1 SUMMARY

Given functions $f_i(x_1, x_2, ..., x_n)$, i=1,2,...,m, this subroutine **evaluates an approximation to the Jacobian matrix** $\mathbf{J} = \{\partial f_i/\partial x_j\}$ using finite differences. The routine is intended for the case when **J** is sparse or band structured and has additional entries which given the functions $f_i(\mathbf{x})$ construct the sparsity pattern for **J**.

Derivatives are estimated by

$$\partial f_i/\partial x_j \approx \frac{1}{2h_j}\{f_i(x_1,..., x_j+h_j,..., x_n - f_i(x_1,..., x_j-h_j,...,x_n)\}$$

where the steplengths h_j are automatically chosen by the routine within bounds specified by the user.

The method is described by Curtis, Powell, and Reid, J. Inst. Maths. Applics. **13** (1974) 117-120, and Curtis and Reid, J. Inst. Maths. Applics. **13** (1974) 121-126.

ATTRIBUTES — Version: 1.1.0. **Types:** Real (single, double). **Calls:** `FD15`. **Original date:** June 2001. **Remark:** `TD22` is a threadsafe version of `TD12`. **Origin:** J. K. Reid, Rutherford Appleton Laboratory.

2 HOW TO USE THE PACKAGE

2.1 Entries and argument lists

Although there are both single and double precision versions of the routine available, the user is **strongly advised** to use the double precision version unless single precision on his or her machine actually means 8-byte arithmetic.

'Reverse communication' is employed to permit the user to access any data needed to calculate the functions f_i. To use `TD22` he or she must call it repeatedly under the control of the parameter `IFLAG`. The initial call must be with `IFLAG` set to unity, the point at which evaluation is required placed in the array `X`, and the corresponding function values placed in the array `F`. On a return with `IFLAG` > `0`, code for placing in `F` the values of the functions at the point then held in `X` must be executed and the subroutine recalled. All other arguments must not be altered. A return with `IFLAG=0` indicates successful termination and the error condition is marked by `IFLAG` = `-1`.

There are four entries:

(a) `TD22AI/ID` must be called before any of the other entries are called to initialize the user options and the private workspace.

(b) `TD22A/AD` calculates the Jacobian matrix when the sparsity pattern is known.

(c) `TD22B/BD` constructs the sparsity pattern by returning to the user for n different function values.

(d) `TD22C/CD` constructs the sparsity pattern of a band matrix.

2.1.1 Initialization

The single precision version

```
CALL TD22I(ICNTL,CNTL,KEEP,RKEEP)
```

The double precision version

```
CALL TD22ID(ICNTL,CNTL,KEEP,RKEEP)
```

`ICNTL` is an `INTEGER` array of length 5 that need not be set by the user. On return it contains default values (see Section 2.2 for details).

`CNTL` is a `REAL` (`DOUBLE PRECISION` in the D version) array of length 7 that need not be set by the user. On return it contains default values (see Section 2.2 for details).

`KEEP` is an `INTEGER` array of length 20 that need not be set by the user. It used by `TD22` as private workspace and must not be altered by the user.

`RKEEP` is a `REAL` (`DOUBLE PRECISION` in the D version) array of length 10 that need not be set by the user. It used by `TD22` as private workspace and must not be altered by the user.

2.1.2 To calculate the approximate Jacobian

The single precision version

```
CALL TD22A(M,N,IRN,IP,H,X,F,HMAX,A,IG,W,IFLAG,ICNTL,CNTL,INFO,KEEP,RKEEP)
```

The double precision version

```
CALL TD22AD(M,N,IRN,IP,H,X,F,HMAX,A,IG,W,IFLAG,ICNTL,CNTL,INFO,KEEP,RKEEP)
```

`M` is an `INTEGER` variable set by the user to the number of functions $f_i(\mathbf{x})$, that is the number of rows in the Jacobian matrix. It is not altered by the subroutine.

