NAG Library Function Document

nag_inteq_abel_weak_weights (d05byc)

1 Purpose

nag_inteq_abel_weak_weights (d05byc) computes the fractional quadrature weights associated with the Backward Differentiation Formulae (BDF) of orders 4, 5 and 6. These weights can then be used in the solution of weakly singular equations of Abel type.

2 Specification

```
#include <nag.h>
#include <nagd05.h>

void nag_inteq_abel_weak_weights (Integer iorder, Integer iq,
     double omega[], double sw[], NagError *fail)
```

3 Description

nag_inteq_abel_weak_weights (d05byc) computes the weights $W_{i,j}$ and ω_i for a family of quadrature rules related to a BDF method for approximating the integral:

$$\frac{1}{\sqrt{\pi}}\int_0^t \frac{\phi(s)}{\sqrt{t-s}}ds \simeq \sqrt{h}\sum_{j=0}^{2p-2} W_{i,j}\phi(j\times h)+\sqrt{h}\sum_{j=2p-1}^{i}\omega_{i-j}\phi(j\times h), \quad 0\leq t\leq T, \tag{1}$$

with $t=i\times h(i\geq 0)$, for some given h. In (1), p is the order of the BDF method used and $W_{i,j}$, ω_i are the fractional starting and the fractional convolution weights respectively. The algorithm for the generation of ω_i is based on Newton's iteration. Fast Fourier transform (FFT) techniques are used for computing these weights and subsequently $W_{i,j}$ (see Baker and Derakhshan (1987) and Henrici (1979) for practical details and Lubich (1986) for theoretical details). Some special functions can be represented as the fractional integrals of simpler functions and fractional quadratures can be employed for their computation (see Lubich (1986)). A description of how these weights can be used in the solution of weakly singular equations of Abel type is given in Section 9.

4 References

Baker C T H and Derakhshan M S (1987) Computational approximations to some power series *Approximation Theory* (eds L Collatz, G Meinardus and G NÏrnberger) **81** 11–20

Henrici P (1979) Fast Fourier methods in computational complex analysis *SIAM Rev.* **21** 481–529

Lubich Ch (1986) Discretized fractional calculus *SIAM J. Math. Anal.* **17** 704–719

5 Arguments

1: **iorder** – Integer *Input*

On entry: p, the order of the BDF method to be used.

Constraint: $4\leq$ **iorder** ≤ 6.

2: **iq** – Integer *Input*

On entry: determines the number of weights to be computed. By setting **iq** to a value, $2^{\mathbf{iq}+1}$ fractional convolution weights are computed.

Constraint: **iq** ≥ 0.

3: **omega**$[2^{\mathbf{iq}+2}]$ – double *Output*

On exit: the first $2^{\mathbf{iq}+1}$ elements of **omega** contains the fractional convolution weights ω_i, for $i = 0, 1, \dots, 2^{\mathbf{iq}+1} - 1$. The remainder of the array is used as workspace.

4: **sw**$[N \times (\mathbf{2} \times \mathbf{iorder} - \mathbf{1})]$ – double *Output*

Note: the (i, j)th element of the matrix is stored in **sw**$[(j - 1) \times N + i - 1]$.

On exit: **sw**$[j \times N + i - 1]$ contains the fractional starting weights $W_{i,j}$, for $i = 1, 2, \dots, N$ and $j = 0, 1, \dots, 2 \times \mathbf{iorder} - 2$, where $N = \left(2^{\mathbf{iq}+1} + 2 \times \mathbf{iorder} - 1\right)$.

5: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 2.3.1.2 in How to Use the NAG Library and its Documentation for further information.

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

NE_INT

On entry, **iorder** = ⟨*value*⟩.
Constraint: $4 \leq \mathbf{iorder} \leq 6$.

On entry, **iq** = ⟨*value*⟩.
Constraint: $\mathbf{iq} \geq 0$.

NE_INTERNAL_ERROR

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please contact NAG for assistance.

