DDAE: an integrator for ODEs, DAEs and DDEs, part I

P.W. Sharp*   F. Krogh†

July 16, 2002

1 Introduction

DDAE is a variable order, variable stepsize Adams and BDF Fortran integrator for solving initial-value ordinary differential equations (ODEs), initial-value differential-algebraic equations (DAEs) of index 0 and 1, and delay differential equations (DDEs). The differential equations can be mixed order, and the DDEs can have both state-dependent and multiple delays and can include DAEs.

DDAE has a large number of optional inputs. Options permit the user to perform a wide range of tasks and to take advantage of features of a problem to improve efficiency. The options include those for varying the interpolation, saving the solution, controlling the stepsize and order selection, and solving for g-stops. DDAE also has reverse communication (returning to the driver calling DDAE for function evaluations) as an option. This makes it easy for the user to call DDAE from other software and to use special software for solving linear equations.

A distinctive feature of DDAE is the ability to group the equations and use different options for different groups. This can lead to a marked reduction in the CPU time. For example, the equations could be divided into non-stiff and stiff equations, and Adams and BDF methods used for the two groups respectively.

This report summarises the features of DDAE with an emphasis on the options.

Table of contents

*Department of Mathematics, University of Auckland, Private Bag 92019, Auckland, NEW ZEALAND. sharp@math.auckland.ac.nz

†Math à la Carte, Inc., P.O. Box 616, Tujunga, CA 91043, fkrogh@mathalacarte.com

2 Problems

2.1 Explicit ODEs

DDAE solves initial-value explicit ODEs of the form

$$z_i^{(d_i)} = f_i(t, \mathbf{y}), \quad \mathbf{y}(t_0) = \mathbf{y}_0, \quad i = 1, 2, \ldots, n, \tag{1}$$

where $z_i^{(k)}$ is the k-th derivative of z_i with respect to t, $d_i > 0$ is the order of the i-th differential equation, and $\mathbf{y}$ is the vector

$$[z_1, z_1', \ldots, z_1^{(d_1-1)}, z_2, \ldots, z_n^{(d_n-1)}]^T.$$

First-order problems have $d_i = 1$ and second-order problems $d_i = 2$. The size of $\mathbf{y}$ is $d_1 + \ldots + d_n$ which is n for first-order problems and $2n$ for second-order problems.

2.2 Implicit ODEs and DAEs

DDAE solves initial-value implicit ODEs and DAEs of the form

$$\mathbf{f}(t, \mathbf{y}, \bar{\mathbf{z}}) = \mathbf{0}, \quad \mathbf{y}(t_0) = \mathbf{y}_0, \tag{2}$$

where $\mathbf{f}$ is an n-vector, $\mathbf{y}$ is defined as for (1) and $\bar{\mathbf{z}} = [z_1^{(d_1)}, z_2^{(d_2)}, \ldots, z_n^{(d_n)}]^T$. Algebraic variables are regarded as satisfying zero-th order differential equations i.e. $d_i = 0$. The system (2) is a DAE if any of the d_i are zero and an implicit ODE otherwise.

DDAE assumes there exists a $\bar{\mathbf{z}}_0 = \bar{\mathbf{z}}(t_0)$ such that the initial conditions are consistent i.e. $\mathbf{f}(t_0, \mathbf{y}_0, \bar{\mathbf{z}}_0) = \mathbf{0}$. The vector $\bar{\mathbf{z}}_0$ can be supplied by the user or DDAE can calculate a suitable vector.

2.3 DDEs

The DDEs solved by DDAE have the form

$$z_i^{(d_i)} = f_i(t, \mathbf{y}, \mathbf{y}(\alpha_1(t, \mathbf{y})), \ldots, \mathbf{y}(\alpha_k(t, \mathbf{y}))), \quad t \geq t_0, \quad i = 1, 2, \ldots, n, \tag{3}$$

where $\alpha_\ell \leq t$, $\ell = 1, \ldots, k$. The user can supply the initial conditions for $\mathbf{y}$ as a function or supply $\mathbf{y}(t_0)$ and require DDAE find a function that makes the first m derivatives of the solution continuous at $t = t_0$.

2.4 Implicit ODEs and DAEs with delays

Implicit ODEs and DAEs with delays solved by DDAE have the form

$$\mathbf{f}(t, \mathbf{y}, \bar{\mathbf{z}}, \mathbf{y}(\alpha_1(t, \mathbf{y}), \ldots, \mathbf{y}(\alpha_k(t, \mathbf{y}))) = \mathbf{0}, \tag{4}$$

where $\mathbf{f}$ is an n-vector, $\alpha_\ell \leq t$, $\ell = 1, \ldots, k$, and the vectors $\mathbf{y}$ and $\bar{\mathbf{z}}$ are defined as in (2). The user can supply the initial conditions for $\mathbf{y}$ as a function or supply $\mathbf{y}(t_0)$ and require DDAE find a function that makes the first m derivatives of the solution continuous at $t = t_0$. DDAE assumes there exists a $\bar{\mathbf{z}}_0 = \bar{\mathbf{z}}(t_0)$ such that the initial conditions are consistent. The vector $\bar{\mathbf{z}}_0$ can be supplied by the user or DDAE can calculate a suitable vector.

