NAG Toolbox for MATLAB

d05bw

1 Purpose

d05bw 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 Syntax

```
[omega, lensw, sw, ifail] = d05bw(method, iorder, nomg, nwt)
```

3 Description

d05bw 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:

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

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

5.1 Compulsory Input Parameters

1: **method – string**

The type of method to be used.

method = 'A'

For Adams type formulae.

method = 'B'

For Backward Differentiation Formulae.

Constraint: **method** = 'A' or 'B'.

2: **iorder – int32 scalar**

The order of the method to be used.

Constraints:

if **method** = 'A', $3 \leq$ **iorder** ≤ 6;
if **method** = 'B', $2 \leq$ **iorder** ≤ 5.

3: **nomg – int32 scalar**

the number of convolution weights.

Constraint: **nomg** ≥ 1.

4: **nwt – int32 scalar**

p, the number of columns in the starting weights.

Constraints:

if **method** = 'A', **nwt** = **iorder** $- 1$;
if **method** = 'B', **nwt** = **iorder**.

5.2 Optional Input Parameters

None.

5.3 Input Parameters Omitted from the MATLAB Interface

ldsw

5.4 Output Parameters

1: **omega(nomg) – double array**

Contains the first **nomg** convolution weights.

2: **lensw – int32 scalar**

The number of rows in the weights $W_{i,j}$.

3: **sw(ldsw,nwt) – double array**

sw$(i, j+1)$ contains the weights $W_{i,j}$, for $i = 1, 2, \ldots,$ **lensw**; $j = 0, 1, \ldots,$ **nwt** $- 1$.

4: **ifail – int32 scalar**

0 unless the function detects an error (see Section 6).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

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** ≠ **iorder** − 1,
or **method** = 'B' and **nwt** ≠ **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 you may find useful in solving a linear Volterra integral equation of the form

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

using d05bw. 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

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

where $y_n \simeq y(nh)$. We propose the following algorithm for computing y_n from (3) after a call to d05bw:

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

$$y_n = f(nh) + h\sum_{j=0}^{\mathbf{nwt}-1} \mathbf{sw}(n,j+1)K(n,h,j,h)y_j, \qquad n = 1,2,\ldots,\mathbf{nwt}-1.$$

(b) Compute the inhomogeneous terms

$$\sigma_n = f(nh) + h\sum_{j=0}^{\mathbf{nwt}-1} \mathbf{sw}(n,j+1)K(n,h,j,h)y_j, \qquad n = \mathbf{nwt}, \mathbf{nwt}+1,\ldots,N.$$

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

$$(1 - h \times \mathbf{omega}(1)K(n,h,n,h))y_n = \sigma_n + h\sum_{j=\mathbf{nwt}}^{n-1} \mathbf{omega}(n-j+1)K(n,h,j,h)y_j.$$

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

9 Example

```
method = 'BDF';
iorder = int32(4);
nomg = int32(10);
nwt = int32(4);
[omega, lensw, sw, ifail] = d05bw(method, iorder, nomg, nwt)
```

```
omega =
```

```
    0.4800
    0.9216
    1.0783
    1.0504
    0.9962
    0.9797
    0.9894
    1.0003
    1.0034
    1.0017
lensw =
          13
sw =
    0.3750    0.7917   -0.2083    0.0417
    0.3333    1.3333    0.3333         0
    0.3750    1.1250    1.1250    0.3750
    0.4800    0.7467    1.5467    0.7467
    0.5499    0.5719    1.5879    0.8886
    0.5647    0.5829    1.5016    0.8709
    0.5545    0.6385    1.4514    0.8254
    0.5458    0.6629    1.4550    0.8098
    0.5449    0.6578    1.4741    0.8170
    0.5474    0.6471    1.4837    0.8262
    0.5491    0.6428    1.4831    0.8292
    0.5492    0.6438    1.4798    0.8279
    0.5488    0.6457    1.4783    0.8263
ifail =
           0
```