1 SUMMARY

To integrate a set of **first order ordinary differential equations**

$$y_i' = f_i(y_1,y_2,..,y_n,x) \qquad i=1,2,...,n$$

over a given range $x_0 \le x \le x_e$ given initial values $y_i(x_0)=y_i^0$ at $x=x_0$. The Runge-Kutta method due to Merson is used and the steplengths are controlled automatically by the subroutine so that at each step the truncation error should satisfy an accuracy requirement specified by the user. The Merson truncation error estimates are used to determine step lengths. The accuracy is not guaranteed.

The range is specified by giving the spacing Δx and m the number of equally spaced print points $x=x_0+k\Delta x$ $k=0,1,2,.,m$, needed to cover the range. When the integration reaches each of these points an output subroutine is called to print out the current values. This may be either the standard output subroutine provided with `DA02A/AD` or one provided by the user. There are additional options to either do no printing or print at intermediate integration points as well as at the print points. When the end of the range is reached the integration can be continued by re-entering `DA02A/AD` with the current point x and y values unchanged.

The user must provide a subroutine to compute values of the derivative functions $f_i(y_1,y_2,..,y_n,x)$ $i=1,2,...,n$.

ATTRIBUTES — **Version:** 1.0.0. **Remark:** The user should also consider the merits of more powerful subroutines like `DC03AD` and `DC04AD` which have been shown to be superior subroutines for many types of problem and generally competitive for most others. **Types:** `DA02A; DA02AD.` **Calls:** `FD05`. **Original date:** March 1972. **Origin:** A.R.Curtis and A.B.Smith*, Harwell.

2 HOW TO USE THE PACKAGE

2.1 The argument list and calling sequence

The single precision version:

```
CALL DA02A(Y,N,X,DX,M,H,W,EPS,*n)
```

The double precision version:

```
CALL DA02AD(Y,N,X,DX,M,H,W,EPS,*n)
```

`Y` is a `REAL` (`DOUBLE PRECISION` in the D version) array of length at least n, which must be set by the user to the current values of y_i, $i=1,2,...,n$. Initially the user must set `Y(I), I=1,N` to the initial values y_i^0 $i=1,2,...,n$. On return from the subroutine the array will contain the values corresponding to the end point.

`N` is an `INTEGER` variable which must be set by the user to n , the number of equations.

`X` is a `REAL` (`DOUBLE PRECISION` in the D version) variable and is used for the current value of x. Initially `X` must be set by the user to the initial point x_0. On return from the subroutine it will be set to the last point $x_e = x_0 + m\Delta x$.

`DX` is a `REAL` (`DOUBLE PRECISION` in the D version) variable which must be set by the user to Δx the step size defining the output points. The output subroutine will be called at points $x=x_0+k\Delta x$, $k=0,1,2,.,m$.

`M` is an `INTEGER` variable which must be set by the user to specify the number of output points $x=x_0+k\Delta x$, $k=0,1,2,.,m$. There are three options

(a) if the output subroutine is to be called only at every output point set `M` to m.

(b) if the output subroutine is to be called at every integration step in addition to every output point set `M` to

$-m$. **N.B.** This may produce a lot of output and the user is advised to use this for debug purposes only.

(c) if the output subroutine is not to be called at all set `M` to zero. The range of integration in this case will be just `DX`.

More information about controlling the calls to the output subroutine is given in §2.3.

`H` is a `REAL` (`DOUBLE PRECISION` in the D version) variable which must be set by the user to h_0 the initial steplength used in the integration. The steplength is controlled by the subroutine and so this initial value is not critical as it should soon be adjusted by the steplength control algorithm. A value of `H=DX` will often be as good a choice as any. On return `H` will have been set to the steplength used on the last step attempted (on an error return this will not have been a successful one).

`W` is a `REAL` (`DOUBLE PRECISION` in the D version) array of length at least $6n$ which is used by the subroutine as workspace. On return the first n elements will be set to the Merson error estimates from the last step.

`EPS` is a `REAL` (`DOUBLE PRECISION` in the D version) variable which must be set by the user to specify ε the maximum truncation error to be tolerated at each step. Set `EPS` to ε if the accuracy required is to be absolute and set `EPS` to $-\varepsilon$ if it is to be relative.