2.5 Remarks

Ordinarily, DDAE calculates the solution to (1), (2), (3) or (4) at a set of output points specified by the user. DDAE has options that permit the user to print the solution at the end of each integration step or when the solution has satisfied a prescribed condition. It is also possible to have DDAE save a piecewise continuous approximation to the solution.

The user can require the solution for a ODE, DAE or DDE satisfy constraints. After calculating the solution at the end of a step, DDAE projects onto the constraints.

3 Integrations using default values

If the problem being integrated is a system of first-order ordinary differential equations, DDAE can be used with default values for the options. DDAE assumes the equations are non-stiff and does the integration using Adams methods.

The user supplies a driver to call DDAE, a derivative subroutine `ddaef` to evaluate the first derivative and an output subroutine `ddaeo` to print the solution. The user specifies the first value of t at which the solution is required. DDAE integrates to this point and calls `ddaeo`. After the solution is printed, the integration is continued by specifying a new output point. DDAE returns to the driver upon reaching the final output point or if an error occurs.

3.1 Argument list and subroutines

A call to DDAE is of the form

call ddae (`t`, `y`, `f`, `iw`, `w`, `neqn`, `tol`, `opt`, `ddaef`, `ddaeo`)

`t` A real scalar that holds the current value of t. On the first call to DDAE, `t` must be t_0. On exit from DDAE, if no error occurred, `t` is t at the final output point; if an error occurred, the value in `t` will depend on the error.

`y()` A one-dimensional real array that holds information about $\mathbf{z}$ and its derivatives. On the first call to DDAE, `y(`i`)` must be $\mathbf{y}_{0,i}$, $i = 1, \ldots, n_y$, where n_y is the size of the vector $\mathbf{y}$. On exit from DDAE, if no error occurred, `y(`i`)` is the numerical approximation to y_i, $i = 1, \ldots, n_y$, at the final output point; if an error occurred, the contents of `y` will depend on the error.

`f()` A one-dimensional real array of size at least the size of the vector $\mathbf{f}$.

`iw()` A one-dimensional integer array. Elements `iw(1)`, `iw(2)` and `iw(3)` pass information about the state of the integration to and from DDAE and `iw(4)` and `iw(5)` hold the size of arrays. The user must set `iw(1)`, `iw(4)` and `iw(5)` before the first call to DDAE.

`iw(1)` - this must be 0 on the first call to DDAE or when restarting an integration. On exit from DDAE, `iw(1)` will either be 5 or 6 when using the default values for the options. If `iw(1)` = 5, the final output point was reached; if `iw(1)`= 6, an error occurred. The value of `iw(2)` and in some cases `iw(3)` describe the error.

`iw(2)` - used on exit from DDAE as a flag for an error return.

`iw(3)` - not used when using the default values for the options.

`iw(4)` - the size of the array `iw`.

`iw(5)` - the size of the array `w`.

`w()` A one-dimensional real array for work space and to hold real scalars that may be of interest to the user. The minimum allowable size of `w` depends on n, the precision of the arithmetic and the options.

On the first call to DDAE, `w(1)` must be t at the first output point when the default values are used for the options.

`neqn` An integer scalar equal to n.

`tol` A real scalar equal to the error tolerance. DDAE attempts to keep the global error proportional to `tol`. The proportionality depends on the kind of error control used, the problem and the range of integration. For first-order ODEs with default error control, DDAE adjusts the stepsize so the root mean square norm of the estimated local error for $\mathbf{y}$ is no larger than $c \max\{1, \sqrt{s}\}$ tol, where $c < 1$. The quantity s is $\max\{\omega, ||\mathbf{y}_0||_2^2\}$ at the start of the integration, where ω is a small floating point number included to make s non-zero. At the end of the j-th step, s is updated to $(s + ||\mathbf{y}_j||_2^2)/2$, where $\mathbf{y}_j$ is the numerical approximation at the end of the step.

`opt()` A one-dimensional real array to hold the options selected by the user. When the default values are used for the options, `opt` can have size one and `opt(1)` must be 0 (entered as $0.d0$ since `opt` is a real array) on the first call to DDAE. Setting `iopt(1) = 0` indicates the user has not selected any options.

`ddaef` A subroutine that evaluates $\mathbf{f}$. When the default values are used for the options, `ddaef` has the form

```
subroutine ddaef (y, f, iw, w)
double precision y(*), f(*), w(*)
integer iw(*)
Compute f(i) = f_i(t,y), i = 1,2,...,n.
return
end
```

The element `w(1)` is t for the evaluation of $\mathbf{f}$ and `y(`i`)`, $i = 1, \ldots, n_y$, is y_i for the evaluation, where n_y is the size of $\mathbf{y}$. The arguments `iw` and `w` are the work arrays that appear in the argument list for DDAE. If these arrays are larger than required by DDAE, the extra space can be used to pass values between `ddaef` and the driver. Some options require the use of `iw` and `w` in `ddaef`.

The argument list for `ddaef` is the same when options are used, but the form of the subroutine can be different.