`N` is an `INTEGER` variable set by the user to the number of variables x_j, that is the number of columns in the Jacobian matrix. It is not altered by the subroutine.

`IRN` is an `INTEGER` array of size the number of nonzeros in the Jacobian matrix. It must be set by the user to hold the row indices of these nonzeros, stored by columns, for example,

$$\frac{\partial f_1}{\partial x_1}, \frac{\partial f_3}{\partial x_1}, ..., \frac{\partial f_m}{\partial x_1}, \frac{\partial f_2}{\partial x_2}, ...$$

`IRN` is not altered by the subroutine.

`IP` is an `INTEGER` array of size `N+1` that must be set by the user so that `IP(J)` is the position in `IRN` of the start of column `J` of the Jacobian matrix, `J=1,2,...,N`, and `IP(N+1)` is the position of the first unused location in `IRN`. It is not altered by the subroutine.

`H` is a `REAL` (`DOUBLE PRECISION` in the D version) array of size `N` required by `TD22A/AD` to specify the step lengths to be used when estimating the derivatives by finite differences (see Section 4). These step lengths are adjusted by `TD22A/AD` to give good accuracy and are left set on the assumption that `TD22A/AD` is likely to be called later with similar values for the variables `X(J), J=l,2,...,N`, in which case no resetting is necessary.

`X` is a `REAL` (`DOUBLE PRECISION` in the D version) array of size `N` that must be set initially by the user to the point x_j, j=1,2,...,n, at which the approximate Jacobian is required. It is altered by the subroutine between intermediate calls and is finally restored to its initial value.

`F` is a `REAL` (`DOUBLE PRECISION` in the D version) array of size `M` whose elements must be set by the user initially and before every intermediate call to $f_i(x_1, x_2,..., x_n)$, i=1,2,...,m. It is not altered by the subroutine, except on the final return when it is restored to its initial value.

`HMAX` is a `REAL` (`DOUBLE PRECISION` in the D version) array of size `N` set by the user to upper bounds for h_j, j=1,2,...,n, but if `HMAX(1)<0` then all bounds are taken as |`HMAX(1)`| and the size may be one. Lower bounds for h_j are taken as `EPS1` times the corresponding upper bound, where `EPS1` specifies the relative machine precision (see `CNTL(5)` in Section 2.2).

`A` is a `REAL` (`DOUBLE PRECISION` in the D version) array of size the number of nonzero derivatives into which the nonzero derivatives are placed.

`IG` is an `INTEGER` work array of size at least `2*N+1`. `IG(1)` should be set to zero before a first entry to `TD22A/AD` for a particular sparsity structure; during this entry `IG(1),...,IG(2*N+1)` are found and are required for subsequent entries to `TD22A/AD` for this sparsity structure.

`W` is a `REAL` (`DOUBLE PRECISION` in the D version) array of size `M+2*N` used as workspace.

`IFLAG` is an `INTEGER` variable which must be set by the user to the value 1 on the first entry to the subroutine. On return, the value 0 indicates that the calculation is complete and positive values indicate that the subroutine should be recalled after calculating $f_i(x_1, x_2,..., x_n)$, i=1,2,...,m, and placing them in `F` without changing other arguments.

`ICNTL` is an `INTEGER` array of length 5 that contains control parameters. Default values for the elements are set by `TD22I/ID.` Details of the control parameters are given in Section 2.2. This argument is not altered by the subroutine.

`CNTL` is a `REAL` (`DOUBLE PRECISION` in the D version) array of length 7 that contains control parameters. Default values for the elements are set by `TD22I/ID.` Details of the control parameters are given in Section 2.2. This argument is not altered by the subroutine.

`INFO` is an `INTEGER` array of length 5 that need not be set by the user. On return, it holds information on the calculation, see Section 2.2.

`KEEP` is an `INTEGER` array of length 20 used by `TD22` for private workspace. It must be initialized by calling `TD22I/ID` and must not be altered by the user.

