1 SUMMARY

To solve the **two point boundary value** problem for the **non-linear parabolic partial** differential equation

$$\frac{\partial u}{\partial t} = a\frac{\partial^2 u}{\partial x^2} + b\frac{\partial u}{\partial x} + cu + d \quad x_1 \le x \le x_n$$

where a, b, c and d are functions of x, t, u and $\partial u/\partial x$, given boundary conditions of the form

$$p\frac{\partial u}{\partial t} = q + ru + s\frac{\partial u}{\partial x}$$

at the points x_1 and x_n. Given δt and the solution $u(x,t)$ at t, on n equally spaced points x_i, i=1,2,...,n the subroutine advances the integration one time step to obtain $u(x,t+\delta t)$.

Using the solution at t as a first approximation the equation is linearized and successively re-linearized until the required accuracy is obtained. The library subroutine `DP11` is used to solve the linearized equation which in turn uses `DD11`.

The user must provide a subroutine to compute the functions a, b, c and d given current values of x, t, u and $\partial u/\partial x$.

ATTRIBUTES — **Version:** 1.0.0. **Types:** `DP12A`; `DP12AD`. **Calls:** `DP11`, `TA03` and `TD01`. **Original date:** June 1966. **Origin:** P.Hallowell, Atlas Lab., Chilton, Oxon.

2 HOW TO USE THE PACKAGE

2.1 The argument lists

The single precision version

```
CALL DP12A(U,X1,XN,N,T0,DT,UX1,UXN,DUDT,E,LIM,UA,NUM,K,W)
```

The double precision version

```
CALL DP12AD(U,X1,XN,N,T0,DT,UX1,UXN,DUDT,E,LIM,UA,NUM,K,W)
```

`U` is a `REAL` (`DOUBLE PRECISION` in the D version) array of length at least n. The user must set the first n elements of `U` to the solution of the equation at $t=t_0$. That is, `U(i)` $= u(x_i,t_0)$, i=1, 2,...,n. On return, these elements will be overwritten with the solution at $t_0+\delta t$.

`X1` and `XN` are `REAL` (`DOUBLE PRECISION` in the D version) variables which must be set by the user to the end points of the range of x, i.e. $x_1 \le x \le x_n$.

`N` is an `INTEGER` variable which must be set by the user to n the number of x points covering the range $x_1 \le x \le x_n$. The n points will include the end points and be equally spaced.

`T0` is a `REAL` (`DOUBLE PRECISION` in the D version) variable which must be set by the user to the value of t_0 corresponding to the solution provided in `U`. On return `T0` will be set to $t_0 + \delta t$ (δt passed in `DT`).

`DT` is a `REAL` (`DOUBLE PRECISION` in the D version) variable which must be set by the user to the time step δt which the integration is to be advanced.

`UX1` is a `REAL` (`DOUBLE PRECISION` in the D version) array of length 4 which must be set by the user to the boundary conditions at $x=x_1$. Given boundary conditions of the form

$$p\frac{\partial u}{\partial t} = q + ru + s\frac{\partial u}{\partial x}$$

the first four elements of UX1 must be set to:

UX1(1) = p, UX1(2) = q, UX1(3) = r, UX1(4) = s.

UXN is a REAL (DOUBLE PRECISION in the D version) array of length 4 which must be set by the user to the boundary conditions at $x = x_n$ (treat as argument UX1).

DUDT is a REAL (DOUBLE PRECISION in the D version) array of length at least n which is used to pass values of $\partial u/\partial t$ at t_0 to the subroutine when they are known (see argument NUM for further details). On return DUDT will contain the values of $\partial u/\partial t$ at $t_0+\delta t$ unless the subroutine failed to achieve the requested accuracy and then it returns DUDT set to $\partial u/\partial t$ at t_0.

E is a REAL (DOUBLE PRECISION in the D version) array of length at least n which is normally set by the subroutine to the differences between the first approximation to $u(x_i,t+\delta t)$, i=1, 2,..., n, and the final values returned in U. That is, E contains the difference between the solution ignoring 3rd order and higher order differences and the final solution obtained by taking these into account, see DD11 for more details.

