NAG Library Function Document

nag_1d_cheb_interp_fit (e02afc)

1 Purpose

nag_1d_cheb_interp_fit (e02afc) computes the coefficients of a polynomial, in its Chebyshev series form, which interpolates (passes exactly through) data at a special set of points. Least squares polynomial approximations can also be obtained.

2 Specification

```
#include <nag.h>
#include <nage02.h>
```

```
void nag_1d_cheb_interp_fit (Integer nplus1, const double f[], double a[],
     NagError *fail)
```

3 Description

nag_1d_cheb_interp_fit (e02afc) computes the coefficients a_j, for $j = 1, 2, \ldots, n+1$, in the Chebyshev series

$$\frac{1}{2}a_1T_0(\bar{x}) + a_2T_1(\bar{x}) + a_3T_2(\bar{x}) + \cdots + a_{n+1}T_n(\bar{x}),$$

which interpolates the data f_r at the points

$$\bar{x}_r = \cos((r-1)\pi/n), r = 1, 2, \ldots, n+1.$$

Here $T_j(\bar{x})$ denotes the Chebyshev polynomial of the first kind of degree j with argument $\bar{x}$. The use of these points minimizes the risk of unwanted fluctuations in the polynomial and is recommended when you can choose the data abscissae, e.g., when the data is given as a graph. For further advantages of this choice of points, see Clenshaw (1962).

In terms of your original variables, x say, the values of x at which the data f_r are to be provided are

$$x_r = \frac{1}{2}(x_{\max} - x_{\min})\cos((r-1)\pi/n) + \frac{1}{2}(x_{\max} + x_{\min}), \quad r = 1, 2, \ldots, n+1$$

where $x_{\max}$ and $x_{\min}$ are respectively the upper and lower ends of the range of x over which you wish to interpolate.

Truncation of the resulting series after the term involving a_{i+1}, say, yields a least squares approximation to the data. This approximation, $p(\bar{x})$, say, is the polynomial of degree i which minimizes

$$\frac{1}{2}\epsilon_1^2 + \epsilon_2^2 + \epsilon_3^2 + \cdots + \epsilon_n^2 + \frac{1}{2}\epsilon_{n+1}^2,$$

where the residual $\epsilon_r = p(\bar{x}_r) - f_r$, for $r = 1, 2, \ldots, n+1$.

The method employed is based on the application of the three-term recurrence relation due to Clenshaw (1955) for the evaluation of the defining expression for the Chebyshev coefficients (see, for example, Clenshaw (1962)). The modifications to this recurrence relation suggested by Reinsch and Gentleman (see Gentleman (1969)) are used to give greater numerical stability.

For further details of the algorithm and its use see Cox (1974), Cox and Hayes (1973).

Subsequent evaluation of the computed polynomial, perhaps truncated after an appropriate number of terms, should be carried out using nag_1d_cheb_eval (e02aec).

4 References

Clenshaw C W (1955) A note on the summation of Chebyshev series *Math. Tables Aids Comput.* **9** 118–120

Clenshaw C W (1962) Chebyshev Series for Mathematical Functions *Mathematical tables* HMSO

Cox M G (1974) A data-fitting package for the non-specialist user *Software for Numerical Mathematics* (ed D J Evans) Academic Press

Cox M G and Hayes J G (1973) Curve fitting: a guide and suite of algorithms for the non-specialist user *NPL Report NAC26* National Physical Laboratory

Gentleman W M (1969) An error analysis of Goertzel's (Watt's) method for computing Fourier coefficients *Comput. J.* **12** 160–165

5 Arguments

1: **nplus1** – Integer *Input*

On entry: the number $n+1$ of data points (one greater than the degree n of the interpolating polynomial).

Constraint: **nplus1** ≥ 2.

2: **f**[**nplus1**] – const double *Input*

On entry: for $r = 1, 2, \ldots, n+1$, **f**$[r-1]$ must contain f_r the value of the dependent variable (ordinate) corresponding to the value

$$\bar{x}_r = \cos\left(\frac{\pi(r-1)}{n}\right)$$

of the independent variable (abscissa) $\bar{x}$, or equivalently to the value

$$x_r = \frac{1}{2}(x_{\text{max}} - x_{\text{min}}) \cos(\pi(r-1)/n) + \frac{1}{2}(x_{\text{max}} + x_{\text{min}})$$

of your original variable x. Here x_{max} and x_{min} are respectively the upper and lower ends of the range over which you wish to interpolate.

3: **a**[**nplus1**] – double *Output*

On exit: **a**$[j-1]$ is the coefficient a_j in the interpolating polynomial, for $j = 1, 2, \ldots, n+1$.

4: **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_INT_ARG_LT

On entry, **nplus1** must not be less than 2: **nplus1** $= \langle value \rangle$.

7 Accuracy

The rounding errors committed are such that the computed coefficients are exact for a slightly perturbed set of ordinates $f_r + \delta f_r$. The ratio of the sum of the absolute values of the δf_r to the sum of the absolute values of the f_r is less than a small multiple of $(n+1)\epsilon$, where ϵ is the ***machine precision***.

8 Parallelism and Performance

nag_1d_cheb_interp_fit (e02afc) is not threaded in any implementation.

9 Further Comments

The time taken by nag_1d_cheb_interp_fit (e02afc) is approximately proportional to $(n+1)^2 + 30$.

For choice of degree when using the function for least squares approximation, see the e02 Chapter Introduction.

