NAG Library Function Document

nag_ode_bvp_ps_lin_quad_weights (d02uyc)

1 Purpose

nag_ode_bvp_ps_lin_quad_weights (d02uyc) obtains the weights for Clenshaw–Curtis quadrature at Chebyshev points. This allows for fast approximations of integrals for functions specified on Chebyshev Gauss–Lobatto points on $[-1, 1]$.

2 Specification

```
#include <nag.h>
#include <nagd02.h>
void nag_ode_bvp_ps_lin_quad_weights (Integer n, double w[], NagError *fail)
```

3 Description

nag_ode_bvp_ps_lin_quad_weights (d02uyc) obtains the weights for Clenshaw–Curtis quadrature at Chebyshev points.

Given the (Clenshaw–Curtis) weights w_i, for $i = 0, 1, \ldots, n$, and function values $f_i = f(t_i)$ (where $t_i = -\cos(i \times \pi/n)$, for $i = 0, 1, \ldots, n$, are the Chebyshev Gauss–Lobatto points), then $\int_{-1}^{1} f(x)dx \approx \sum_{i=0}^{n} w_i f_i$.

For a function discretized on a Chebyshev Gauss–Lobatto grid on $[a, b]$ the resultant summation must be multiplied by the factor $(b - a)/2$.

4 References

Trefethen L N (2000) *Spectral Methods in MATLAB* SIAM

5 Arguments

1: **n** – Integer *Input*

On entry: n, where the number of grid points is $n + 1$.

Constraint: **n** > 0 and **n** is even.

2: **w**[**n** + **1**] – double *Output*

On exit: the Clenshaw–Curtis quadrature weights, w_i, for $i = 0, 1, \ldots, n$.

3: **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 3.2.1.2 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, $\mathbf{n} = \langle value \rangle$.
Constraint: $\mathbf{n} > 0$.

On entry, $\mathbf{n} = \langle value \rangle$.
Constraint: **n** is even.

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 3.6.6 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 3.6.5 in How to Use the NAG Library and its Documentation for further information.

7 Accuracy

The accuracy should be close to ***machine precision***.

8 Parallelism and Performance

nag_ode_bvp_ps_lin_quad_weights (d02uyc) is not threaded in any implementation.

9 Further Comments

A real array of length $2n$ is internally allocated.

10 Example

This example approximates the integral $\int_{-1}^{3} 3x^2 dx$ using 65 Clenshaw–Curtis weights and a 65-point Chebyshev Gauss–Lobatto grid on $[-1, 3]$.

10.1 Program Text

```
/* nag_ode_bvp_ps_lin_quad_weights (d02uyc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagd02.h>
#include <nagx02.h>

#ifdef __cplusplus
extern "C"
{
#endif
  static double NAG_CALL exact(double x);
```

```
#ifdef __cplusplus
}
#endif

int main(void)
{
  /*  Scalars */
  Integer exit_status = 0;
  Integer i, n;
  double a = -1.0, b = 3.0;
  double integ, scale, uerr;
  double teneps = 10.0 * nag_machine_precision;
  /*  Arrays */
  double *f = 0, *w = 0, *x = 0;
  /* NAG types */
  Nag_Boolean reqerr = Nag_FALSE, reqwgt = Nag_FALSE;
  NagError fail;

  INIT_FAIL(fail);

  printf("nag_ode_bvp_ps_lin_quad_weights (d02uyc) "
         "Example Program Results\n\n");

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

#ifdef _WIN32
  scanf_s("%" NAG_IFMT "%*[^\n] ", &n);
#else
  scanf("%" NAG_IFMT "%*[^\n] ", &n);
#endif
  if (!(f = NAG_ALLOC((n + 1), double)) ||
      !(w = NAG_ALLOC((n + 1), double)) || !(x = NAG_ALLOC((n + 1), double))
        )
  {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }

  /* Set up solution grid:
   * nag_ode_bvp_ps_lin_cgl_grid (d02ucc).
   * Chebyshev Gauss-Lobatto grid generation.
   */
  nag_ode_bvp_ps_lin_cgl_grid(n, a, b, x, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_ode_bvp_ps_lin_cgl_grid (d02ucc).\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
  }

  /* Set up problem right hand sides for grid. */
  for (i = 0; i < n + 1; i++)
    f[i] = exact(x[i]);

  scale = 0.5 * (b - a);

  /* Solve on equally spaced grid:
   * nag_ode_bvp_ps_lin_quad_weights (d02uyc).
   * Clenshaw-Curtis quadrature weights for integration using computed
   * Chebyshev coefficients.
   */
  nag_ode_bvp_ps_lin_quad_weights(n, w, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_ode_bvp_ps_lin_quad_weights (d02uyc).\n%s\n",
           fail.message);
```

```
    exit_status = 1;
    goto END;
  }

  /* Apply the weights, w, to the function values, f, and scale. */
  integ = 0.0;
  for (i = 0; i < n + 1; i++)
    integ = integ + w[i] * f[i];
  integ = scale * integ;

  /* Print function values and weights if required. */
  if (reqwgt) {
    printf("f(x) and integral weights\n\n");
    printf("       x         f(x)        w\n");
    for (i = 0; i < n + 1; i++) {
      printf("%10.4f %10.4f %10.4f\n", x[i], f[i], w[i]);
    }
    printf("\n");
  }

  /* Print approximation to integral. */
  printf("Integral of f(x) from %6.1f to %6.1f = %13.5f\n", a, b, integ);
  if (reqerr) {
    uerr = fabs(integ - 28.0);
    printf("Integral is within a multiple ");
    printf("%8" NAG_IFMT " ", 10 * ((Integer) (uerr / teneps) + 1));
    printf(" of machine precision.\n");
  }
END:
  NAG_FREE(f);
  NAG_FREE(w);
  NAG_FREE(x);
  return exit_status;
}

static double NAG_CALL exact(double x)
{
  return 3.0 * pow(x, 2);
}
```

10.2 Program Data

```
nag_ode_bvp_ps_lin_quad_weights (d02uyc) Example Program Data
  64           : n
```

10.3 Program Results

```
nag_ode_bvp_ps_lin_quad_weights (d02uyc) Example Program Results

Integral of f(x) from   -1.0 to    3.0 =      28.00000
```