`ddaeo` The name of the subroutine for processing output points. `ddaeo` is called whenever an output point is reached, unless the option for reverse communication (returning to the driver calling DDAE for function evaluations) is selected. The first call to `ddaeo` occurs at $t =$ `w(1)`. After printing the solution at this point, the user can specify a new output point by changing `w(1)` to the required value. If `w(1)` is changed, DDAE will continue with the integration; if `w(1)` is unchanged, DDAE will return to the calling program.

When the default values are used for the options, `ddaeo` has the form

```
subroutine ddaeo (y, f, iw, w)
double precision y(*), f(*), w(*)
integer iw(*)
Print the solution (stored in y) for t = w(1) .
If required, specify a new output point by changing w(1).
return
end
```

The element `w(1)` is `t` is t at the output point and `y(`i`)`, $i = 1, \ldots, n_y$, is the numerical approximation to y_i at t, where n_y is the size of $\mathbf{y}$. The arguments `iw` and `w` are the work arrays that appear in the argument list for DDAE. If these arrays are larger than required by DDAE, the extra space can be used to pass values between `ddaeo` and the driver. Some options require the use of `iw` and `w` in `ddaeo`.

The argument list for `ddaeo` is the same when options are used, but the form of the subroutine can be different.

3.2 Example 1

Euler's equation of motion for a rotating rigid body subject to no external forces form a system of three first-order differential equations. Enright and Pyrce [1] give the example

$$\left.\begin{array}{rcr} z_1' &=& z_2 z_3 \\ z_2' &=& -z_1 z_3 \\ z_3' &=& -0.51 z_1 z_2 \end{array}\right\} \quad t > 0, \quad z(0) = [0, 1, 1]^T. \tag{5}$$

We use DDAE to find the solution to (5) at $t = 5, 10, 15, 20$ and 25, for an error tolerance of 10^{-10}. Figure 1 gives a sample driver and Figure 2 gives a sample derivative subroutine and output subroutine.

4 Integrations using options

4.1 Description of the options

Table 1 gives a summary of the options. The first column is the numerical code for the option, the second column is the identifier, the third column is a brief description of the option and the fourth column is the number of arguments for the options.

The user specifies options by entering their numerical code into the array `opt`. The codes need not be in ascending order. Some options have arguments and these are entered into `opt` immediately following the numerical code for the option. The arguments for an option must be entered and when there is more than one, they must be entered in the correct order.

As noted in the introduction, the user can group equations. Some options, called group options, apply to individual groups of equations defined by the user. Other options, called global options, apply to all equations. Options 1 to 17 are groups options, Options 41 to 63 are global options and Options 18 to 40 are reserved for future use.

The user must specify all group options they wish to use before specifying global options. If the user does not select a particular option for the first group, DDAE uses the default value for the option. This includes the case when there are no groups. If the user does not select a particular option for the second or subsequent group, the value of the option for the previous group is used.

Option		Brief description	No. arguments
0	`daeend`	No more options.	0
1	`daeidx`	Index of the last equation in the group.	1
2	`daeord`	Order of the equations.	1
3	`daeimp`	Implicit and explicit derivatives.	1
4	`daestf`	Stiffness and non-stiffness.	2
5	`daeitl`	Consistency of the initial conditions.	1
6	`daespa`	Sparsity of the functions.	To be decided
7	`daeabs`	Absolute error control.	1
8	`daerel`	Relative error control.	2
9	`daedia`	Control of diagnostic output.	3
10	`daemul`	Multirate.	To be decided
11	`daeptl`	Partial correction.	0
12	`daebnd`	Band width of the iteration matrix.	2
13	`daeign`	Ignore the equations in this group.	0
14	`daelwr`	Lower derivatives are available.	1
15	`daernd`	Reduce the round-off error.	1
16	`daemxk`	Maximum integration order.	2
17	`daeerf`	Error in $\mathbf{f}$.	2
18-40		Reserved for future use.	
41	`daercf`	Reverse communication for the functions.	0
42	`daercl`	Reverse communication for the linear algebra.	1
43	`daerco`	Reverse communication for output.	0
44	`daenoc`	Number of constraints.	1
45	`daehbg`	Initial stepsize.	1
46	`daehmx`	Maximum allowable stepsize.	1
47	`daehfx`	Factor for stepsize changes.	1
48	`daegbl`	Save the global solution.	user specified
49	`daeuhk`	User-supplied stepsize and order.	2
50	`daeshk`	Save the stepsize and order.	2
51	`daeplt`	Print the solution for plotting.	user specified
52	`daenof`	Number of derivatives per accepted step.	1
53	`daevbl`	Arc length as the independent variable.	2
54	`daeitp`	Default rules for interpolation.	user specified
55	`daecnt`	Continuous interpolants.	0
56	`daestp`	Maximum number of steps.	1
57	`daept0`	Output at the initial point.	0
58	`daedel`	Evenly spaced output points.	1
59	`daepnt`	Getting the solution at the final point.	1
60	`daenxt`	The next output point.	1
61	`daeste`	Output at the end of steps.	1
62	`daegex`	Number of extrapolatory g-stops.	0
63	`daegin`	Number of interpolatory g-stops.	0

Table 1: A summary of the options for DDAE.

