that for differential equations with order $d > 1$, the error control is on $y^{(d-1)}$.

To get the same absolute error tolerance for all equations, set $L_1 = \text{NEQ}$ and $\varepsilon_1 =$ the absolute local error tolerance desired.

If different absolute error tolerances are desired for different equations, set $\varepsilon_k =$ the tolerance desired for the k^{th} group, and $L_k =$ the index of the last equation in the k^{th} group.

Other types of error control are available and are indicated by giving L_k a non-positive value. Example: $L_1,\ L_2, ... = 2,\ 6,\ -3,\ 10,\ 12,\ 0,\ 15$ would indicate an absolute error test with tolerance ε_1 on equations 1 and 2, an absolute error test with tolerance ε_2 on equations 3 to 6, an $L_k = -3$ type of relative error test (defined below) with base tolerance ε_3 and factor ε_4 on equations 7 to 10, an absolute error test with tolerance ε_5 on equations 11 and 12, and no error testing on equations 13 to 15 (provided $\varepsilon_6 = -1.0$).

To describe the general case, let k either be 1 or be such that $L_{k-1} > 0$. The index of the first equation in the group defined for this value of k is

$$\hat{I}_k = \begin{cases} 1 & \text{if } k = 1 \\ L_{k-1} + 1 & \text{if } k > 1 \end{cases}$$

and the index of the last equation in the group is

$$I_k = \begin{cases} L_k & \text{if } L_k > 0 \\ L_{k+1} & \text{if } L_k \le 0. \end{cases}$$

One must have $I_k \ge \hat{I}_k$, and the value of k used in defining the next group will be one greater than the subscript of L which gives the value of I_k.

$L_k > 0$ Use an absolute error tolerance of ε_k for the group.

$L_k = 0$ Do not check errors for the group (to prevent accidental use of this option one must also set $\varepsilon_k = \varepsilon_{k+1} = -1.0$). (The value in ε_{k+1} is not checked.)

$L_k < 0$ Use a relative error test, with base tolerance (usually fixed) in ε_k and the relative error factor in ε_{k+1}. The actual absolute local error tolerance used is then $\varepsilon_k \times \varepsilon_{k+1}$.

$L_k = -1$ User stores the relative factor in ε_{k+1}.

$L_k \le -2$ User initializes ε_{k+1}; the integrator updates it on each step to equal

$$\max \left[\varepsilon_{k+1}, \frac{|h|}{I_k - \hat{I}_k + 1} \sum_{j=\hat{I}_k}^{I_k} |F(j)| \right] \times \rho_{|L_k|-1},$$

$$\text{where} \quad \begin{array}{ll} \rho_1 = 1 & L_k = -2 \\ \rho_2 = 15/16 & L_k = -3 \\ \rho_3 = 3/4 & L_k = -4 \\ \rho_4 = 1/2 & L_k = -5 \end{array}$$

One must have $L_k \ge -5$. The setting $L_k = -3$ is suggested unless the user has a reason for a different choice.

The relative error test built into the integrator is based on making the error relative to the change in the solution over a single step. To assist in the initial choice of ε_{k+1} we offer the following (equivalent) suggestions.

1. $\varepsilon_{k+1} \approx e/\varepsilon_k$, where e is the maximum absolute error per step, over the first few steps for any Y in the current group.
2. $\varepsilon_{k+1} \approx$ the maximum absolute change for any Y in the group in a single step over the first few steps. (Substitute $h \times$ F for the absolute change in Y if you find it more convenient.) In making

this estimate one should ignore the fact that h (and hence the change in Y) is quite small initially because of the need to start the integration.

17 (Arg: K17) Direct integration of higher order equations. If K17 > 0, then all differential equations are of order K17. (If this option is not used, it is assumed that K17 $= 1$.) If K17 < 0, then the order of the i^{th} equation should be stored in KORD($-$K17 $+ i - 1$). (The description of KORD mentions locations in KORD that are reserved.) If this option is used, Y() contains all derivatives with order less than that of the differential equation, stored as indicated for $\mathbf{y}$ in Eq. (2) under "A. Purpose." Y() must have dimension $\geq 2\times$(total order of the system). Note also that the error control is on $z^{(d-1)}$ and not on z.

