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 \( k \)th 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

```c
#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 \( k \)th Chebyshev polynomial evaluated at a point \( x \in [-1, 1] \); that is, \( T = \cos(k \times \arccos(x)) \).

4 References


5 Arguments

1: \( k \) – Integer \( \quad \text{Input} \)

   \( \text{On entry:} \) the order of the Chebyshev polynomial.

   \( \text{Constraint:} \ k \geq 0. \)

2: \( x \) – double \( \quad \text{Input} \)

   \( \text{On entry:} \) the point at which to evaluate the polynomial.

   \( \text{Constraint:} \ -1.0 \leq x \leq 1.0. \)

3: \( t \) – double * \( \quad \text{Output} \)

   \( \text{On exit:} \) the value, \( T \), of the Chebyshev polynomial order \( k \) evaluated at \( x \).

4: \( \text{fail} \) – NagError * \( \quad \text{Input/Output} \)

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

6 Error Indicators and Warnings

**NE_ALLOC_FAIL**

Dynamic memory allocation failed.

See Section 3.2.1.2 in the Essential Introduction for further information.

**NE_BAD_PARAM**

On entry, argument \( \langle \text{value} \rangle \) had an illegal value.
NE_INT
  On entry, k = \langle value\rangle.
  Constraint: 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.

  An unexpected error has been triggered by this function. Please contact NAG.
  See Section 3.6.6 in the Essential Introduction 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 the Essential Introduction for further information.

NE_REAL
  On entry, x = \langle value\rangle.
  Constraint: -1.0 \leq 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
Chebyshev 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 2014 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
} /* end of extern "C" block */
#endif

NAG Library Manual
```c
#endif
#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 */
    #ifdef _WIN32
        scanf_s("%*\[\n");
    #else
        scanf("%*\[\n");
    #endif
    #ifdef _WIN32
        scanf_s("%"NAG_IFMT ", &n);
    #else
        scanf("%"NAG_IFMT ", &n);
    #endif
    #ifdef _WIN32
        scanf_s("%"NAG_IFMT ", &m);
    #else
        scanf("%"NAG_IFMT ", &m);
    #endif
    #ifdef _WIN32
        scanf_s("%"NAG_IFMT "", &n);
    #else
        scanf("%"NAG_IFMT "", &n);
    #endif
    #ifdef _WIN32
        scanf_s("%"NAG_IFMT "", &m);
    #else
        scanf("%"NAG_IFMT "", &m);
    #endif
    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

<table>
<thead>
<tr>
<th>x_even</th>
<th>x_map</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.7540</td>
<td>-1.0000</td>
<td>1.3715</td>
</tr>
<tr>
<td>-0.4634</td>
<td>-0.7500</td>
<td>1.1261</td>
</tr>
<tr>
<td>-0.1728</td>
<td>-0.5000</td>
<td>1.0158</td>
</tr>
<tr>
<td>0.1178</td>
<td>-0.2500</td>
<td>1.0067</td>
</tr>
<tr>
<td>Value</td>
<td>0.4084</td>
<td>0.6990</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>Column 1</td>
<td>0.0000</td>
<td>0.2500</td>
</tr>
<tr>
<td>Column 2</td>
<td>1.0731</td>
<td>1.1961</td>
</tr>
</tbody>
</table>