`RKEEP` is a `REAL` (`DOUBLE PRECISION` in the D version) array of length 10 used by `TD22` for private workspace. It must be initialized by calling `TD22I/ID` and must not be altered by the user.

2.1.3 To construct the sparsity pattern using function values

The single precision version

```
CALL TD22B(M,N,IRN,IP,H,X,F,W,IA,IFLAG,ICNTL,INFO,KEEP,RKEEP)
```

The double precision version

```
CALL TD22BD(M,N,IRN,IP,H,X,F,W,IA,IFLAG,ICNTL,INFO,KEEP,RKEEP)
```

`M` is an `INTEGER` variable set by the user to the number of functions $f_i(\mathbf{x})$, that is the number of rows in the Jacobian matrix. It is not altered by the subroutine.

`N` is an `INTEGER` variable set by the user to the number of variables x_j, that is the number of columns in the Jacobian matrix. It is not altered by the subroutine.

`IRN,IP` are `INTEGER` arrays as described in section 2.1.2. They need not be set on entry and on final return hold the sparsity structure of the Jacobian matrix.

`H` is a `REAL` (`DOUBLE PRECISION` in the D version) array of size `N` required by `TD22B/BD` to specify the step lengths to be used when altering the components of **x** to find the dependencies. It is not altered by the subroutine.

`X` is a `REAL` (`DOUBLE PRECISION` in the D version) array of size `N` which must be set initially by the user to the point x_j, j=1,2,...,n, at which the approximate Jacobian is required. It is altered by the subroutine between intermediate calls and is finally restored to its initial value.

`F` is a `REAL` (`DOUBLE PRECISION` in the D version) array of size `M` whose elements must be set by the user initially and before every intermediate call to $f_i(x_1, x_2,..., x_n)$, i=1,2,...,m. It is not altered by the subroutine.

`W` is a `REAL` (`DOUBLE PRECISION` in the D version) array of size `M` used as workspace.

`IA` is an `INTEGER` variable specifying the size of the arrays `A` and `IRN`. It is not altered by the subroutine.

`IFLAG` is an `INTEGER` variable which must be set by the user to the value 1 on the first entry to the subroutine. On return, the value 0 indicates that the calculation is complete, and positive values indicate that the subroutine should be recalled after calculating $f_i(x_1, x_2,..., x_n)$, i=1,2,...,m, and placing them in `F` without changing other arguments. On an error, see `INFO(1)` in Section 2.2, `IFLAG` has a negative value.

`ICNTL` is an `INTEGER` array of length 5 that contains control parameters. Default values for the elements are set by

`TD22I/ID`. Details of the control parameters are given in Section 2.2. This argument is not altered by the subroutine.

`INFO` is an `INTEGER` array of length 5 that need not be set by the user. On return, it holds information on the calculation, see Section 2.2.

`KEEP` is an `INTEGER` array of length 20 used by `TD22` for private workspace. It must be initialized by calling `TD22I/ID` and must not be altered by the user.

`RKEEP` is a `REAL` (`DOUBLE PRECISION` in the D version) array of length 10 used by `TD22` for private workspace. It must be initialized by calling `TD22I/ID` and must not be altered by the user.

2.1.4 To construct the sparsity pattern of a band matrix

The single precision version

```
CALL TD22C(N,IRN,IP,IA,MBD,ICNTL,INFO)
```

The double precision version

```
CALL TD22CD(N,IRN,IP,IA,MBD,ICNTL,INFO)
```

`N` is an `INTEGER` variable set by the user to the number of variables x_j, that is the number of columns in the Jacobian matrix. It is not altered by the subroutine.

`IRN,IP` are `INTEGER` arrays as described in section 2.1.2. They need not be set on entry and on final return hold the sparsity structure of the Jacobian matrix.

`IA` is an `INTEGER` variable specifying the size of the arrays `A` and `IRN`. It is not altered by the subroutine.

`MBD` is an `INTEGER` variable which must be set by the user to the semi-bandwidth of the Jacobian, so that $\partial f_i/\partial x_j = 0$ if $|i-j| \geq$ `MBD`. It is not altered by the subroutine.