18 (Arg: K18) Grouping of equations for derivative evaluation. If K18 $= k > 0$, then a grouping of equations (frequently useful for variational equations) is indicated in KORD(k), KORD($k+1$), ... Let $j_i =$ KORD($k + i - 1$), for $i = 1, 2, ..., n$ where $j_n =$ NEQ, and $|j_{k+1}| \geq |j_k|$. Equations are grouped into: equations 1 to $|j_1|$, equations $|j_1| + 1$ to $|j_2|$, ..., equations $|j_{n-1}| + 1$ to $|j_n|$. Predicted derivatives for any group are only computed after the corrected values for the previous group have been computed. For equations after the first group, DIVAF is called with KORD(1) $= 3$ and KORD(3) $= i$, when predicted values of the derivatives for equations $|j_i| + 1$ to $|j_{i+1}|$ are to be computed. (One can have i incremented without any active equations in a group by setting $j_i = 0$.) Usually when this option is used, only one extra set of equations is introduced. When there are extra equations KORD(3) will contain the number of extra groups when DIVAF is called with KORD(1) $= 2$. At this time corrected derivatives should be computed for all equations, presumably taking advantage of the fact that the values of certain subexpressions will not have changed.

19 (Arg: K19) Integration order control. If K19 > 0, then the user has grouped equations to specify some kind of control over the integration orders selected by the integrator. Let $L_k =$ KORD(K19 $+ k - 1$), and define I_k, and $\hat{I}_k$ as for Option 16. Options available are

$L_k > 0$	different equations in the group may be integrated with different integration orders;
$L_k = 0$	integration orders for the group are not to be changed;
$L_k = -1$	orders may be changed, but all equations are integrated with the same order method;
$L_k = -2$	same as $L_k = -1$, except order is not to be increased; and
$L_k = -j$	($j > 2$) same as $L_k = -1$, except order is not to be decreased and is not to be increased greater than j.

20 (Arg: K20) Save estimated local error. If K20 > 0, the integrator stores the estimated local error committed for equation i into F(K20$+i-1$), $i = 1, 2,...,$ NEQ whenever corrected values of Y are computed.

21 (Arg: K21) The user is to store a value for TOLG in F(K21). This value is used as the last argument, TOL, to DZERO when iterating for a 0 when locating a zero for G-stop, see Chapter 8.1. The default is to use TOLG $= 0$, which locates the 0 as accurately as possible. A positive value isolates the 0 to within an interval of length TOLG. A negative value iterates until a value of $|g_i| < |$TOLG$|$ is found. The negative value should only be used if all of your G functions are scaled in the same way. A positive value is recommended if your G functions are a bit noisy on short intervals. For example one might use TOLG = the minimum value for $\max|\varepsilon_i(t)/f_i(t)|$ where the maximum is computed for t in the interval of integration, the minimum is computed over all equations, and $\varepsilon_i(t)$ is the absolute error tolerance applied to the i^{th} equation at time t. With such a TOLG, the error in locating the G-stop will be on the same order as the usual integration error, and when the G-stops are time consuming to compute this can save significant time.

Any option, once set, remains set until turned off. (This has implications only if one is running a sequence of problems, or is using the call to DIVAOP described below in "B.7 Other Options.")

B.7 Other Options

From either DIVAF or DIVAO, one can get an immediate return to the program which called the integrator by setting KORD(1) < 0. (If KORD(1) is set < 0 from inside DIVAO and option 9 is being used to save the solution, then DIVAO is called with KORD(1) $= 9$ before the return is made.) Of course, one must be prepared to treat negative values of KORD(1) after the call to the integrator if this option is used. If the integrator is called after such a return, computation continues just as if the value of KORD(1) had not been changed in the first place.

In DIVAO, one can cause the integration to be restarted at the current value of TSPECS(1) by setting KORD(1) $= 0$. This feature should be used if the current output point corresponds to a discontinuity in the derivatives.

Except at the initial point of an integration one can interpolate to an arbitrary value of the independent variable when DIVAO is called. Let TINT() be an array of length 2 with the same type as TSPECS. Set TINT(1) = the value of t to be interpolated to, and TINT(2) = d, where d has the same sign as TSPECS(2) and $|d|$ is the greatest distance back from the current base time that one wants to permit for interpolation. Then, call DIVAIN as indicated above in Section B.2 with TINT substituted for TSPECS.

B.7.a Changing Options

Options can be set up or changed any time and from any place with the following call.

CALL DIVAOP (IOPTOP, FOPT)

where IOPTOP() is defined in the same way as IOPT() in Section B.1 above, and FOPT() is used only if Option 2 or 12 is specified, in which case it is a vector with the same type as F(). If FOPT is not the same as F, then IOPTOP(1) must contain -1111, with the usual option specification following. The arguments L2 and K12 are pointers into FOPT() giving the location of the next output point, or the location of HINC, HDEC, HMIN, and HMAX, for Options 2 and 12 respectively. Options 15 and 17 will not have any effect on the computation until DIVA is called with KORD(1) = 0. Options 3 and 5 will not take effect until a return is made from DIVAO, after DIVAO has been called with KORD(1) = 1, 2, 3, or 5, or DIVA is called with KORD(1) = 0.

