1 SUMMARY

The subroutine finds a real zero of a continuous real function $y(x)$ in a given interval $a \le x \le b$, where $y(a)$ and $y(b)$ have opposite signs. An attempt is made to find two arguments $x_a \le x_b$, which bracket the root x, such that $x_b - x_a \le \varepsilon$, where ε is specified by the user.

The subroutine returns to the calling program for values of the function $y(x)$. The user must set up the calling sequence as described in section 2.3 so that for any x in $[a,b]$, the value of $y(x)$ is calculated and then control passed back to `NB02A/AD`.

ATTRIBUTES — **Version:** 1.0.0. **Types:** `NB02A, NB02AD`. **Calls:** `FD05A`. **Original date:** July 1985. **Origin:** A.R.Curtis, Harwell.

2 HOW TO USE THE PACKAGE

2.1 Argument list

The single precision version.

```
CALL NB02A(K,A,B,EPS,X,Y)
```

The double precision version.

```
CALL NB02AD(K,A,B,EPS,X,Y)
```

`K` is an `INTEGER` variable which must be set by the user on the initial call. `K` should be set to 0 (or −1 if the values of $y(a)$ and $y(b)$ have been set in `COMMON`, see section 2.3). The value of `K` is changed by the subroutine, and on return to the calling program `K` will have the following values:

1 when the subroutine requires a value of $y(x)$,

2 when a root is found,

3 for an error return when $y(a)$ and $y(b)$ have the same sign.

`A,B` are `REAL` (`DOUBLE PRECISION` in the D version) variables which must be set by the user to specify the bounds a,b on the zero. They are not altered.

`EPS` is a `REAL` (`DOUBLE PRECISION` in the D version) variable which must be set by the user to specify a tolerance on the value of x for which $y(x)=0$, or to zero to obtain x to machine precision. The subroutine accepts x as a root if either:

(a) $x_a \le x \le x_b$, where $x_b - x_a \le \varepsilon$ and $y(x_a)$, $y(x_b)$ have opposite signs, or

(b) $x_a = x = x_b$ and $y(x)=0$.

`X` is `REAL` (`DOUBLE PRECISION` in the D version) variable whose value is altered by the subroutine. `X` need not be set by the user on the initial call. On a return with `K=1` the function value y(`X`) is required, and on a return with `K=2` the value of `X` is the calculated position of the zero.

`Y` is a `REAL` (`DOUBLE PRECISION` in the D version) variable , which need not be set by the user on the first call to the subroutine. On a return with `K=1` the user's program must set `Y` to y(`X`), and on a return with `K=2` the value of `Y` is equal to y(`X`).

2.2 Use of Common

The single precision version

```
COMMON/NB02B/XA,XB,YA,YB,IT
```

The double precision version

```
COMMON/NB02BD/XA,XB,YA,YB,IT
```

XA,XB are REAL (DOUBLE PRECISION in the D version) variables which must be set prior to a call with K=-1 to the bounds *a*,*b* on the zero. On return with K=2 they hold the best bracket found for the zero.

YA,YB are REAL (DOUBLE PRECISION in the D version) variables which must be set prior to a call with K=-1 to *y*(*a*) and *y*(*b*). On return with K=2 they hold *y*(XA) and *y*(XB).

IT is an INTEGER variable which is set to the number of returns with K=1.

2.3 Calling sequence

There two forms of the calling sequence, distinguished by the value of K on the first call. In the normal form K is set to 0 on the first entry and the first and second function values requested are *y*(A) and *y*(B). The alternative form (K=-1) is used when *y*(A) and *y*(B) are set in COMMON before the initial call.

Calling sequence for the K=0 form:

```
   Instructions to set A,B and EPS.
   K=0
10 CALL NB02A(K,A,B,EPS,X,Y)
   GO TO (20,30,40),K
20 Instructions to set Y=y(x).
   GO TO 10
30 Instructions following a normal return, with X as the required root.
40 Instructions following an error return, when y(A) and y(B) have the same sign.
```

Calling sequence for the K=-1 form:

```
   COMMON/NB02B/XA,XB,YA,YB,IT
   Instructions to set A,B and EPS.
   XA=A
   XB=B
   Instructions to set YA=y(A) and YB=y(B).
   K=-1
10 CALL NB02A(K,A,B,EPS,X,Y)
   (as before)
```

3 GENERAL INFORMATION

Use of common: Common area NB02B/BD is used, see §2.2.

Other routines called directly: None.

Input/output: None.

4 METHOD

The method is essentially the same as for NB01A/AD and consists of the following stages.

Firstly $y(a)$ and $y(b)$ are evaluated and an error return (K=3) is given if they have the same sign. This differs from NB01A/AD which includes a binary subdivision phase to try to find a bracket.

Secondly, having found an interval in which $y(x)$ changes sign, this interval is reduced in length by either linear interpolation for a root of $y(x)$, of by bisection. The interpolation may be based on the bracket or on the two most recently calculated values of $y(x)$. Bisection is used if an interpolation does not halve the smallest calculated function value, or if an interpolation gives a point that is outside the current bracket.

4.1 Accuracy

The subroutine normally achieves the requested accuracy, unless EPS is too small on entry when it returns with XA, XB differing as little as possible having regard to computer rounding errors. If a function value of zero is returned for some value of X then XA and XB will be set to X and YA and YB will be set to zero.

5 EXAMPLE OF USE

As a simple example the following code calls NB02AD to find the zero of the function $y=x\tan(x)-1$ in the region $0 \le x \le 1$.

```
      DOUBLE PRECISION A,B,EPS,X,Y
      DATA EPS/1.D-12/,A/0.D0/,B/1.D0/
      K=0
  10  CALL NB02AD(K,A,B,EPS,X,Y)
      GO TO (20,30,40)K
C  SET Y = Y(X)
  20  Y=X*DTAN(X)-1
      GO TO 10
C  WRITE X FOR NORMAL RETURN
  30  WRITE(6,35)X
  35  FORMAT(//'  RETURN FROM NB02, ROOT = ',D10.4)
      STOP
C  ERROR RETURN
  40  WRITE(6,45)
  45  FORMAT(' ERROR RETURN')
      STOP
      END
```

This produces the following output

```
 RETURN FROM NB02, ROOT = 0.8603D+00
```