The description below of each option begins with the numerical code. This is followed by a list of arguments if any. This list is enclosed in parenthesis and if an argument is followed by an equals sign, the value after the equals sign is the default value for the argument. Throughout the descriptions, u denotes the unit round-off.

The numerical code for all options and the arguments of some options are integers. Since `opt` is a real array, the integers must be stored as floating point numbers. If an option or an argument is supposed to have an integer value, DDAE uses the nearest integer to the floating point value supplied by the user.

0. This indicates the end of the options. This was used in Example 1 and must be used with every problem.

1. $(m_1 = n)$. The argument m_1 is the index of the last equation in the group. This argument is necessary only if the user wants more than one group. If the end of the group options is reached without getting this option with $m_1 = n$, then it is as if this option was specified with $m_1 = n$ just before the first global option. If Option 1 is specified more than once, the values of m_1 must be strictly increasing.

2. $(m_2 = 1)$. The order of the differential equations in the group. The equations in the group must be the same order.

3. $(m_3 = 1)$. If $m_3 = 1$, the highest derivative for all the equations in the group are explicitly defined; if $m_3 = -1$, the highest derivative of at least one and possibly all of the equations in the group is implicitly defined.

4. $(m_4 = 0,\ n_4 = 1)$. This option controls which equations are treated as non-stiff and which as stiff. The non-stiff equations are integrated using Adams methods and the stiff equations using BDF methods. It is possible to use Adams and BDF methods on the same equation when the equation is order two or higher. If Option 4 is not selected, Adams methods are used for all equations.

When Option 4 is selected, the difference table stored by DDAE contains the $(d - m_4)$-th derivative of $\mathbf{z}$ for the group, where d is the order of the differential equations in the group and $0 \le m_4 \le d$. Hence if $m_4 = 0$, the d-th derivative is stored and DDAE will use Adams methods to integrate all dependent variables in the group. If $m_4 > 0$, DDAE will use BDF methods to integrate $\mathbf{z}^{(d-1)}, \ldots, \mathbf{z}^{(d-m_4)}$ and Adams methods to integrate $\mathbf{z}^{(d-m_4-1)}, \ldots, \mathbf{z}$. Table 2 summarise what methods are used for equations of order one, two and three. If $d = 0$, there is no integration formula for the group.

m_4	First-order	Second-order		Third-order		
	$\mathbf{z}$	$\mathbf{z}$	$\mathbf{z}'$	$\mathbf{z}$	$\mathbf{z}'$	$\mathbf{z}''$
0	Adams	Adams	Adams	Adams	Adams	Adams
1	BDF	Adams	BDF	Adams	Adams	BDF
2	-	BDF	BDF	Adams	BDF	BDF
3	-	-	-	BDF	BDF	BDF

Table 2: The integration methods used for different values of m_4.

If the equations are non-stiff, m_4 should be 0. If the equations are first order and stiff, m_4 should be 1. For higher order equations that may be stiff, the best value for m_4 will depend on the relative sizes of the partial derivatives of $\mathbf{f}$ with respect to $\mathbf{z}$ and its derivatives.

The second argument n_4 of Option 4 controls the switching between Adams and BDF methods during an integration. If $n_4 = 1$, there will be no switching. If $n_4 = -1$, DDAE will begin the integration with the methods specified by m_4 and switch backward and forward between Adams and BDF methods to make the integration more efficient.

5. ($m_5 = 1$). This option gives information about the consistency of the initial conditions when solving a DAE. If $m_5 = 1$, the initial conditions are consistent. If $m_5 = -1$, the initial conditions are not consistent and DDAE will use an iterative scheme to find $\bar{\mathbf{z}}_0$ that makes the initial conditions consistent. The starting value for the iterative scheme will be the $\bar{\mathbf{z}}_0$ supplied by the user.

6. Information about the sparsity structure of the function $\mathbf{f}$ when solving stiff ODEs, implicit ODEs and DAEs. To be decided.

The next two options specify the type of error control. Let θ_k, $k = 1, \dots, N_g$, be a scaling factor for the tolerance tol where N_g is the number of groups, and let $e_{k,j}$ be the local error estimate for the j-th equation in the k-th group. The local error estimate is for the $(d_k - 1)$-st derivative, $k = 1, \dots, N_g$, where d_k is the order of the equations in the k-th group. Define

$$E = \max_{1 \le k \le N_g} \left[\frac{1}{n_k \, \mathrm{tol}^2 \, \theta_k^2} \sum_{j=1}^{n_k} e_{k,j}^2 \right]^{1/2} \tag{6}$$

where n_k is the number of equations in the k-th group.

E is the maximum of the root mean square norms, scaled by tol and θ_k, for the groups. On each step, DDAE attempts to select the stepsize so that $-\log_e E = \alpha > 1$, where α is a constant set in DDAE.

The scaling factor θ_k is $\max\{a_7, a_8\sqrt{s}\}$, where a_7 and a_8 are constants and s is a function of the solution. The default value of a_7 and a_8 is one; the value can be overridden using Options 7 and 8 respectively. The quantity s is defined in Option 8.