B.7.b Debugging Output

One can obtain output of various variables for debugging or other purposes with the following call.

CALL DIVADB(LPRINT, TSPECS, Y, F, KORD, TEXT)

LPRINT is a two-digit decimal integer n_1n_2 which determines print as follows

- $\mathbf{n_1}$ = **1** do not print anything from the integration variables in the call.
  - = **2** print TSPECS(), current Y(), base Y(), current F(), all pertinent contents of KORD, and the error tolerances stored in F().
  - = **3** same as $n_1 = 2$ plus difference tables up to the highest difference used.
  - = **4** same as $n_1 = 3$ plus all in the storage allocated for the differences.
- $\mathbf{n_2}$ = **1** do not print any variables in the integrator common blocks.
  - = **2** print all scalar variables in the interpolation common block, DIVASC (or SIVASC). These variables are described in Section B.3.
  - = **3** same as $n_2 = 2$ plus all scalar variables in the main integration common block. All variables used in this integration package are described as comments in the file IVACOM.
  - = **4** same as $n_2 = 3$ plus everything used from the arrays XI(), BETA(), ALPHA(), first column of G(), GS(), RBQ(), and SIGMA().
  - = **5** same as $n_2 = 4$ plus everything used in the arrays G(), D(), DS() and V().

TSPECS(), Y(), F(),KORD() [in] are all defined as in DIVA.

TEXT [in] (Type: CHARACTER) Message to be printed preceding the output. The first character serves as a vertical space control. It is usually most convenient to make TEXT a literal, *e.g.*, CALL DIVADB (..., ′0END OF DIVAF′)

B.7.c Viewing Some Internal Variables

One can get the values of some variables from the common blocks used by the integrator as follows:

INTEGER ID(5)
DOUBLE PRECISION RD(3)

CALL DIVACO(ID, RD)

where

ID [out] is an integer array where the following are returned:

ID(1) = KEMAX = index of equation with the largest error estimate.

ID(2) = KSTEP = current step number.

ID(3) = NUMDT = number of differences used for each equation

ID(4) and ID(5) are reserved for future use.

RD [out] is a double precision array where the following are returned.

RD(1) = EMAX = max. ratio of estimated error to requested error.

RD(2) and RD(3) are reserved for future use.

B.8 Modifications for Single Precision Usage

Change any entry or common block names starting with DIVA to start with SIVA. Change all DOUBLE PRECISION type statements to REAL. The value of k, used in Sections B.1.a and B.1.d in determining the dimension, IFDIM, of F() will be smaller for single precision. See Chapter 19.4, and the listings for mixed precision, and extended precisions.

B.9 Special Treatment of Weak Discontinuities

This option is provided to make the integration more efficient when there are small discontinuities in either the solution, the derivatives, or perhaps in a higher derivative. This option can get over a discontinuity faster than doing a restart, and is both faster and more reliable than ignoring the discontinuity. If the magnitude of the discontinuity in the derivative is larger than 1% of the average local value of the derivative, then one is slightly better off doing a restart.

To signal a discontinuity, in DIVAO at the point of discontinuity set KORD(1) = 0, and if there is no discontinuity in Y, set KORD(2) = 2, else set KORD(2) = 3 and store the difference (new Y at discontinuity) − (old Y at discontinuity) in Y before returning.

C. Examples and Remarks

The program DRDIVA with output ODDIVA, illustrates integrating the equations of motion for a simple two body problem. Initial conditions are selected to give circular motion. This example uses the option for integrating second order equations.

D. Functional Description

The Initial Value problem stated in Section A is solved using a variable order Adams predictor-corrector method. Details of the algorithm can be found in [1].

Although written in 1974, these subroutines still represent the current state of the art for non stiff ordinary differential equations. In comparison with other variable order Adams methods, SIVA/DIVA requires fewer derivative evaluations except at crude tolerances where it is less efficient than some, and has a wider selection of optional features.

References

1. Fred T. Krogh, *Changing stepsize in the integration of differential equations using modified divided differences*, in Dale G. Bettis, editor, **Proceedings of the Conference on the Numerical Solution of Ordinary Differential Equations**, *Lecture Notes in Mathematics 362*, 22–71, Springer Verlag, Berlin (1974). Also, JPL Internal Technical Memorandum 312, March 20, 1973.