`ICNTL` is an `INTEGER` array of length 5 that contains control parameters. Default values for the elements are set by `TD22I/ID`. Details of the control parameters are given in Section 2.2. This argument is not altered by the subroutine.

`INFO` is an `INTEGER` array of length 5 that need not be set by the user. On return, it holds information on the calculation, see Section 2.2.

2.2 The control and information arrays

The elements of the arrays `ICNTL` and `CNTL` control the action of the subroutines. Default values are set by `TD22I/ID`.

`ICNTL(1)` specifies the unit number to be used to output error messages. A negative value will suppress output. It has a default value of 6, set by `TD22I/ID`.

`ICNTL(2)` has default value 2, set by `TD22I/ID`. This may be reset by the user to 1 to economise in the number of function calls. In that case, no estimation of errors is made, no adjustments are made to step sizes, and the less accurate one-sided difference approximation

$$\frac{\partial f_i}{\partial x_j} \approx \frac{f_i(x_1,x_2,...,x_j+h_j,...,x_n) - f_i(x_1,x_2,...,x_n)}{h_j}$$

is used in place of the one given in Section 4.

`ICNTL(3)` to `ICNTL(5)` are not used at present and should not be altered.

`CNTL(1)` is used to pass a value for the variable `UMIN` described in Section 4. Its default value is 10, set by `TD22I/ID`.

`CNTL(2)` is used to pass a value for the variable `UAIM` described in Section 4. Its default value is 100, set by `TD22I/ID`.

`CNTL(3)` is used to pass a value for the variable `UMAX` described in Section 4. Its default value is 1000, set by `TD22I/ID`.

`CNTL(4)` is used to pass a value for the variable `EPS` described in Section 4. Its default value is calculated by `TD22I/ID` using `FD15`.

`CNTL(5)` is used to pass a value for the variable `EPS1` described in Section 4. It is set by set by `TD22I/ID` to the same default value as `EPS`.

`CNTL(6)` holds the relative roundoff in values of *f*. It is set by set by `TD22I/ID` to half of the default value of `EPS`. If *f* is calculated by a method with relative accuracy `EPSF`, this value should be set in `CNTL(6)`.

`CNTL(7)` is not used at present and should not be altered.

The `INFO` array argument which must be of length 5 is used to return information back to the user.

`INFO(1)` is used as an error indicator. It is set to zero when there are no errors otherwise it is set to a nonzero value. At present only one error is possible, i.e. when `IA` is found to be too small by `TD22B/BD` or `TD22C/CD`; then `INFO(1)` is returned set to 1, `INFO(2)` set to a suggested value for `IA` and an error message is output on unit `ICNTL(1)`. For `TD22B/BD`, the calculation may be continued by copying the data in `IRN` to an array of size at least `INFO(2)` and recalling the subroutine with the larger array, its size in `IA`, and `IFLAG` set to 2. For `TD22C/CD`, the calculation will be successful if repeated with a new array `IRN` of size `INFO(2)`.

`INFO(2)` is used to return a supplementary value associated with the error indicated by `INFO(1)`.

`INFO(3)` is only set by `TD22A/AD` and returns the number of groups used (see Section 4).

`INFO(4)` and `INFO(5)` are not used at present.

3 GENERAL INFORMATION

Use of common: none.

Workspace: passed as arguments, see `IG`,`W`, `KEEP`, and `RKEEP`.

Other routines called directly: the function `FD15A/AD` is called by `TD22I/ID`.

Input/output: error messages, see `ICNTL(1)` and `INFO(1)` in Section 2.2.

4 METHOD

On entry to `TD22C/CD`, the appropriate sparsity pattern is set up in a straightforward way and `IA` is checked.