7. ($a_7 = 1$). This gives absolute error control with $\theta_k^2 = a_7^2$.

8. ($a_8 = 1, s = 0$). This gives relative error control with

$$\theta_k^2 = a_8^2 s.$$

The quantity s is initialised by the user at the start of an integration and updated by DDAE to

$$\frac{1}{2}(s + \sum (z_i^{(d_k - 1)})^2)$$

at the end of each step, where z_i is the numerical approximation to the i-th component of z at the end of the step and the sum is over the components of $\mathbf{z}$ in the group. If the user initialises s to zero, DDAE sets it to

$$\max\{\omega, \sum (z_i^{(d_k - 1)})^2 + h^2 \sum (z_i^{(d_k)})^2\}$$

for the first step, where h is the stepsize and ω, included to make s non-zero, is a small floating point number fixed in DDAE.

9. ($m_9 = 0$, $n_9 = 0$, p_9). This option controls the diagnostic output for the group. The argument m_9 specifies what is printed each time, n_9 specifies the number of times the output is produced and p_9 is the unit number for output, with $p_9 = 0$ meaning the output is to the standard output unit. If $n_9 > 0$, the output is printed for n_9 attempted steps. Currently, the acceptable values of m_9 and their meaning are

0 Nothing is printed. The same effect is achieved by setting $n_9 = 0$.

1 The value of the independent variable, the stepsize h and the norm of the local error estimate for each group is printed after the local error estimate is formed.

2 The output for $m_9 = 1$ and information about the order and stepsize selection, is printed immediately after the local error estimate is formed.

3 The output for $m_9 = 2$, the predicted solution and if the step was accepted, the corrected solution, is printed at the end of the step.

If $m_9 = 0$, a call to the subroutine `ddaeop` can be used to turn on the output.

10. The multirate option. This option allows the integration of different equations using different stepsizes. To be decided.

11. This option applies to steps on which DDAE does two derivative evaluations (the usual case). Selecting Option 11 stipulates the first evaluation for the current group is done only after the second evaluation for the previous groups, a technique known as partial correction. Option 11 can be used whenever the derivative of previous groups does not depend on or depends weakly on the solution for the current group.

An important application of Option 11 is the integration of state and variational equations. If the state equations are the first group and the variational equations the second group, the matrix of partial derivatives in the variational equations can be evaluated just once each step.

The default is not to use partial correction.

12. (m_{12}, n_{12}). The iteration matrix for stiff ODEs, implicit ODEs and DAEs is banded. The arguments m_{12} and n_{12} are respectively the number of non-zero diagonals above and below the main diagonal of the iteration matrix. Thus for example, a penta-diagonal matrix has $m_{12} = 2$ and $n_{12} = 2$.

13. This option tells DDAE to ignore the equations in the current group. DDAE will not update the difference table or the solution for the equations, and will not use the equations in the stepsize and order selection. The user must ensure the derivative for the other groups are correctly evaluated and provide space for the equations in the current group.

Option 13 is useful when the system of equations contains equations that are significant for only part of the integration. For example, the system may model a combination of several chemical reactions and one of the reactions does not become significant until halfway through the integration. The equations for this reaction could be ignored for the first half of the integration and turned on for the second half.

14. ($m_{14} = 0$). Whenever the d_i-th derivative ($d_i > 0$) of z_i is evaluated, either by calling `ddaef` or by reverse communication, the derivatives of order $d_i - m_{14}$ to order $d_i - 1$ are also evaluated. These lower order derivatives must be stored in the argument `y`. When lower derivatives are easy to compute, using them can make the integration more efficient. Option 52 with $m_{52} = 1$ can often be used to advantage with Option 14.

15. ($m_{15} = 0$). This option specifies if a special technique will be used to reduce the round-off error. The acceptable values of m_{15} and their meaning are

0	The technique will not be used.
1	The technique will be used on the independent variable only.
2	The technique will be used on the independent variable and $\mathbf{y}$.

The gain in precision for the dependent variables is not large, typically no more than half a decimal digit, rarely more than one decimal digit, and occurs only for tolerances near limiting precision. We recommend Option 15 is used for the dependent variables if $\text{tol} \le 100u$.

The gains in precision for the independent variable are similar, but they are not restricted to tolerances near limiting precision. For example, if the independent variable is large in magnitude, the gains can occur for $\text{tol} \gg 100u$.

If m_{15} is non-zero for any group, the technique is used on the independent variable.

16. (m_{16}, n_{16}). The arguments m_{16} and n_{16} are respectively the maximum integration order for Adams and BDF methods applied to equations in the group. Setting the arguments to a positive value smaller than the order DDAE would normally use will reduce the storage requirements for DDAE, enabling the solution of a problem that otherwise would be too large. However, reducing the order is likely to increase the CPU time.

If m_{16} or n_{16} is negative, DDAE will set the maximum order according to the amount of storage provided by the user in the work arrays.

Restricting the maximum order for BDF methods may also be useful when the differential equations have eigenvalues near the imaginary axis. BDF methods frequently use too high an order in this case and although DDAE attempts to handle such integrations efficiently, it may be useful to limit the order.

