NAG Toolbox for MATLAB

e02ag

1 Purpose

e02ag computes constrained weighted least-squares polynomial approximations in Chebyshev-series form to an arbitrary set of data points. The values of the approximations and any number of their derivatives can be specified at selected points.

2 Syntax

```
[a, s, np1, wrk, ifail] = e02ag(k, xmin, xmax, x, y, w, xf, yf, ip,
lwrk, 'm', m, 'mf', mf)
```

3 Description

e02ag determines least-squares polynomial approximations of degrees up to k to the set of data points (x_r, y_r) with weights w_r, for $r = 1, 2, \ldots, m$. The value of k, the maximum degree required, is prescribed by you. At each of the values $\mathbf{xf}_r$, for $r = 1, 2, \ldots, \mathbf{mf}$, of the independent variable x, the approximations and their derivatives up to order p_r are constrained to have one of the user-specified values $\mathbf{yf}_s$, for $s = 1, 2, \ldots, n$, where $n = \mathbf{mf} + \sum_{r=0}^{\mathbf{mf}} p_r$.

The approximation of degree i has the property that, subject to the imposed constraints, it minimizes σ_i, the sum of the squares of the weighted residuals ϵ_r, for $r = 1, 2, \ldots, m$, where

$$\epsilon_r = w_r(y_r - f_i(x_r))$$

and $f_i(x_r)$ is the value of the polynomial approximation of degree i at the rth data point.

Each polynomial is represented in Chebyshev-series form with normalized argument $\bar{x}$. This argument lies in the range -1 to $+1$ and is related to the original variable x by the linear transformation

$$\bar{x} = \frac{2x - (x_{\max} + x_{\min})}{(x_{\max} - x_{\min})}$$

where $x_{\min}$ and $x_{\max}$, specified by you, are respectively the lower and upper end points of the interval of x over which the polynomials are to be defined.

The polynomial approximation of degree i can be written as

$$\tfrac{1}{2}a_{i,0} + a_{i,1}T_1(\bar{x}) + \cdots + a_{ij}T_j(\bar{x}) + \cdots + a_{ii}T_i(\bar{x})$$

where $T_j(\bar{x})$ is the Chebyshev polynomial of the first kind of degree j with argument $\bar{x}$. For $i = n, n+1, \ldots, k$, the function produces the values of the coefficients a_{ij}, for $j = 0, 1, \ldots, i$, together with the value of the root-mean-square residual, S_i, defined as

$$\sqrt{\frac{\sigma_i}{(m' + n - i - 1)}},$$

where m' is the number of data points with nonzero weight.

Values of the approximations may subsequently be computed using e02ae or e02ak.

First e02ag determines a polynomial $\mu(\bar{x})$, of degree $n - 1$, which satisfies the given constraints, and a polynomial $\nu(\bar{x})$, of degree n, which has value (or derivative) zero wherever a constrained value (or derivative) is specified. It then fits $y_r - \mu(x_r)$, for $r = 1, 2, \ldots, m$, with polynomials of the required degree in $\bar{x}$ each with factor $\nu(\bar{x})$. Finally the coefficients of $\mu(\bar{x})$ are added to the coefficients of these fits to give the coefficients of the constrained polynomial approximations to the data points (x_r, y_r), for $r = 1, 2, \ldots, m$. The method employed is given in Hayes 1970: it is an extension of Forsythe's orthogonal polynomials method (see Forsythe 1957) as modified by Clenshaw (see Clenshaw 1960).

4 References

Clenshaw C W 1960 Curve fitting with a digital computer *Comput. J.* **2** 170–173

Forsythe G E 1957 Generation and use of orthogonal polynomials for data fitting with a digital computer *J. Soc. Indust. Appl. Math.* **5** 74–88

Hayes J G (ed.) 1970 *Numerical Approximation to Functions and Data* Athlone Press, London

5 Parameters

5.1 Compulsory Input Parameters

1: **k – int32 scalar**

k, the maximum degree required.

Constraint: $n \leq \mathbf{k} \leq m'' + n - 1$ where n is the total number of constraints and m'' is the number of data points with nonzero weights and distinct abscissae which do not coincide with any of the $\mathbf{xf}_r$.

2: **xmin – double scalar**

3: **xmax – double scalar**

The lower and upper end points, respectively, of the interval $[x_{\min}, x_{\max}]$. Unless there are specific reasons to the contrary, it is recommended that **xmin** and **xmax** be set respectively to the lowest and highest value among the x_r and $\mathbf{xf}_r$. This avoids the danger of extrapolation provided there is a constraint point or data point with nonzero weight at each end point.

Constraint: **xmax** > **xmin**.

4: **x(m) – double array**

$\mathbf{x}(r)$ must contain the value x_r of the independent variable at the rth data point, for $r = 1, 2, \ldots, m$.

Constraint: the $\mathbf{x}(r)$ must be in nondecreasing order and satisfy $\mathbf{xmin} \leq \mathbf{x}(r) \leq \mathbf{xmax}$.

5: **y(m) – double array**

$\mathbf{y}(r)$ must contain y_r, the value of the dependent variable at the rth data point, for $r = 1, 2, \ldots, m$.