An unexpected error has been triggered by this function. Please contact NAG.
See Section 2.7.6 in How to Use the NAG Library and its Documentation for further information.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 2.7.5 in How to Use the NAG Library and its Documentation for further information.

7 Accuracy

Not applicable.

8 Parallelism and Performance

nag_inteq_abel_weak_weights (d05byc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_inteq_abel_weak_weights (d05byc) makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the x06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this function. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

Fractional quadrature weights can be used for solving weakly singular integral equations of Abel type. In this section, we propose the following algorithm which you may find useful in solving a linear weakly singular integral equation of the form

$$y(t) = f(t) + \frac{1}{\sqrt{\pi}} \int_0^t \frac{K(t,s)y(s)}{\sqrt{t-s}} ds, \quad 0 \le t \le T, \tag{2}$$

using nag_inteq_abel_weak_weights (d05byc). 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 $T = N \times h$. Discretization of (2) yields

$$y_i = f(i \times h) + \sqrt{h} \sum_{j=0}^{2p-2} W_{i,j} K(i \times h, j \times h) y_j + \sqrt{h} \sum_{j=2p-1}^{i} \omega_{i-j} K(i \times h, j \times h) y_j, \tag{3}$$

where $y_i \simeq y(i \times h)$, for $i = 1, 2, \ldots, N$. We propose the following algorithm for computing y_i from (3) after a call to nag_inteq_abel_weak_weights (d05byc):

(a) Set $N = 2^{\mathbf{iq}+1} + 2 \times \mathbf{iorder} - 2$ and $h = T/N$.

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

$$y_i = f(i \times h) + \sqrt{h} \sum_{j=0}^{2 \times \mathbf{iorder} - 2} \mathbf{sw}[j \times N + i] K(i \times h, j \times h) y_j, \quad i = 1, 2, \ldots, 2 \times \mathbf{iorder} - 2.$$

(c) Compute the inhomogeneous terms

$$\sigma_i = f(i \times h) + \sqrt{h} \sum_{j=0}^{2 \times \mathbf{iorder} - 2} \mathbf{sw}[j \times N + i] K(i \times h, j \times h) y_j, \quad i = 2 \times \mathbf{iorder} - 1, 2 \times \mathbf{iorder}, \ldots, N.$$

(d) Start the iteration for $i = 2 \times \mathbf{iorder} - 1, 2 \times \mathbf{iorder}, \ldots, N$ to compute y_i from:

$$\left(1 - \sqrt{h}\,\mathbf{omega}[0] K(i \times h, i \times h)\right) y_i = \sigma_i + \sqrt{h} \sum_{j=2 \times \mathbf{iorder} - 1}^{i-1} \mathbf{omega}[i-j] K(i \times h, j \times h) y_j.$$

Note that for nonlinear weakly singular equations, the solution of a nonlinear algebraic system is required at step (b) and a single nonlinear equation at step (d).

10 Example

The following example generates the first 16 fractional convolution and 23 fractional starting weights generated by the fourth-order BDF method.