10 Example

Determine the Chebyshev coefficients of the polynomial which interpolates the data $\bar{x}_r, f_r$, for $r = 1, 2, \ldots, 11$, where $\bar{x}_r = \cos((r-1)\pi/10)$ and $f_r = e^{\bar{x}_r}$. Evaluate, for comparison with the values of f_r, the resulting Chebyshev series at $\bar{x}_r$, for $r = 1, 2, \ldots, 11$.

The example program supplied is written in a general form that will enable polynomial interpolations of arbitrary data at the cosine points $\cos((r-1)\pi/n)$, for $r = 1, 2, \ldots, n+1$ to be obtained for any n ($= \mathbf{nplus1} - 1$). Note that nag_1d_cheb_eval (e02aec) is used to evaluate the interpolating polynomial. The program is self-starting in that any number of datasets can be supplied.

10.1 Program Text

```
/* nag_1d_cheb_interp_fit (e02afc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 *
 */

#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nage02.h>
#include <nagx01.h>
#include <math.h>

int main(void)
{
  Integer exit_status = 0;
  double *an = 0, d1, *f = 0, fit, pi, piby2n, *xcap = 0;

  Integer i, j, n;
  Integer r;
  NagError fail;

  INIT_FAIL(fail);

  printf("nag_1d_cheb_interp_fit (e02afc) Example Program Results \n");

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

  /* nag_pi (x01aac).
   * pi
   */
  pi = nag_pi;
#ifdef _WIN32
```

```
  while ((scanf_s("%" NAG_IFMT "", &n)) != EOF)
#else
  while ((scanf("%" NAG_IFMT "", &n)) != EOF)
#endif
  {
    if (n > 0) {
      if (!(an = NAG_ALLOC(n + 1, double)) ||
          !(f = NAG_ALLOC(n + 1, double)) ||
          !(xcap = NAG_ALLOC(n + 1, double)))
      {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
      }
    }
    else {
      printf("Invalid n.\n");
      exit_status = 1;
      return exit_status;
    }

    piby2n = pi * 0.5 / (double) n;
    for (r = 0; r < n + 1; ++r)
#ifdef _WIN32
      scanf_s("%lf", &f[r]);
#else
      scanf("%lf", &f[r]);
#endif

    for (r = 0; r < n + 1; ++r) {
      i = r;
      /* The following method of evaluating  xcap = cos(pi*i/n)
       *  ensures that the computed value has a small relative error
       *  and, moreover, is bounded in modulus by unity for all
       *  i = 0, 1, ..., n.  (It is assumed that the sine routine
       *  produces a result with a small relative error for values
       *  of the argument between  -PI/4  and  PI/4).
       */
      if (2 * i <= n) {
        d1 = sin(piby2n * i);
        xcap[i] = 1.0 - d1 * d1 * 2.0;
      }
      else if (2 * i > n * 3) {
        d1 = sin(piby2n * (n - i));
        xcap[i] = d1 * d1 * 2.0 - 1.0;
      }
      else {
        xcap[i] = sin(piby2n * (n - 2 * i));
      }
    }
    /* nag_1d_cheb_interp_fit (e02afc).
     * Computes the coefficients of a Chebyshev series
     * polynomial for interpolated data
     */
    nag_1d_cheb_interp_fit(n + 1, f, an, &fail);
    if (fail.code != NE_NOERROR) {
      printf("Error from nag_1d_cheb_interp_fit (e02afc).\n%s\n",
             fail.message);
      exit_status = 1;
      goto END;
    }

    printf("\n          Chebyshev \n");
    printf("   J   coefficient A(J) \n");
    for (j = 0; j < n + 1; ++j)
      printf(" %3" NAG_IFMT "%14.7f\n", j + 1, an[j]);
    printf("\n   R    Abscissa   Ordinate      Fit \n");
    for (r = 0; r < n + 1; ++r) {
      /* nag_1d_cheb_eval (e02aec).
       * Evaluates the coefficients of a Chebyshev series
       * polynomial
```

```
      */
      nag_1d_cheb_eval(n + 1, an, xcap[r], &fit, &fail);
      if (fail.code != NE_NOERROR) {
        printf("Error from nag_1d_cheb_eval (e02aec).\n%s\n", fail.message);
        exit_status = 1;
        goto END;
      }

      printf(" %3" NAG_IFMT "%11.4f%11.4f%11.4f\n", r + 1, xcap[r], f[r],
             fit);
    }
  END:
    NAG_FREE(an);
    NAG_FREE(f);
    NAG_FREE(xcap);
  }
  return exit_status;
}
```

10.2 Program Data

```
nag_1d_cheb_interp_fit (e02afc) Example Program Data
  10
     2.7182
     2.5884
     2.2456
     1.7999
     1.3620
     1.0000
     0.7341
     0.5555
     0.4452
     0.3863
     0.3678
```

10.3 Program Results

```
nag_1d_cheb_interp_fit (e02afc) Example Program Results

          Chebyshev
   J   coefficient A(J)
   1     2.5320000
   2     1.1303095
   3     0.2714893
   4     0.0443462
   5     0.0055004
   6     0.0005400
   7     0.0000307
   8    -0.0000006
   9    -0.0000004
  10     0.0000049
  11    -0.0000200

   R    Abscissa   Ordinate      Fit
   1     1.0000     2.7182     2.7182
   2     0.9511     2.5884     2.5884
   3     0.8090     2.2456     2.2456
   4     0.5878     1.7999     1.7999
   5     0.3090     1.3620     1.3620
   6     0.0000     1.0000     1.0000
   7    -0.3090     0.7341     0.7341
   8    -0.5878     0.5555     0.5555
   9    -0.8090     0.4452     0.4452
  10    -0.9511     0.3863     0.3863
  11    -1.0000     0.3678     0.3678
```