1 SUMMARY

This subroutine **divides a polynomial by a linear factor to obtain the coefficients of the reduced polynomial,** i.e. given a polynomial of degree n

$$P(x) = a_1 + a_2 x + ... + a_{n+1} x^n$$

with real coefficients and given a real linear factor $(x-\xi)$, it calculates b_i i=1,2,...,n such that

$$P(x) \equiv (x-\xi)(b_1 + b_2 x + ... + b_n x^{n-1}) + r$$

The remainder r is assumed to be zero, i.e. ξ is assumed to be a close approximation to a root of $P(x)$. The method avoids magnifying inaccuracies in ξ during the calculation. Note that $b_n = a_{n+1}$.

ATTRIBUTES — Version: 1.0.0. **Types:** `PD04A`, `PD04AD`. **Original date:** May 1980. **Origin:** C.Birch*, Harwell.

2 HOW TO USE THE PACKAGE

2.1 The argument list and calling sequence

The single precision version

```
CALL PD04A(A,B,ROOT,N,NP1)
```

The double precision version

```
CALL PD04AD(A,B,ROOT,N,NP1)
```

`A` is a `REAL` (`DOUBLE PRECISION` in the D version) array which must be set by the user to contain the coefficients a_i i=1,2,...,n+1 of the original polynomial $P(x)$. The array length must be at least n+1 (see argument `NP1`).

`B` is a `REAL` (`DOUBLE PRECISION` in the D version) array which is set by the subroutine to contain b_i i=1,2,...,n the coefficients of the reduced polynomial. The length of the array must be at least n.

`ROOT` is a `REAL` (`DOUBLE PRECISION` in the D version) variable which must be set by the user to the value of the estimate of the root ξ.

`N` is an `INTEGER` variable which must be set by the user to n the degree of the polynomial $P(x)$.

`NP1` is an `INTEGER` variable which must be set by the user to the value n+1. It is used in the subroutine to dimension the array A.

3 GENERAL INFORMATION

Use of common: none.

Workspace: none.

Other routines called directly: none.

Input/output: none.

Restrictions:

$n > 0$,
`NP1` $= n+1$.

4 METHOD

The subroutine first finds k such that $|a_k \xi^{k-1}|$ takes its maximum value. Then it performs the deflation

$$b_n = a_{n+1},$$

$$b_i = \xi b_{i+1} + a_{i+1} \quad i=n-1,n-2,...,k$$

and

$$b_1 = -a_1/\xi,$$

$$b_i = (b_{i-1} - a_i)/\xi \quad i=2,3,...,k-1.$$

It has been shown by G.Peters and J.H.Wilkinson, J. Inst. Maths. Applics. 8 (1971), pp 21, that this method will always produce a reduced polynomial $B(x)$ such that $(x-\xi)B(x)$ differs little from the original polynomial $P(x)$. The code has been carefully designed to avoid any risk of overflow during the search for k.