On entry to `TD22B/BD`, the variable `X(1)` is changed to `X(1)+H(1)` and the corresponding changes to **f** are taken to indicate the nonzeros in the first column of the Jacobian matrix. `X(1)` is then restored to its original value and the same procedure is applied to each column. Note that it is important to avoid any special values (for example, zeros) of `X(J)`, `J=1,N`, which cause freak zeros in the Jacobian matrix and it is important that the steps `H(J)` be large enough so that all the appropriate changes to **f** are noticeable.

If `IA` is found to be too small, the number of nonzeros in the current column, `J` say, is counted so that the total number of nonzeros in columns 1 to `J` is known. This allows us to estimate the total by multiplying by `REAL(N)/J`. To give a little leeway, we use `REAL(N+1)/J` when calculating an estimate to return in `INFO(2)`.

On entry to `TD22A/AD`, unless `ICNTL(2)` has the value 1 (see Section 2.2), the derivatives are estimated by the difference formula

$$\frac{\partial f_i}{\partial x_j} \approx \frac{f_i(x_1, x_2, \ldots, x_j + h_j, \ldots, x_n) - f_i(x_1, x_2, \ldots, x_j - h_j, \ldots, x_n)}{2h_j}.$$

The truncation error in this approximation is estimated using the second difference and the roundoff error is estimated from the effect on $f_i(\mathbf{x})$ of roundoff-level changes to x_j. Each h_j, is adjusted so that, if possible, the greatest ratio of roundoff error to truncation error for a column of the Jacobian is near `UAIM` and within the range (`UMIN`,`UMAX`). `EPS` is the greatest relative roundoff in a single operation and is needed for the roundoff error estimate. The steps h_j are not permitted to leave the range

$$[\max(\mathtt{EPS}*|x_j|, \mathtt{EPS1*HMAX(J)}), \mathtt{HMAX(J)}].$$

To economise on the number of function evaluations, the columns of the Jacobian are formed into groups such that no two columns of the same group have a nonzero in the same row so that all the changes h_j corresponding to one group may be made together. The array `IG` is used to store store this grouping. For further details of `TD22`, see the three references given below.

References

Curtis, A.R., Powell, M.J.D. and Reid, J.K. (1974). On the estimation of sparse Jacobian matrices. J. Inst. Maths. Applics. **13** (1974) 117-120.

Reid, J.K. (1972). Fortran subroutines for the solution of sparse systems of nonlinear equations. Harwell report R.7293.

Curtis, A.R. and Reid, J.K. (1972). The choice of step lengths when using differences to approximate Jacobian matrices. J. Inst. Maths. Applics. **13** (1974) 121-126.

5 EXAMPLE OF USE

As a simple example, suppose that we wish to estimate the Jacobian matrix of the functions

$$\begin{aligned} f_1(\mathbf{x}) &= x_1x_2 \\ f_2(\mathbf{x}) &= x_1 + x_3^2 \\ f_3(\mathbf{x}) &= x_4x_5 + x_6 \\ f_4(\mathbf{x}) &= x_3 - x_4/x_5 \\ f_5(\mathbf{x}) &= 1 - 2x_6 \end{aligned}$$

at the point $\mathbf{x} = (1, 2, ..., 6)^T$. The Jacobian has the sparsity pattern

$$\begin{matrix} \times & \times & 0 & 0 & 0 & 0 \\ \times & 0 & \times & 0 & 0 & 0 \\ 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & \times & \times & \times & 0 \\ 0 & 0 & 0 & 0 & 0 & \times \end{matrix} \cdot$$