E. Error Procedures and Restrictions

All error diagnostics and messages are handled by calls to the library subroutines MESS and SMESS or DMESS of Chapter 19.3. The following error indicator flags are possible. When the return is made to the user's program, KORD(1) will have the same value as the error indicator flag specified below. Note that if the integrator is called again, the integration will be continued if KORD(1) < 10, or if 10 < KORD(1) < 20 and the user has taken some special action. In all other cases the program is stopped. The user can change the printing and/or stopping actions taken by the error message program or change the file to which such messages are sent by calling MESS.

2 The local error tolerance specified by the user is apparently too small. If no action is taken by the user, the current tolerance TOLC will be changed to the new tolerance TOLN. If KORD(2) is set = 0, the current error tolerance in not changed and internal tests are modified to make the noise test more difficult to satisfy.

3 The stepsize has been cut back so rapidly that the integrator is going to do a restart (a discontinuity is a possibility). If KORD(2) is set = 0, the last step is merely repeated with half the stepsize instead of doing a restart.

11 The current step length is less than the minimum step length at the conclusion of the starting phase of the integration. If HMIN (see Option 12) is decreased ≤ |TSPECS(2)|, the integration continues as if this had been the value of HMIN in the first place. If KORD(2) is set = 0, the integrator stays in the starting phase. If nothing is done the subroutine executes STOP.

12 The integrator needs to take a step smaller than HMIN in order to maintain the requested local error. A reduction in HMIN is treated as in 11 above. If KORD(2) is set = 0, the integrator continues with the old stepsize even though error estimates are too large. If nothing is done the subroutine executes STOP.

13 Either the new output point precedes the last one, or it has the same index and the same value. If the integrator is called without defining a different value for the next output point, the subroutine executes STOP.

21 The step length, H, is so small that when TN + H is formed, the result is the same as TN, where TN is the current base value of the independent variable. If this problem is not due to a non-integrable singularity, it can probably be corrected by translating t so that TN is closer to 0.

22 A local error tolerance of zero has been requested.

23 The program has been stopped. The error diagnostic preceding this one tells why.

24 An impossible option has been specified.

24 An option has been specified improperly; values printed should indicate the problem.

24 Either not enough storage has been allocated, or storage required for an option overlaps with storage required by another option. In the error message output, an option number of 0 indicates storage required even if no options are present.

24 The G-stop subroutine has been called and values in KORD(1) and KORD(2) are such that no reasonable action can be taken.

F. Supporting Information

The source language is ANSI Fortran 77. In single precision this package uses common blocks SIVAEV, SIVASC, and SIVAMC internally. In double precision, DIVAEV, DIVASC, and DIVAMC are used.

Subroutine designed and written by: Fred T. Krogh, JPL, October 1987; Revised September 1990, November 1991, April 1992.

Entry	**Required Files**
DIVA	AMACH, DIVA, DMESS, MESS, OPTCHK
DIVAA	AMACH, DIVA, DMESS, MESS, OPTCHK
DIVACO	AMACH, DIVA, DMESS, MESS, OPTCHK
DIVADB	AMACH, DIVADB, DMESS, MESS
DIVAG	AMACH, DIVA, DIVAG, DMESS, DZERO, MESS, OPTCHK
DIVAIN	AMACH, DIVA, DMESS, MESS, OPTCHK
DIVAOP	AMACH, DIVA, DMESS, MESS, OPTCHK

Entry	**Required Files**
SIVA	AMACH, MESS, OPTCHK, SIVA, SMESS
SIVAA	AMACH, MESS, OPTCHK, SIVA, SMESS
SIVACO	AMACH, MESS, OPTCHK, SIVA, SMESS
SIVADB	AMACH, MESS, SIVADB, SMESS
SIVAG	AMACH, MESS, OPTCHK, SIVA, SIVAG, SMESS, SZERO
SIVAIN	AMACH, MESS, OPTCHK, SIVA, SMESS
SIVAOP	AMACH, MESS, OPTCHK, SIVA, SMESS