10.1 Program Text

```
/* nag_inteq_abel_weak_weights (d05byc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagd05.h>
```

```
int main(void)
{
  /* Scalars */
  Integer exit_status = 0;
  Integer i, iorder, iq, j, lomega, n, ncols, ncwt, nmax;
  /* Arrays */
  double *omega = 0, *sw = 0;
  /* NAG types */
  NagError fail;

  INIT_FAIL(fail);

  printf("nag_inteq_abel_weak_weights (d05byc) Example Program Results\n");

  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

#ifdef _WIN32
  scanf_s("%" NAG_IFMT "%*[^\n] ", &iorder);
#else
  scanf("%" NAG_IFMT "%*[^\n] ", &iorder);
#endif
#ifdef _WIN32
  scanf_s("%" NAG_IFMT "%*[^\n] ", &iq);
#else
  scanf("%" NAG_IFMT "%*[^\n] ", &iq);
#endif

  ncwt = pow(2, iq + 1);
  lomega = 2 * ncwt;
  ncols = 2 * iorder - 1;
  nmax = ncwt + ncols;

  if (!(omega = NAG_ALLOC(lomega, double)) ||
      !(sw = NAG_ALLOC(ncols * nmax, double))
         )
  {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }

  /*
     nag_inteq_abel_weak_weights (d05byc).
     Generate weights for use in solving weakly singular Abel-type equations.
   */
  nag_inteq_abel_weak_weights(iorder, iq, omega, sw, &fail);

  if (fail.code != NE_NOERROR) {
    printf("Error from nag_inteq_abel_weak_weights (d05byc).\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
  }

  printf("\nFractional convolution weights\n\n");
  for (i = 0; i < ncwt; i++)
    printf("%5" NAG_IFMT " %9.4f\n", i, omega[i]);

  printf("\nFractional starting weights W\n\n");

#define SW(I, J) sw[J * nmax + I - 1]

  for (n = 1; n <= nmax; n++) {
    printf("%5" NAG_IFMT " ", n);
    for (j = 0; j < ncols; j++)
      printf("%9.4f", SW(n, j));
```

```
    printf("\n");
  }

#undef SW

END:

  NAG_FREE(sw);
  NAG_FREE(omega);

  return exit_status;
}
```

10.2 Program Data

None.

10.3 Program Results

```
nag_inteq_abel_weak_weights (d05byc) Example Program Results

Fractional convolution weights

    0    0.6928
    1    0.6651
    2    0.4589
    3    0.3175
    4    0.2622
    5    0.2451
    6    0.2323
    7    0.2164
    8    0.2006
    9    0.1878
   10    0.1780
   11    0.1700
   12    0.1629
   13    0.1566
   14    0.1508
   15    0.1457

Fractional starting weights W

    1    0.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000
    2    0.0565   2.8928  -6.7497  11.6491 -11.1355   5.5374  -1.1223
    3    0.0371   1.7401  -2.8628   6.5207  -6.4058   3.2249  -0.6583
    4    0.0300   1.3207  -2.4642   6.3612  -5.4478   2.7025  -0.5481
    5    0.0258   1.1217  -2.2620   5.3683  -3.7553   2.2132  -0.4549
    6    0.0230   0.9862  -2.0034   4.5005  -3.2772   2.7262  -0.4320
    7    0.0208   0.9001  -1.8989   4.2847  -3.5881   2.8201   0.2253
    8    0.0190   0.8506  -1.9250   4.4164  -4.0181   2.7932   0.1564
    9    0.0173   0.8177  -1.9697   4.5348  -4.2425   2.7458  -0.0697
   10    0.0160   0.7886  -1.9781   4.5318  -4.2769   2.6997  -0.2127
   11    0.0149   0.7603  -1.9548   4.4545  -4.2332   2.6541  -0.2620
   12    0.0140   0.7338  -1.9198   4.3619  -4.1782   2.6059  -0.2716
   13    0.0132   0.7097  -1.8842   4.2754  -4.1246   2.5544  -0.2767
   14    0.0125   0.6880  -1.8497   4.1933  -4.0662   2.5011  -0.2845
   15    0.0119   0.6681  -1.8153   4.1109  -4.0004   2.4479  -0.2915
   16    0.0114   0.6497  -1.7805   4.0279  -3.9304   2.3962  -0.2951
   17    0.0110   0.6327  -1.7461   3.9463  -3.8598   2.3466  -0.2958
   18    0.0105   0.6168  -1.7126   3.8677  -3.7907   2.2990  -0.2950
   19    0.0102   0.6020  -1.6804   3.7926  -3.7238   2.2536  -0.2935
   20    0.0098   0.5882  -1.6495   3.7209  -3.6589   2.2101  -0.2917
   21    0.0095   0.5752  -1.6199   3.6523  -3.5961   2.1686  -0.2895
   22    0.0093   0.5631  -1.5916   3.5867  -3.5356   2.1291  -0.2871
   23    0.0090   0.5517  -1.5644   3.5240  -3.4774   2.0914  -0.2844
```