We may then obtain the required Jacobian values using the following piece of code.

```
C     TEST OF TD22
      INTEGER M, N, IA, LIP, LW, LIG
      PARAMETER ( M = 5, N = 6, IA = 30 )
      PARAMETER ( LIP = N + 1, LW = M + 2*N, LIG = 2*N + 1 )
      INTEGER   I, IRN( IA ), IP( LIP ), IG( LIG ), IFLAG, J, J1, J2
      DOUBLE PRECISION X( N ), F( M ), A( IA ), W( LW ),
     *                 H( N ), HMAX( N )
      DOUBLE PRECISION CNTL(7), RKEEP(10)
      INTEGER INFO(5), ICNTL(5), KEEP(20)
      EXTERNAL         FUNC
      DATA        X / 1.0D+0, 2.0D+0, 3.0D+0, 4.0D+0, 5.0D+0, 6.0D+0 /
      DATA        H / 1.0D-1, 2.0D-1, 3.0D-1, 4.0D-1, 5.0D-1, 6.0D-1 /
      CALL TD22ID( ICNTL, CNTL, KEEP, RKEEP )
      HMAX( 1 ) = - 1.0D+0
      IFLAG = 1
    5 CONTINUE
         CALL FUNC( X, F )
         CALL TD22BD( M, N, IRN, IP, H, X, F, W, IA, IFLAG, ICNTL, INFO,
     +                KEEP, RKEEP )
      IF( IFLAG.GT.0) GO TO 5
      WRITE( 6, 2000 )
      DO 10 I = 1, N
         J1   = IP( I )
         J2   = IP( I + 1 ) - 1
         IF ( J1 .LE. J2 ) WRITE( 6, 2010 ) I, ( IRN( J ), J = J1, J2 )
   10 CONTINUE
      IG( 1 ) = 0
   15 CONTINUE
         CALL FUNC( X, F )
         CALL TD22AD( M, N, IRN, IP, H, X, F, HMAX, A, IG, W, IFLAG,
     +                ICNTL, CNTL, INFO, KEEP, RKEEP )
      IF( IFLAG.GT.0) GO TO 15
      WRITE( 6, 2020 )
      DO 20 I = 1, N
         J1   = IP( I )
         J2   = IP( I + 1 ) - 1
         IF ( J1 .LE. J2 ) WRITE( 6, 2030 ) I, ( A( J ), J = J1, J2 )
   20 CONTINUE
      STOP
 2000 FORMAT( /, ' On return from TD22BD :', / )
 2010 FORMAT( ' column ', I2, ' of the Jacobian has',
     *        ' nonzero(s) in row(s)', 4I2, ( 40I2 ) )
 2020 FORMAT( /, ' On return from TD22AD :', / )
 2030 FORMAT( ' value(s) of nonzero(s) in column', I2,
     *        ' of the Jacobian :',  1P, 2D9.1, ( 1P, 8D9.1 ) )
      END
      SUBROUTINE FUNC( X, F )
      DOUBLE PRECISION X( 6 ), F( 5 )
      F( 1 ) = X( 1 ) * X( 2 )
      F( 2 ) = X( 1 ) + X( 3 ) ** 2
      F( 3 ) = X( 4 ) * X( 5 ) + X( 6 )
      F( 4 ) = X( 3 ) - X( 4 ) / X( 5 )
      F( 5 ) = 1.0D+0 - 2.0D+0 * X( 6 )
      RETURN
      END
```

This produces the following output:

```
 On return from TD22BD :

 column  1 of the Jacobian has nonzero(s) in row(s) 1 2
 column  2 of the Jacobian has nonzero(s) in row(s) 1
 column  3 of the Jacobian has nonzero(s) in row(s) 2 4
 column  4 of the Jacobian has nonzero(s) in row(s) 3 4
 column  5 of the Jacobian has nonzero(s) in row(s) 3 4
 column  6 of the Jacobian has nonzero(s) in row(s) 3 5

 On return from TD22AD :

 value(s) of nonzero(s) in column 1 of the Jacobian :  2.0D+00  1.0D+00
 value(s) of nonzero(s) in column 2 of the Jacobian :  1.0D+00
 value(s) of nonzero(s) in column 3 of the Jacobian :  6.0D+00  1.0D+00
 value(s) of nonzero(s) in column 4 of the Jacobian :  5.0D+00 -2.0D-01
 value(s) of nonzero(s) in column 5 of the Jacobian :  4.0D+00  1.6D-01
```

```
  value(s) of nonzero(s) in column 6 of the Jacobian :  1.0D+00 -2.0D+00
```