NAG Library Function Document

nag_1d_minimax_polynomial (e02alc)

1 Purpose

nag_1d_minimax_polynomial (e02alc) calculates a minimax polynomial fit to a set of data points.

2 Specification

```
#include <nag.h>
#include <nage02.h>
void nag_1d_minimax_polynomial (Integer n, const double x[],
     const double y[], Integer m, double a[], double *ref, NagError *fail)
```

3 Description

Given a set of data points (x_i, y_i), for $i = 1, 2, \ldots, n$, nag_1d_minimax_polynomial (e02alc) uses the exchange algorithm to compute an mth-degree polynomial

$$\mathrm{P}(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_m x^m$$

such that $\max_i |\mathrm{P}(x_i) - y_i|$ is a minimum.

The function also returns a number whose absolute value is the final reference deviation (see Section 5). The function is an adaptation of Boothroyd (1967).

4 References

Boothroyd J B (1967) Algorithm 318 *Comm. ACM* **10** 801

Stieffel E (1959) Numerical methods of Tchebycheff approximation *On Numerical Approximation* (ed R E Langer) 217–232 University of Wisconsin Press

5 Arguments

1: **n** – Integer *Input*

On entry: n, the number of data points.

Constraint: $\mathbf{n} \geq 1$.

2: **x**[**n**] – const double *Input*

On entry: the values of the x coordinates, x_i, for $i = 1, 2, \ldots, n$.

Constraint: $x_1 < x_2 < \ldots < x_n$.

3: **y**[**n**] – const double *Input*

On entry: the values of the y coordinates, y_i, for $i = 1, 2, \ldots, n$.

4: **m** – Integer *Input*

On entry: m, where m is the degree of the polynomial to be found.

Constraint: $0 \leq \mathbf{m} < \min(100, \mathbf{n} - 1)$.

5: **a**[**m** + **1**] – double *Output*

On exit: the coefficients a_i of the minimax polynomial, for $i = 0, 1, \ldots, m$.

6: **ref** – double * *Output*

On exit: the final reference deviation, i.e., the maximum deviation of the computed polynomial evaluated at x_i from the reference values y_i, for $i = 1, 2, \ldots, n$. **ref** may return a negative value which indicates that the algorithm started to cycle due to round-off errors.

7: **fail** – NagError * *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.

NE_BAD_PARAM

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

NE_INT

On entry, **m** = ⟨*value*⟩.
Constraint: **m** < 100.

On entry, **m** = ⟨*value*⟩.
Constraint: **m** ≥ 0.

On entry, **n** = ⟨*value*⟩.
Constraint: **n** ≥ 1.

NE_INT_2

On entry, **m** = ⟨*value*⟩ and **n** = ⟨*value*⟩.
Constraint: **m** < **n** − 1.

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_NOT_STRICTLY_INCREASING

On entry, $i = \langle value \rangle$, $\mathbf{x}[i] = \langle value \rangle$ and $\mathbf{x}[i-1] = \langle value \rangle$.
Constraint: $\mathbf{x}[i] > \mathbf{x}[i-1]$.

7 Accuracy

This is dependent on the given data points and on the degree of the polynomial. The data points should represent a fairly smooth function which does not contain regions with markedly different behaviours. For large numbers of data points (**n** > 100, say), rounding error will affect the computation regardless of the quality of the data; in this case, relatively small degree polynomials ($\mathbf{m} \ll \sqrt{\mathbf{n}}$) may be used when this is consistent with the required approximation. A limit of 99 is placed on the degree of polynomial since it is known from experiment that a complete loss of accuracy often results from using such high degree polynomials in this form of the algorithm.

8 Parallelism and Performance

nag_1d_minimax_polynomial (e02alc) is not threaded by NAG in any implementation.

nag_1d_minimax_polynomial (e02alc) 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 Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

The time taken increases with m.

10 Example

This example calculates a minimax fit with a polynomial of degree 5 to the exponential function evaluated at 21 points over the interval $[0, 1]$. It then prints values of the function and the fitted polynomial.

