NAG Toolbox

nag_1d_minimax_polynomial (e02al)

1 Purpose

nag_1d_minimax_polynomial (e02al) calculates a minimax polynomial fit to a set of data points.

2 Syntax

```
[a, ref, ifail] = nag_1d_minimax_polynomial(x, y, m, 'n', n)
[a, ref, ifail] = e02al(x, y, m, 'n', n)
```

3 Description

Given a set of data points (x_i, y_i), for $i = 1, 2, \ldots, n$, nag_1d_minimax_polynomial (e02al) 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 Parameters

5.1 Compulsory Input Parameters

1: **x**(**n**) – REAL (KIND=nag_wp) array

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

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

2: **y**(**n**) – REAL (KIND=nag_wp) array

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

3: **m** – INTEGER

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

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

5.2 Optional Input Parameters

1: **n** – INTEGER

Default: the dimension of the arrays **x**, **y**. (An error is raised if these dimensions are not equal.)

n, the number of data points.

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

5.3 Output Parameters

1: **a**$(\mathbf{m}+\mathbf{1})$ – REAL (KIND=nag_wp) array

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

2: **ref** – REAL (KIND=nag_wp)

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.

3: **ifail** – INTEGER

ifail $= 0$ unless the function detects an error (see Section 5).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 1$

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

ifail $= 2$

Constraint: $\mathbf{m} < 100$.

Constraint: $\mathbf{m} < \mathbf{n} - 1$.

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

ifail $= 3$

Constraint: $\mathbf{x}(i+1) > \mathbf{x}(i)$.

ifail $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

ifail $= -399$

Your licence key may have expired or may not have been installed correctly.

ifail $= -999$

Dynamic memory allocation failed.

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 ($\mathbf{n} > 100$, 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 Further Comments

The time taken increases with m.

9 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.

9.1 Program Text

```
     function e02al_example

fprintf('e02al example results\n\n');

% Minimax polynomial of degree 5 that fits exp(x) on grid in [0,1].
x = [0:0.05:1];
y = exp(x);
m = nag_int(5);
[a, ref, ifail] = e02al(x, y, m);

fprintf('   Polynomial coefficients\n');
fprintf('        %7.4f\n',a(1:m+1));
fprintf('\n   Reference deviation = %8.2e\n\n', ref);
fprintf('   x     Fit      exp(x)   Residual\n');

xx = [0:0.1:1];
p  = a(m+1:-1:1);
s  = polyval(p,xx);
yy = exp(xx);

for i=1:11
  fprintf('%6.2f%9.4f%9.4f%11.2e\n',xx(i), s(i), yy(i), s(i) - yy(i));
end
```

9.2 Program Results

```
     e02al example results

   Polynomial coefficients
         1.0000
         1.0001
         0.4991
         0.1704
         0.0348
         0.0139

   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
```