NAG Fortran Library Routine Document

D05BWF

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details.

1 Purpose

D05BWF computes the quadrature weights associated with the Adams methods of orders three to six and the Backward Differentiation Formulae (BDF) methods of orders two to five. These rules, which are referred to as reducible quadrature rules, can then be used in the solution of Volterra integral and integro-differential equations.

2 Specification

```
SUBROUTINE D05BWF(METHOD, IORDER, OMEGA, NOMG, LENSW, SW, LDSW, NWT,
1                 IFAIL)
INTEGER           IORDER, NOMG, LENSW, LDSW, NWT, IFAIL
real              OMEGA(NOMG), SW(LDSW,NWT)
CHARACTER*1       METHOD
```

3 Description

D05BWF computes the weights $W_{n,j}$ and ω_i for a family of quadrature rules related to the Adams methods of orders three to six and the BDF methods of orders two to five, for approximating the integral:

$$\int_0^t \phi(s)ds \simeq h\sum_{j=0}^{p-1} W_{n,j}\phi(jh) + h\sum_{j=p}^{n} \omega_{n-j}\phi(jh), \quad 0 \leq t \leq T, \tag{1}$$

with $t = nh, (n \geq 0)$ for some given constant h.

In (1), h is a uniform mesh, p is related to the order of the method being used and $W_{n,j}$, ω_i are the starting and the convolution weights respectively. A description of how these weights can be used in the solution of a Volterra integral equation of the second kind is given in Section 8. For a general discussion of these methods, see Wolkenfelt (1982) for more details.

4 References

Lambert J D (1973) *Computational Methods in Ordinary Differential Equations* John Wiley

Wolkenfelt P H M (1982) The construction of reducible quadrature rules for Volterra integral and integro-differential equations *IMA J. Numer. Anal.* **2** 131–152

5 Parameters

1: METHOD – CHARACTER*1 *Input*

On entry: the type of method to be used.

For Adams type formulae set METHOD = 'A'.

For Backward Differentiation Formulae set METHOD = 'B'.

Constraint: METHOD = 'A' or 'B'.

2: IORDER – INTEGER *Input*

On entry: the order of the method to be used.

Constraints:

if METHOD = 'A', $3 \leq$ IORDER ≤ 6,
if METHOD = 'B', $2 \leq$ IORDER ≤ 5.

3: OMEGA(NOMG) – ***real*** array *Output*

On exit: contains the first NOMG convolution weights.

4: NOMG – INTEGER *Input*

On entry: the number of convolution weights.

Constraint: NOMG ≥ 1.

5: LENSW – INTEGER *Output*

On exit: the number of rows in the weights $W_{i,j}$.

6: SW(LDSW,NWT) – ***real*** array *Output*

On exit: SW$(i, j+1)$ contains the weights $W_{i,j}$, for $i = 1,2,...,$LENSW; $j = 0, 1, \ldots,$ NWT $- 1$.

7: LDSW – INTEGER *Input*

On entry: the first dimension of the array SW as declared in the (sub)program from which D05BWF is called.

Constraints:

if METHOD = 'A', LDSW $\geq$ NOMG + IORDER $- 2$,
if METHOD = 'B', LDSW $\geq$ NOMG + IORDER $- 1$.

8: NWT – INTEGER *Input*

On entry: the number of columns in the starting weights, p.

Constraints:

if METHOD = 'A', NWT = IORDER $- 1$,
if METHOD = 'B', NWT = IORDER.

9: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. Users who are unfamiliar with this parameter should refer to Chapter P01 for details.

On exit: IFAIL $= 0$ unless the routine detects an error (see Section 6).

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, for users not familiar with this parameter the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

6 Error Indicators and Warnings

If on entry IFAIL $= 0$ or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

IFAIL $= 1$

On entry, METHOD $\neq$ 'A' or 'B'.

IFAIL = 2

On entry, IORDER < 2 or IORDER > 6,
or NOMG < 1.

IFAIL = 3

On entry, METHOD = 'A' and IORDER = 2,
or METHOD = 'B' and IORDER = 6.

IFAIL = 4

On entry, METHOD = 'A' and NWT $\neq$ IORDER $- 1$,
or METHOD = 'B' and NWT $\neq$ IORDER.

IFAIL = 5

On entry, METHOD = 'A' and LDSW $<$ NOMG $+$ IORDER $- 2$,
or METHOD = 'B' and LDSW $<$ NOMG $+$ IORDER $- 1$.

7 Accuracy

None.