The exceptions occur when LIM < 0 (then the elements of E are always set to zero) and when the subroutine fails to achieve the requested accuracy. In the second case the elements of E are set to the differences between the last two linearisations if the failure occurred during the application of 3rd and 4th differences, otherwise they are set to zero.

LIM is an INTEGER variable which must be set by the user to a nonzero value to control the calculation.

If LIM > 0 then LIM is taken to be the limit on the number of times that the correction terms based on the 3rd and 4th differences are calculated. When the required accuracy is reached NUM is set equal to the number of times the correction terms have been calculated. If the correction terms are calculated LIM times without reaching the required accuracy the subroutine returns with NUM set to –1 and DUDT set to $\partial u/\partial t$ at t_0.

If LIM=0 3rd and higher differences are not taken into account; that is the subroutine returns after solving the finite difference equations based on the 1st and 2nd differences. In this case both E and NUM are set equal to zero.

LIM is always returned set to its absolute value.

UA is a REAL (DOUBLE PRECISION in the D version) variable which must be set by the user to the absolute accuracy required. It may be altered by the routine.

NUM is an INTEGER variable which set both by the user and the subroutine.

The user must set NUM to a value greater than zero if values of $\partial u/\partial t$ at t_0 are being supplied in the first n elements of DUDT. If these values are not known NUM should be set to zero and the subroutine will calculate its own approximations.

On return the subroutine will set NUM to zero or to the number of times correction terms have been calculated depending on the setting of LIM (see argument LIM).

K is an INTEGER variable which must be set by the user to a non-negative integer. If K is set positive the subroutine will provide the subroutine FUNCTS with an approximation to $\partial u/\partial x$ at $t = t_0+\delta t$, otherwise if K=0 the approximation is not provided. Thus K should be set greater than zero unless none of the coefficients a, b, c and d involve $\partial u/\partial x$ in which case K should be set to zero.

W is a REAL (DOUBLE PRECISION in the D version) array of length at least 19n+6 words used by the routine for workspace.

2.2 Calculating the equation coefficients

The user must provide a subroutine to calculate the coefficients a, b, c and d in the equation at given values of u, $\partial u/\partial x$, x and t, see section 1. The provision of u and $\partial u/\partial x$ is controlled by the DP12 argument K (see §2.1). The subroutine must take the form

The single precision version

```
SUBROUTINE FUNCTS(A,B,C,D,T,X,IU,IDUDX,W)
REAL A,B,C,D,T,X
INTEGER IU,IDUDX
REAL W(*)
```

The double precision version

```
SUBROUTINE FUNCTS(A,B,C,D,T,X,IU,IDUDX,W)
DOUBLE PRECISION A,B,C,D,T,X
INTEGER IU,IDUDX
DOUBLE PRECISION W(*)
```

where W is the workspace array passed as the last argument to DP12A/AD. The subroutine must set A$=a(x,t,u,\partial u/\partial x)$, B$=b(x,t,u,\partial u/\partial x)$, C$=c(x,t,u,\partial u/\partial x)$ and D$=d(x,t,u,\partial u/\partial x)$, given the value of x in X, the value of t in T and the value of u in W(IU). If an approximation to $\partial u/\partial x$ has been requested with K>0 it is located in W(IDUDX), otherwise IDUDX should not be used. Note: the name of the subroutine must be FUNCTS and this must be used for both single and double precision versions.

3 GENERAL INFORMATION

Use of common: None.

Workspace: 19n+6 words provided in W.

Other routines called directly: DP11A/AD, TA03A/AD and TD01A/AD are called. The user must provide a subroutine called FUNCTS to calculate the equation coefficients.

Input/output: None.

4 METHOD

Using the solution at $t=t_0$ as an initial approximation to the solution at $t=t_0+\delta t$, the equation is linearised and then successively relinearised until the required accuracy is obtained. The documentation for the library subroutines DD11 and DP11 contained further information on the finite-difference scheme used. At this stage the solution has ignored 3rd and higher differences; if required these can then be used to calculate correction terms to improve the solution. This will normally be necessary unless the step length used is small enough to ensure that 3rd differences are less than the accuracy required.