`*n` which must be set by the user to specify the Fortran statement to which control is returned in the event of an error, e.g. if a return to statement number 10 was required then code

```
CALL DA02A(.....,*10)
```

The error return is made if the steplength has had to be reduced to such an extent that it has become insignificant compared to the current value of x, i.e. when $x+h=x$ to the word length of the computer, where h is the steplength. If this error occurs a diagnostic message is printed.

2.2 The subroutine to evaluate the derivatives

The user **must provide** a subroutine to calculate values of the derivative functions $f_i(y_1,y_2,..,y_n,x)$ $i=1,2,...,n$. This must be called `DYBDX` and have the form

```
SUBROUTINE DYBDX(Y,F,N,X)
DIMENSION Y(N),F(N)
```

`Y` is a `REAL` (`DOUBLE PRECISION` in the D version) array containing the n current values of y_i, $i=1,2,...,n$.

`F` is a `REAL` (`DOUBLE PRECISION` in the D version) array in which the user must pass back the values of the derivative functions $f_i(y_1,y_2,..,y_n,x)$ $i=1,2,...,n$.

`N` is an `INTEGER` variable giving the value of n the number of y values.

`X` is a `REAL` (`DOUBLE PRECISION` in the D version) variable and gives the value of the current point x.

The user must not alter the contents of `Y, X` and `N.` At each step of the integration `DYBDX` is called 5 times. Additional information may be passed to it through Common or by giving it a secondary entry point. The name `DYBDX` is used by both single and double precision versions and the subroutine must be coded to be compatible with the precision of the version of `DA02` being used.

2.3 The Output subroutine

A standard output subroutine is included in `DA02` to print results and follow the progress of the integration. The user may **optionally provide** his own but it must be called `DA02B` (or `DA02BD`) and have the form

The single precision version:

```
SUBROUTINE DA02B(X,Y,N,IP)
DIMENSION Y(N),IP(2)
```

The double precision version:

```
      SUBROUTINE DA02BD(X,Y,N,IP)
      DIMENSION Y(N),IP(2)
```

`X,Y` and `N` are as defined in §2.2 for `DYBDX`.

`IP` is an `INTEGER` array of length 2 words which can be used to control the printing.

`IP(1)` gives the print step number starting at 0 for the initial values at $x=x_0$ and going up to m at $x=x_e$. To get the output subroutine called at all the intermediate integration points the user can reset `IP(1)` to be negative, this will operate until the next print point is reached when it will require resetting to negative again if these calls are to continue. Intermediate calls can be stopped at any time by resetting `IP(1)` positive.

`IP(2)` will contain the value of `NSTEP` (which could be negative). If the magnitude of `IP(2)` is set less than or equal to the current print number the integration will be terminated at the current print point, or the next print point if the call to the output subroutine is an intermediate one. The range may be dynamically extended by increasing the magnitude of `IP(2)`. Unless any of these special effects are desired `IP` need not be altered.

3 GENERAL INFORMATION

Use of Common: none.

Workspace: $6n$ words provided by the user in the array `W`.

Other subroutines: calls `DA01A/AD`, `FD05A/AD`, `I_AMAX`, an output subroutine `DA02B/BD` which can be either the standard one or one provided by the user, and calls a user supplied subroutine `DYBDX` for derivative calculations. Note that `DA02A/AD` contains its own copy of the deprecated routine `DA01A/AD`.

Input/Output: the standard output subroutine prints the current x value and corresponding y_i, i=1,2,...,n values. If x is at a print point the print point number is also printed, at intermediate points this is replaced by an * character. There are no page throws. On an error exit a diagnostic is printed.

4 METHOD

`DA02A/AD` is a driver subroutine for the subroutine `DA01A/AD` which uses the 4th order Runge-Kutta method due to R.H. Merson, see (1947) Proc. of Symp. on Data Processing, W.R.E., South Australia, and uses 5th order terms to estimate truncation errors.

Suppose the Merson error estimates to be e_i, i=1,2,...,n and the accuracy requirement is ε, then one of the adjustment factors

(i) absolute accuracy $$w=\frac{1}{|\varepsilon|}\max_i\{|e_i|\}$$

(ii) relative accuracy $$w=\frac{1}{|\varepsilon|}\max_i\left\{\left|\frac{e_i}{y_i}\right|\right\}$$

is computed. If the errors are not acceptable, i.e. $w>1$, the integration is repeated with a smaller step. Acceptable or not the next step h is always computed using the formula $h_{new}=h_{old}\,(0.9w^{-0.2})$. To be on the safe side the adjustment factor is restricted to $10^{-5}\leq\omega\leq10^{8}$.