1 SUMMARY

The subroutine finds a real zero of a continuous real function $y(x)$ in a given interval $a \le x \le b$.

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 2.2 so that for any x in $[a,b]$, the value of $y(x)$ is calculated and then control is passed back to `NB01A/AD`.

ATTRIBUTES — **Version:** 1.0.0. **Types:** `NB01A, NB01AD`. **Calls:** None. **Original date:** July 1985. 116 lines. **Origin:** M.J.D.Powell and S.Marlow, Harwell.

2 HOW TO USE THE PACKAGE

2.1 Argument list

The single precision version.

```
CALL NB01A(K,A,B,ERR,X,Y,MAXIT)
```

The double precision version.

```
CALL NB01AD(K,A,B,ERR,X,Y,MAXIT)
```

`K` is an `INTEGER` variable which must be set by the user to 0 on the first call to the subroutine. 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 no change in the sign of $y(x)$ has been found in the interval $a \le x \le b$,

4 for an error return when a change in the sign of $y(x)$ has been found, but the position of the root has not been calculated to the required accuracy, due to the limit imposed by |`MAXIT`|.

`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.

`ERR` is a `REAL` (`DOUBLE PRECISION` in the D version) variable which must be set by the user to specify a tolerance on the required value of $y(x)$. The subroutine accepts x as the required root when $|y(x)| \le$ `ERR`. If `ERR` is very small or zero, the subroutine calculates as accurate a root as the roundoff errors permit. `ERR` is not altered.

`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. On return with `K=4` the value of `X` is the best calculated estimate of the required root.

`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`).

`MAXIT` is an `INTEGER` which must be set by the user to limit the number of function evaluations requested by the subroutine and to control the printing. The limit on the number of requests for y(`X`) is set by `MAX(2,|MAXIT|)`. If `MAXIT`<0 the values of `X` and `Y` are printed, on the standard print unit, in the order in which they are calculated.

2.2 Calling sequence

The calling sequence should of the following form:

```
   K=0
10 CALL NB01A(K,A,B,ERR,X,Y,MAXIT)
   GO TO (20,30,40,50),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 no change in the sign of y(x) has been found in the interval A≤x≤B.
50 Instructions following an error return, when a change in the sign of y(x) has been found, but the position of the
   root has not been calculated to the required accuracy, due to the limit imposed by |MAXIT|.
```

3 GENERAL INFORMATION

Use of common: None.

Other routines called directly: None.

Input/output: Optional, depending on the sign on MAXIT.

4 METHOD

The method is composed of the following stages.

Firstly a search is made for a bracket, $[\alpha,\beta]$ such that $y(\alpha)$ and $y(\beta)$ have opposite signs. If $[a,b]$ is not a suitable bracket then a binary subdivision is made until either a bracket is found or the limit imposed by the value of MAXIT is reached.

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.

Note The first and second function values requested by the subroutine are $y(a)$ and $y(b)$. This information is given in case the calculation of $y(a)$ and/or $y(b)$ is easy, compared with the evaluation of $y(x)$ for a general value of x.

4.1 Accuracy

It is possible that the required accuracy cannot be achieved due to computer rounding errors. In this event a normal (K=2) return is made if it is known that the best calculable value of X has been found. The user can recognise this case by comparing the final value of Y with ERR.

5 EXAMPLE OF USE

As a simple example the following code calls NB01AD to find the zero of the function $y=x\tan(x)-1$ in the region $0\le x\le 1$.

```
      DOUBLE PRECISION A,B,ERR,X,Y
      DATA ERR/1.D-12/,A/0.D0/,B/1.D0/,MAXIT/15/
      K=0
  10  CALL NB01AD(K,A,B,EPS,X,Y,MAXIT)
      GO TO (20,30,40,50)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 NB01, ROOT = ',D10.4)
      STOP
C  ERROR RETURN
  40  WRITE(6,45)
  45  FORMAT(' ERROR RETURN')
C  ERROR RETURN WITH APPROXIMATE POSITION OF ROOT
  50  WRITE(6,55)X
  55  FORMAT(' ERROR RETURN, APPROXIMATE POSITION OF ROOT ',D10.4)
      STOP
      END
```

This produces the following output

```
  RETURN FROM NB01, ROOT = 0.8603D+00
```