The maximum order set in DDAE assumes double precision arithmetic. If the arithmetic is higher precision and an accurate integration is required, the CPU time could be reduced if Option 16 is used to increase the maximum order for Adams methods.

The default is to use the maximum order set in DDAE.

17. ($a_{17} = u$, $b_{17} = u$). The arguments a_{17} and b_{17} are the absolute and relative error in an evaluation of $\mathbf{f}$, assuming the vector arguments of $\mathbf{f}$ are known exactly.

Options 18 to 40 are reserved for future use.

This completes the options for the groups. The remaining options apply to all equations.

41. The functions defining the problem are evaluated by reverse communication and not by calling `ddaef`. The functions consist of $\mathbf{f}$ and lower order derivatives, the Jacobian of $\mathbf{f}$, the residual of g-stops and constraints, and the partial derivatives of the constraints. When an evaluation is required, DDAE returns to the calling program with `iw(1)` $= 1$. The user should do the evaluation as specified by `iw(2)` and `iw(3)`, store the evaluation in the argument `f` and call DDAE.

It is not permissible to evaluate some functions using reverse communication and some using forward communication (calling the function routines from within DDAE). The functions must all be evaluated using the same form of communication. Thus for example, when solving a system of stiff ODEs, it is not permissible to evaluate the derivative using forward communication and the Jacobian using reverse communication.

The default is to use forward communication.

42. (m_{42}). Reverse communication is used when solving the linear system that defines the solution to a stiff ODE, an implicit ODE, or a DAE. DDAE has subroutines that use a direct method to solve dense and banded systems. Option 42 provides an opportunity for the user to reduce the CPU time by using other subroutines.

DDAE returns to the calling program with `iw(1) = 2`. Setting $m_{42} = 1$ means the user will employ a direct method to solve the linear system; $m_{42} = -1$ means the user will employ an iterative method. Making the distinction between direct and iterative methods permits a reduction in CPU time when direct methods are used.

The default is to use forward communication.

43. Reverse communication is used whenever a call to the output subroutine `ddaeo` is required. DDAE returns to the calling program with `iw(1)` $= 3$. The user should take the required action and then call DDAE if further integration is required.

The default is to use forward communication.

44. ($m_{44} = 0$). The argument m_{44} is the number of constraints the user wishes to impose on the solution. Constraints can represent conservation laws that must be satisfied. They can also be used to prevent the numerical solution drifting too far from the true solution when a DAE of index two or higher is transformed to a index zero or index one DAE. The residuals and partial derivatives for the constraints are calculated in `ddaef`.

45. ($a_{45} = 0$). The argument a_{45} is the absolute value of the initial stepsize. a_{45} must be non-negative; if $a_{45} = 0$, DDAE selects a suitable value. The accuracy and efficiency of an integration does not depend critically on a_{45}. The user can employ Option 45 to provide information about the scale of the problem.

46. ($a_{46} = 0$). The argument a_{46} is the absolute value of the maximum allowable stepsize. a_{46} must be non-negative; if $a_{46} = 0$, there is no maximum stepsize.

47. ($a_{47} = 2$). The argument a_{47} is the nominal factor by which the stepsize is changed. a_{47} must be positive. After getting started, DDAE will typically not increase the stepsize unless it can be increased by this factor, and will typically decrease the stepsize by the factor $\max(.5, 1/a_{47})$. Smaller values of a_{47} increase the integration overhead, but are likely to lead to fewer evaluations, giving an overall reduction in the CPU time if $\mathbf{f}$ is expensive to evaluate. Smaller values of a_{47} also give an accuracy in the results that is a smoother function of the tolerance. The increase in overhead is less for BDF methods.

48. ($m_{48} = 0$, n_{48}, $\ldots$) This option is used to save the global solution for components of $\mathbf{z}$. The acceptable values for m_{48} and their meaning are

0	The solution is not saved.
−1	DDAE calls `ddaeo` whenever it is time to save the solution.
1	Prints the solution on the standard output unit whenever it is time to save the solution.
2	DDAE appends the solution to the file `ddae_gbl` on unit number n_{48} as formatted output whenever it is time to save the solution.
3	DDAE appends the solution to the file `ddae_gbl` on unit number n_{48} as unformatted output whenever it is time to save the solution.

If $m_{48} > 0$, the global solution is written to full precision. If $m_{48} < 2$, n_{48} is omitted.

The user indicates the components of **z** they would liked saved by specifying the integers p_1, q_1, p_2, q_2,. . .,p_k, q_k, 0, where $q_1, \ldots, q_k$ are negative and $1 \leq p_1 \leq |q_1| \leq \ldots p_k \leq |q_k| \leq n$ (the integers are stored as floating point numbers in `opt` immediately following the argument or arguments already specified for Option 48). DDAE saves the components p_1 to $|q_1|$, p_2 to $|q_2|$, . . ., p_k to $|q_k|$. For example, if the user specifies the integers $2, -4, 7, -9, 0$, DDAE saves the second, third, fourth, seventh, eighth and ninth components. If the user wishes to save all components, they need specify just the final 0.