10.1 Program Text

```
/* nag_1d_minimax_polynomial (e02alc) Example Program.
 *
 * Copyright 2013 Numerical Algorithms Group.
 *
 * Mark 24, 2013.
 */
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nage02.h>

int main(void)
{

  /* Scalars */
  Integer     exit_status = 0;
  double      dxx, ref, s, t, xx;
  Integer     i, j, m, n, neval;
  /* Local Arrays */
  double      *a = 0,  *x = 0,  *y = 0;
  /* NAG types */
  NagError    fail;

  INIT_FAIL(fail);

  printf("nag_1d_minimax_polynomial (e02alc) Example Program Results\n\n");

  /* Skip heading in data file */
  scanf("%*[^\n] ");

  /* n is number of data points to be fitted,
   * m is degree of fitting polynomial
   * neval is number of evaluation points of fitted polynomial
   */
  scanf("%ld%ld%ld%*[^\n] ", &n, &m, &neval);

  if (
      !(a = NAG_ALLOC((m + 1), double))||
      !(x = NAG_ALLOC((n), double))||
      !(y = NAG_ALLOC((n), double))
      )
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  for (i=0; i<n; i++)
```

```
    scanf("%lf%lf", &x[i], &y[i]);
  scanf("%*[^\n] ");

  /* Fit minimax polynomial of degree m using
   * nag_1d_minimax_polynomial (e02alc).
   */
  nag_1d_minimax_polynomial(n, x, y, m, a, &ref, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_1d_minimax_polynomial (e02alc).\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
    }
  printf("\n  Polynomial coefficients\n");
  for (i=0; i<m+1; i++)
    printf("     %12.4e \n", a[i]);
  printf("\n\n  Reference deviation = %10.2e\n\n", ref);
  printf("  x      Fit     exp(x)   Residual\n");

  /* The neval evaluation points are equispaced on [0,1]. */
  dxx = 1.0/(double)(neval - 1);
  for (j=0; j<neval; j++) {
    xx = (double)(j) * dxx;
    s = a[m];
    for ( i=m-1; i>=0; i--)
      s = s * xx + a[i];
    t = exp(xx);
    printf("%5.2f%9.4f%9.4f%11.2e\n", xx, s, t, s-t);
  }
 END:
  NAG_FREE(a);
  NAG_FREE(x);
  NAG_FREE(y);

  return exit_status;
}
```

10.2 Program Data

```
nag_1d_minimax_polynomial (e02alc) Example Program Data
 21    5   11             : n, m, neval
  0.00      1.0000000000
  0.05      1.0512710964
  0.10      1.1051709181
  0.15      1.1618342427
  0.20      1.2214027582
  0.25      1.2840254167
  0.30      1.3498588076
  0.35      1.4190675486
  0.40      1.4918246976
  0.45      1.5683121855
  0.50      1.6487212707
  0.55      1.7332530179
  0.60      1.8221188004
  0.65      1.9155408290
  0.70      2.0137527075
  0.75      2.1170000166
  0.80      2.2255409285
  0.85      2.3396468519
  0.90      2.4596031112
  0.95      2.5857096593
  1.00      2.7182818285   : (x[i],y[i]), i=0,..,n-1
```

10.3 Program Results

```
nag_1d_minimax_polynomial (e02alc) Example Program Results


  Polynomial coefficients
       1.0000e+00
       1.0001e+00
       4.9909e-01
       1.7042e-01
       3.4784e-02
       1.3909e-02


  Reference deviation =   1.09e-06

  x      Fit     exp(x)   Residual
0.00   1.0000   1.0000  -1.09e-06
0.10   1.1052   1.1052   9.74e-07
0.20   1.2214   1.2214  -7.44e-07
0.30   1.3499   1.3499  -9.18e-07
0.40   1.4918   1.4918   2.99e-07
0.50   1.6487   1.6487   1.09e-06
0.60   1.8221   1.8221   4.59e-07
0.70   2.0138   2.0138  -8.16e-07
0.80   2.2255   2.2255  -8.42e-07
0.90   2.4596   2.4596   8.75e-07
1.00   2.7183   2.7183  -1.09e-06
```