8 Further Comments

Reducible quadrature rules are most appropriate for solving Volterra integral equations (and integro-differential equations). In this section, we propose the following algorithm which the users may find useful in solving a linear Volterra integral equation of the form

$$y(t) = f(t) + \int_0^t K(t,s)y(s)\,ds, \quad 0 \le t \le T, \tag{2}$$

using D05BWF. In (2), $K(t,s)$ and $f(t)$ are given and the solution $y(t)$ is sought on a uniform mesh of size h such that $T = Nh$. Discretization of (2) yields

$$y_n = f(nh) + h\sum_{j=0}^{p-1} W_{n,j}K(nh,jh)y_j + h\sum_{j=p}^{n} \omega_{n-j}K(nh,jh)y_j, \tag{3}$$

where $y_n \simeq y(nh)$. We propose the following algorithm for computing y_n from (3) after a call to D05BWF:

(a) Equation (3) requires starting values, y_j, for $j = 1,2,\ldots,\text{NWT}-1$, with $y_0 = f(0)$. These starting values can be computed by solving the linear system

$$y_n = f(nh) + h\sum_{j=0}^{\text{NWT}-1} \text{SW}(n,j+1)K(nh,jh)y_j, \quad n = 1,2,\ldots,\text{NWT}-1.$$

(b) Compute the inhomogeneous terms

$$\sigma_n = f(nh) + h\sum_{j=0}^{\text{NWT}-1} \text{SW}(n,j+1)K(nh,jh)y_j, \quad n = \text{NWT},\text{NWT}+1,\ldots,N.$$

(c) Start the iteration for $n = \text{NWT},\text{NWT}+1,\ldots,N$ to compute y_n from:

$$(1 - h \times \text{OMEGA}(1)K(nh,nh))y_n = \sigma_n + h\sum_{j=\text{NWT}}^{n-1} \text{OMEGA}(n-j+1)K(nh,jh)y_j.$$

Note that for a nonlinear integral equation, the solution of a nonlinear algebraic system is required at step (a) and a single nonlinear equation at step (c).

9 Example

The following example generates the first ten convolution and thirteen starting weights generated by the fourth-order BDF method.

9.1 Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     D05BWF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      INTEGER          IORDER, NOMG, NWT, LDSW
      PARAMETER        (IORDER=4,NOMG=10,NWT=IORDER,LDSW=NOMG+IORDER-1)
*     .. Local Scalars ..
      INTEGER          IFAIL, J, LENSW, N
*     .. Local Arrays ..
      real             OMEGA(NOMG), SW(LDSW,NWT)
*     .. External Subroutines ..
      EXTERNAL         D05BWF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'D05BWF Example Program Results'
      IFAIL = 0
*
      CALL D05BWF('BDF',IORDER,OMEGA,NOMG,LENSW,SW,LDSW,NWT,IFAIL)
*
      WRITE (NOUT,*)
      WRITE (NOUT,*) 'The convolution weights'
      WRITE (NOUT,*)
*
      DO 20 N = 1, NOMG
         WRITE (NOUT,99999) N - 1, OMEGA(N)
   20 CONTINUE
*
      WRITE (NOUT,*)
      WRITE (NOUT,*) 'The weights W'
      WRITE (NOUT,*)
*
      DO 40 N = 1, LENSW
         WRITE (NOUT,99999) N, (SW(N,J),J=1,NWT)
   40 CONTINUE
*
      STOP
*
99999 FORMAT (1X,I3,4X,6F10.4)
      END
```

9.2 Program Data

None.

9.3 Program Results

```
 D05BWF Example Program Results

 The convolution weights

   0        0.4800
   1        0.9216
   2        1.0783
   3        1.0504
   4        0.9962
   5        0.9797
   6        0.9894
   7        1.0003
```

```
   8        1.0034
   9        1.0017

 The weights W

   1        0.3750    0.7917   -0.2083    0.0417
   2        0.3333    1.3333    0.3333    0.0000
   3        0.3750    1.1250    1.1250    0.3750
   4        0.4800    0.7467    1.5467    0.7467
   5        0.5499    0.5719    1.5879    0.8886
   6        0.5647    0.5829    1.5016    0.8709
   7        0.5545    0.6385    1.4514    0.8254
   8        0.5458    0.6629    1.4550    0.8098
   9        0.5449    0.6578    1.4741    0.8170
  10        0.5474    0.6471    1.4837    0.8262
  11        0.5491    0.6428    1.4831    0.8292
  12        0.5492    0.6438    1.4798    0.8279
  13        0.5488    0.6457    1.4783    0.8263
```