The filename `ddae_gbl` is fixed in DDAE. `ddae_gbl` is opened at the start of an integration as a new file. It is the user's responsibility to ensure the file is not accidentally overwritten by a later integration. `ddae_gbl` is closed by DDAE when the final output point is reached and the user does not continue the integration. If the user stops the integration for any other reason, including after an error return from DDAE, it is the user's responsibility to close the file or let the operating system do so.

49. ($m_{49} = 0$, n_{49}). This option turns off all stepsize and order control and leaves it to the user to supply the stepsize and orders at the start of each step. The acceptable values of m_{49} and their meaning are

0	This turns off the option i.e. DDAE uses its own stepsize and order control.
−1	`ddaeo` is called at the start of each step to get the stepsize and orders.
1	DDAE reads the stepsize and orders from the standard input unit.
2	DDAE reads the stepsize and orders from the file `ddae_hk` as formatted input on unit number n_{49}.
3	DDAE reads the stepsize and orders from the file `ddae_hk` as unformatted input on unit number n_{49}.

If $m_{49} < 2$, n_{49} is omitted.

DDAE does not check if the stepsize and orders are consistent with those for the previous step.

The values for the stepsize and the orders have presumably been saved on a previous integration of a problem whose solution is close to that for the current problem. Option 49 is useful when estimating partial derivatives using finite differencing.

50. ($m_{50} = 0$, n_{50}). DDAE saves the stepsize and orders at the end of each step. The acceptable values of m_{50} and their meaning are

0 This turns off the option i.e. the stepsize and orders are not saved.
−1 `ddaeo` is called at the end of each step.
1 DDAE writes the stepsize and orders to the standard input unit.
2 DDAE writes the stepsize and orders to the file `ddae_hk` as formatted output on unit number n_{50}.
3 DDAE writes the stepsize and orders to the file `ddae_hk` as unformatted output on unit number n_{50}.

If $m_{50} > 0$, the stepsize is written to full precision. If $m_{50} < 2$, n_{50} is omitted.

The format of the records in `ddae_hk` is the same as for Option 49, enabling `ddae_hk` to be used with Option 49. The stepsize and orders in `ddae_hk` can also be used to investigate the behaviour of the solution.

The filename `ddae_hk` is fixed in DDAE. `ddae_hk` is opened at the start of an integration as a new file. It is the user's responsibility to ensure `ddae_hk` is not accidentally overwritten by a later integration. The file is closed when the final output point is reached and the user does not continue the integration. If the user stops the integration for any other reason, including after an error return from DDAE, it is the user's responsibility to close the file or let the operating system do so.

51. ($m_{51} = 0$, n_{51}, p_{51}, $\ldots$). At the end of each step DDAE writes components of the solution in a form suitable for plotting. The components are written as floating point numbers with p_{51} significant digits. The acceptable values of m_{51} and their meaning are

0 This turns off the option i.e. the components are not written.
−1 `ddaeo` is called at the end of each step.
1 DDAE writes the components to the standard input unit.
2 DDAE writes the components to the file `ddae_plt` as formatted output on unit number n_{51}.
3 DDAE writes the stepsize and orders to the file `ddae_plt` as unformatted output on unit number n_{51}.

The argument n_{51} is omitted if $m_{51} < 2$ and p_{51} is omitted if m_{51} is 0, −1 or 3.

To specify which components are written, the user divides the equations into k groups and provides the integers $l_1, u_1, i_1, \ldots, l_k, u_k, i_k, 0$. These integers are defined in the same way as those for Option 54 and stored as floating point numbers in `opt` immediately following the argument or arguments already specified for Option 51.

The filename `ddae_plt` is fixed in DDAE. `ddae_plt` is opened at the start of an integration as a new file. It is the user's responsibility to ensure the file is not accidentally overwritten by a later integration. `ddae_plt` is closed when the final output point is reached and the user does not continue the integration. If the user stops the integration for any other reason, including after an error return from DDAE, it is the user's responsibility to close the file or let the operating system do so.

52. ($m_{52} = 0$). The argument m_{52} specifies the number of derivative evaluations computed each accepted step. This applies to Adams and BDF formulas.

0 DDAE will use either one or two evaluations according to which it thinks will give the more efficient integration.
1 DDAE will use one derivative evaluation.
2 DDAE will use two derivative evaluations.

53. (m_{53}, n_{53}). This option is used to change the independent variable from t to τ, where τ is a measure of the arc length. The change is done by appending a first-order differential equation for t to the system of equations being solved by the user. Unless specified otherwise in `ddaef`, the appended differential equation is

$$\frac{dt}{d\tau} = \left[\sum_{i=m_{53}}^{n_{53}} \left(\frac{e_i}{f_i} \right)^2 \right]^{1/2}$$

where e_i is local error estimate, scaled by the tolerance, for the i-th equation and f_i is the derivative for the i-th equation.

If $m_{53} = 0$, the user must store $dt/d\tau$ in the $(n + 1)$-st location of `f` on the last derivative evaluation of each step.

The values specified for output remain in terms of t, even though the new independent variable is τ.

54. (...) This option controls what interpolated values will be available in the arguments `y` and `f`. Option 54 permits a reduction in CPU time through a reduction in the number of interpolated values computed.