DRDIVA

```
      program DRDIVA
c>> 2010-06-09 DRDIVA  Krogh Used parameters for all dimenssions.
c>> 2001-05-25 DRDIVA  Krogh Minor change for making .f90 version.
c>> 1996-06-14 DRDIVA  Krogh  Small change in output format
c>> 1994-11-02 DRDIVA  Krogh  Changes to use M77CON
c>> 1994-07-18 DRDIVA  Krogh   Last change.
c--D replaces "?": DR?IVA, ?IVA, ?IVAF, ?IVAO
c Sample driver for DIVA --  Set up to solve two second order equations.
c
      integer INEQ, IFDIM, IKDIM, ITDIM, IYDIM
      parameter (INEQ=2,IFDIM=16*INEQ+1,IKDIM=6,ITDIM=4,IYDIM=4*INEQ)
      integer          NEQ, KORD(IKDIM), IOPT(6)
c--D Next line special: P=>D, X=>Q
      double precision TSPECS(ITDIM), Y(IYDIM), T, H, DELT, TFINAL
      integer NDIG
      double precision F(IFDIM), TOL
      equivalence (TSPECS(1), T), (TSPECS(2), H), (TSPECS(3), DELT),
     1  (TSPECS(4), TFINAL)
      external DIVAO, DIVAF
c++SP Default NDIG = 4
c++  Default NDIG = 10
c++ Substitute for NDIG below
      parameter (NDIG = 10)
      parameter (TOL = 10.D0 **(-NDIG))
c
      data   NEQ,     T,     H,                      DELT, TFINAL /
     1          2, 0.D0, 1.D0, 6.283185307179586477D0,    2.D1 /,
     2       Y(1), Y(2), Y(3), Y(4) /
     3       1.D0, 0.D0, 0.D0, 1.D0 /
c
c Set option for error control, local absolute error < TOL.
      data   IOPT(1), IOPT(2), IOPT(3) /
     1            16,       6,   3 /,
c Group the system to be treated as a single unit, set tolerance value
     2              KORD(6), F(3)/
     3                    2, TOL/
c Set option for second order equations
      data   IOPT(4), IOPT(5) /
     1            17,       2 /
c  Flag end of options
      data   IOPT(6) / 0/
c
c Do the integration
c
      KORD(1) = 0
  100 continue
      call DIVA(TSPECS, Y, F, KORD, NEQ, DIVAF, DIVAO,
     1  ITDIM, IYDIM, IFDIM, IKDIM, IOPT)
      if (KORD(1) .NE. 1) go to 100
      stop
      end

      subroutine DIVAF(T, Y, F, KORD)
c Sample derivative subroutine for use with DIVA
c This evaluates derivatives for a simple two body problem.
c
```

```
      integer          KORD
c--D Next line special: P=>D, X=>Q
      double precision T, Y(4), TP
      double precision F(2)
c
c Evaluate the derivatives
c
      TP = Y(1)*Y(1) + Y(3)*Y(3)
      TP = 1.D0 / (TP * SQRT(TP))
      F(1) = -Y(1) * TP
      F(2) = -Y(3) * TP
      return
      end

      subroutine DIVAO(TSPECS, Y, F, KORD)
c Sample output subroutine for use with DIVA.
c This subroutine gives output for a simple two body problem.
c
      integer          KORD
c--D Next line special: P=>D, X=>Q
      double precision TSPECS(4), Y(4)
      double precision F(2)
 1000 format (12X,
     1   'RESULTS FOR A SIMPLE 2-BODY PROBLEM'//
     2   8X, 'T', 13X, 'U/V', 11X, 'UP/VP', 9X, 'UPP/VPP')
 1001 format (1P,SP,4E15.6 / 15X, 3E15.6/' ')
c
c Do the output
c
      if (KORD .EQ. 1) then
        write (*, 1000)
      end if
      write (*, 1001) TSPECS(1), Y(1), Y(2), F(1), Y(3), Y(4), F(2)
c
      return
      end
```

ODDIVA

```
            RESULTS FOR A SIMPLE 2-BODY PROBLEM

        T             U/V           UP/VP         UPP/VPP
  +0.000000E+00  +1.000000E+00  +0.000000E+00  -1.000000E+00
                 +0.000000E+00  +1.000000E+00  -0.000000E+00

  +6.283185E+00  +1.000000E+00  +2.045697E-12  -1.000000E+00
                 -2.963546E-12  +1.000000E+00  -2.379763E-12

  +1.256637E+01  +1.000000E+00  -7.160714E-11  -1.000000E+00
                 +7.058165E-11  +1.000000E+00  -7.080797E-11

  +1.884956E+01  +1.000000E+00  -2.797420E-10  -1.000000E+00
                 +2.789906E-10  +1.000000E+00  -2.906193E-10

  +2.000000E+01  +4.080821E-01  -9.129453E-01  -4.080821E-01
                 +9.129453E-01  +4.080821E-01  -9.129453E-01
```