NAG Library Function Document

nag_ode_bvp_ps_lin_cheb_eval (d02uzc)

1 Purpose

nag_ode_bvp_ps_lin_cheb_eval (d02uzc) returns the value of the kth Chebyshev polynomial evaluated at a point $x \in [-1, 1]$. nag_ode_bvp_ps_lin_cheb_eval (d02uzc) is primarily a utility function for use by the Chebyshev boundary value problem solvers.

2 Specification

```
#include <nag.h>
#include <nagd02.h>
```

```
void nag_ode_bvp_ps_lin_cheb_eval (Integer k, double x, double *t,
     NagError *fail)
```

3 Description

nag_ode_bvp_ps_lin_cheb_eval (d02uzc) returns the value, T, of the kth Chebyshev polynomial evaluated at a point $x \in [-1, 1]$; that is, $T = \cos(k \times \arccos(x))$.

4 References

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

5 Arguments

1: **k** – Integer *Input*

On entry: the order of the Chebyshev polynomial.

Constraint: $\mathbf{k} \geq 0$.

2: **x** – double *Input*

On entry: the point at which to evaluate the polynomial.

Constraint: $-1.0 \leq \mathbf{x} \leq 1.0$.

3: **t** – double * *Output*

On exit: the value, T, of the Chebyshev polynomial order k evaluated at x.

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

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_BAD_PARAM

On entry, argument $\langle$*value*$\rangle$ had an illegal value.

NE_INT

On entry, $\mathbf{k} = \langle$*value*$\rangle$.
Constraint: $\mathbf{k} \geq 0$.

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.

NE_REAL

On entry, $\mathbf{x} = \langle value \rangle$.
Constraint: $-1.0 \leq \mathbf{x} \leq 1.0$.

7 Accuracy

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

8 Parallelism and Performance

Not applicable.

9 Further Comments

None.

10 Example

A set of Chebyshev coefficients is obtained for the function $x + \exp(-x)$ defined on $[-0.24 \times \pi, 0.5 \times \pi]$ using nag_ode_bvp_ps_lin_cgl_grid (d02ucc). At each of a set of new grid points in the domain of the function nag_ode_bvp_ps_lin_cheb_eval (d02uzc) is used to evaluate each Chebshev polynomial in the series representation. The values obtained are multiplied to the Chebyshev coefficients and summed to obtain approximations to the given function at the new grid points.

10.1 Program Text

```
/* nag_ode_bvp_ps_lin_cheb_eval (d02uzc) Example Program.
 *
 * Copyright 2011, Numerical Algorithms Group.
 *
 * Mark 23, 2011.
 */

#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagd02.h>
#include <nagx01.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, k, m, n;
  double       a = -0.24 * nag_pi, b = 0.5 * nag_pi;
  double       deven, dmap, fseries, t, uerr, xeven, xmap;
  double       teneps = 10.0 * nag_machine_precision;
  /*  Arrays */
  double      *c = 0, *f = 0, *x = 0;
```

```
  /* NAG types */
  Nag_Boolean reqerr = Nag_FALSE;
  NagError    fail;

  INIT_FAIL(fail);

  printf("nag_ode_bvp_ps_lin_cheb_eval (d02uzc) Example Program Results \n\n");

  /* Skip heading in data file */
  scanf("%*[^\n] ");
  scanf("%"NAG_IFMT "", &n);
  scanf("%"NAG_IFMT "", &m);
  if (
    !(f = NAG_ALLOC((n + 1), double)) ||
    !(c = NAG_ALLOC((n + 1), double)) ||
    !(x = NAG_ALLOC((n + 1), double))
    )
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  /* Set up Chebyshev 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;
  }

  /* Evaluate function on grid and get interpolating Chebyshev coefficients. */
  for (i = 0; i < n + 1; i++) f[i] = exact(x[i]);

  /* nag_ode_bvp_ps_lin_coeffs (d02uac).
   * Coefficients of Chebyshev interpolating polynomial
   * from function values on Chebyshev grid.
   */
  nag_ode_bvp_ps_lin_coeffs(n, f, c, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_ode_bvp_ps_lin_coeffs (d02uac).\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
  }

  /* Evaluate Chebyshev series manually by evaluating each Chebyshev
   * polynomial in turn at new equispaced (m+1) grid points.
   * Chebyshev series on [-1,1] map of [a,b].
  */
  xmap = -1.0;
  dmap = 2.0/(double) (m - 1);
  xeven = a;
  deven = (b - a)/(double) (m - 1);
  printf("    x_even      x_map      Sum\n");
  uerr = 0.0;
  for (i = 0; i < m; i++) {
    fseries = 0.0;
    for (k = 0; k < n + 1; k++) {
      /* nag_ode_bvp_ps_lin_cheb_eval (d02uzc).
       * Chebyshev polynomial evaluation, T_k(x).
       */
      nag_ode_bvp_ps_lin_cheb_eval(k, xmap, &t, &fail);
      if (fail.code != NE_NOERROR) {
        printf("Error from nag_ode_bvp_ps_lin_cheb_eval (d02uzc).\n%s\n",
               fail.message);
        exit_status = 1;
```

```
        goto END;
      }

      fseries = fseries + c[k] * t;
    }
    uerr = MAX(uerr, fabs(fseries - exact(xeven)));
    printf("%10.4f %10.4f %10.4f \n", xeven, xmap, fseries);
    xmap = MIN(1.0, xmap + dmap);
    xeven = xeven + deven;
  }

  if (reqerr) {
    printf("\nError in coefficient sum is < ");
    printf("%8"NAG_IFMT " ", 10 * ((Integer) (uerr/teneps) + 1));
    printf(" * machine precision.\n");
  }
 END:
  NAG_FREE(c);
  NAG_FREE(f);
  NAG_FREE(x);
  return exit_status;
}

static double NAG_CALL exact(double x)
{
  return x + exp(-x);
}
```

10.2 Program Data

```
nag_ode_bvp_ps_lin_cheb_eval (d02uzc) Example Program Data
  16 9              : n, m
```

10.3 Program Results

```
nag_ode_bvp_ps_lin_cheb_eval (d02uzc) Example Program Results

    x_even      x_map       Sum
   -0.7540    -1.0000     1.3715
   -0.4634    -0.7500     1.1261
   -0.1728    -0.5000     1.0158
    0.1178    -0.2500     1.0067
    0.4084     0.0000     1.0731
    0.6990     0.2500     1.1961
    0.9896     0.5000     1.3613
    1.2802     0.7500     1.5582
    1.5708     1.0000     1.7787
```