The user divides the equations into k groups and stores the integers $l_1, u_1, i_1, \ldots, l_k, u_k, i_k, 0$ as floating point numbers in `opt` immediately following the numerical code (the '54') declaring this option. The integers l_m and u_m specify the derivatives interpolated for the equations in the m-th group and i_m is the index of the last equation in the m-th group. The i_m must be strictly increasing. Unlike Options 1 to 17, the equations in a group can differ in order.

If $l_m < 0$, no interpolation is done for the m-th group. If $0 \leq l_m \leq d$ and $l_m \leq u_m \leq d$, where d is the order of the differential equation, the derivatives of orders $l_m, \ldots, u_m$ for the equations in the m-th group are interpolated. If $u_m < 0$ and $0 \leq l_m \leq d + u_m$, the derivatives of orders $l_m, \ldots, d+u_m$ are interpolated. A negative value of u_m is useful when the equations in a group have different orders.

The interpolated value for the d-th derivative is stored in `f`; the values for lower order derivatives are stored in `y`. Elements in `y` and `f` for which interpolations have not been done will not contain the correct values at the interpolated point.

55. The interpolants will be continuous across the interval of integration. When Option 55 is not selected, interpolants should be as accurate as the integrated values, but they are not continuous.

56. (m_{56}). The argument m_{56} is the maximum number of steps between consecutive output points. The default is to have no limit on the number of steps.

57. DDAE will regard the initial point as an output point. The default is not to do so.

58. (a_{58}). Output points are at $t = t_0 + k\ a_{58}$, $k = 1, 2, \ldots$. The sign of a_{58} must be the same as the direction of integration. The value of a_{58} can be changed when an output point of this type is reached.

59. ($m_{59} = 1$). The argument m_{59} specifies how the solution at the final output point is obtained. If $m_{59} = 1$, DDAE integrates past the point and interpolates; if $m_{59} = -1$, DDAE uses extrapolation. Interpolation ($m_{59} = 1$) should be used in most cases when the equations can be integrated beyond the final output point because this is usually more accurate, more reliable and requires less CPU time than using extrapolation.

60. (a_{60}). The argument a_{60} gives the next output point. When the point is reached, a new point can be set. If none is set, the next output point will be the final output point (stored in `w(1)`). Option 60 is particularly useful when Option 59 is used.

61. (m_{61}). DDAE will call `ddaeo` (or return to the program calling DDAE if reverse communication is being used) at the end of every m_{61} steps.

62. ($m_{62} = 0$). The argument m_{62} is the number of extrapolatory g-stops.

63. ($m_{63} = 0$). The argument m_{63} is the number of interpolatory g-stops.

References

[1] W.H. Enright, J.D. Pryce, *Two FORTRAN packages for assessing initial value methods*, ACM Trans. Math. Soft., **13**, 1 (1987), 1-27.

```
  double precision tfinal
  integer          neqn
  parameter        (neqn = 3, tfinal = 25.d0)
  double precision t,tol
  double precision f(neqn),opt(1),w(20*neqn+100),y(neqn)
  integer          iw(100)
  external         ddaef,ddaeo

  t      = 0.d0   ! The initial conditions
  y(1)   = 0.d0
  y(2)   = 1.d0
  y(3)   = 1.d0
  tol    = 1.d-10        ! The tolerance
  opt(1) = 0.d0          ! Use default values for all options
  iw(1)  = 0             ! Signal the start of the integration
  iw(4)  = 100           ! The size of iw
  iw(5)  = 20*neqn+100   ! The size of w
  w(1)   = 5.d0          ! The first output point

5 continue   ! Attempt to integrate to each output point

     call ddae (t,y,f,iw,w,neqn,tol,opt,ddaef,ddaeo)

     ! Stop when the final output point is reached.

     if (t .ge. tfinal) then
        stop
     end if

  go to 5

  end
```

Figure 1: A sample driver for Example 1.

This figure was cited in Example 1 on page 5.

```
subroutine ddaef (y,f,iw,w)
double precision y(*),f(*),w(*)
integer          iw(*)

f(1) =   y(2)*y(3)         ! Evaluate the derivative
f(2) =  -y(1)*y(3)
f(3) =  -0.51d0*y(1)*y(2)

return
end

subroutine ddaeo (y,f,iw,w)
double precision y(*),f(*),w(*)
integer          iw(*)

double precision tfinal
integer          neqn
parameter        (neqn = 3, tfinal = 25.d0)
double precision deltat
parameter        (delta = 5.d0)

! Print the time and the solution. If the final output point
! has not been reached, add deltat to w(1).

if (w(1) .eq. 5) then
   write (6,'(4x,''t'',11x,''y(1)'',11x,''y(2)'',11x,''y(3)'')')
   write (6,'(54(''-''))')
   write (6,'()')
end if
write (6,'(f8.6,1x,3(e14.7,1x))') w(1),y(1),y(2),y(3)
write (6,'()')

if (w(1) .lt. tfinal) then
   w(1) = w(1) + deltat
end if

return
end
```

Figure 2: A sample derivative (top) and output (bottom) subroutine for Example 1.

This figure was cited in Example 1 on page 5.