6: **w(m) – double array**

$\mathbf{w}(r)$ must contain the weight w_r to be applied to the data point x_r, for $r = 1, 2 \ldots, m$. For advice on the choice of weights see the E02 Chapter Introduction. Negative weights are treated as positive. A zero weight causes the corresponding data point to be ignored. Zero weight should be given to any data point whose x and y values both coincide with those of a constraint (otherwise the denominators involved in the root-mean-square residuals s_i will be slightly in error).

7: **xf(mf) – double array**

$\mathbf{xf}(r)$ must contain the rth value of the independent variable at which a constraint is specified, for $r = 1, 2, \ldots, \mathbf{mf}$.

Constraint: these values need not be ordered but must be distinct and satisfy $\mathbf{xmin} \leq \mathbf{xf}(r) \leq \mathbf{xmax}$.

8: **yf(lyf) – double array**

The values which the approximating polynomials and their derivatives are required to take at the points specified in **xf**. For each value of $\mathbf{xf}(r)$, **yf** contains in successive elements the required value of the approximation, its first derivative, second derivative, $\ldots, p_r$th derivative, for $r = 1, 2, \ldots, \mathbf{mf}$. Thus the value which the kth derivative of each approximation ($k = 0$ referring to the approximation itself) is required to take at the point $\mathbf{xf}(r)$ must be contained in $\mathbf{yf}(s)$, where

$$s = r + k + p_1 + p_2 + \cdots + p_{r-1},$$

for $k = 0, 1, \ldots, p_r$ and $r = 1, 2, \ldots, \mathbf{mf}$. The derivatives are with respect to your variable x.

9: **ip(mf) – int32 array**

ip(r) must contain p_r, the order of the highest-order derivative specified at **xf**(r), for $r = 1, 2, \ldots, \mathbf{mf}$. $p_r = 0$ implies that the value of the approximation at **xf**(r) is specified, but not that of any derivative.

Constraint: **ip**$(r) \geq 0$, for $r = 1, 2, \ldots, \mathbf{mf}$.

10: **lwrk – int32 scalar**

Constraint: **lwrk** $\geq \max(4 \times \mathbf{m} + 3 \times \mathbf{kplus1}, 8 \times n + 5 \times IPMAX + \mathbf{mf} + 10) + 2 \times n + 2$, where $IPMAX = \max(\mathbf{ip}(r))$, for $r = 1, 2, \ldots, \mathbf{mf}$.

5.2 Optional Input Parameters

1: **m – int32 scalar**

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

the number m of data points to be fitted.

Constraint: **m** ≥ 1.

2: **mf – int32 scalar**

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

the number, of values of the independent variable at which a constraint is specified.

Constraint: **mf** ≥ 1.

5.3 Input Parameters Omitted from the MATLAB Interface

kplus1, lda, lyf, np1, iwrk, liwrk

5.4 Output Parameters

1: **a(lda,kplus1) – double array**

a$(i+1, j+1)$ contains the coefficient a_{ij} in the approximating polynomial of degree i, for $i = n, n+1, \ldots, k$ and $j = 0, 1, \ldots, i$.

2: **s(kplus1) – double array**

s$(i+1)$ contains s_i, for $i = n, n+1, \ldots, k$, the root-mean-square residual corresponding to the approximating polynomial of degree i. In the case where the number of data points with nonzero weight is equal to $k + 1 - n$, s_i is indeterminate: the function sets it to zero. For the interpretation of the values of s_i and their use in selecting an appropriate degree, see Section 3.1 in the E02 Chapter Introduction.

3: **np1 – int32 scalar**

$n + 1$, where n is the total number of constraint conditions imposed: $n = \mathbf{mf} + p_1 + p_2 + \cdots + p_{\mathbf{mf}}$.

4: **wrk(lwrk) – double array**

Contains weighted residuals of the highest degree of fit determined (k). The residual at x_r is in element $2(n+1) + 3(m+k+1) + r$, for $r = 1, 2, \ldots, m$. The rest of the array is used as workspace.

5: **ifail – int32 scalar**

0 unless the function detects an error (see Section 6).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 1$

On entry, **m** < 1,
or **kplus1** $< n + 1$,
or **lda** $<$ **kplus1**,
or **mf** < 1,
or **lyf** $< n$,
or **lwrk** is too small (see Section 5),
or **liwrk** $< 2 \times$ **mf** $+ 2$.

(Here n is the total number of constraint conditions.)

ifail $= 2$

ip$(r) < 0$ for some $r = 1, 2, \ldots,$ **mf**.

ifail $= 3$

xmin $\geq$ **xmax**, or **xf**(r) is not in the interval **xmin** to **xmax** for some $r = 1, 2, \ldots,$ **mf**, or the **xf**(r) are not distinct.

ifail $= 4$

x(r) is not in the interval **xmin** to **xmax** for some $r = 1, 2, \ldots,$ **m**.

ifail $= 5$

x$(r) <$ **x**$(r - 1)$ for some $r = 2, 3, \ldots,$ **m**.

ifail $= 6$

kplus1 $> m'' + n$, where m'' is the number of data points with nonzero weight and distinct abscissae which do not coincide with any **xf**(r). Thus there is no unique solution.

ifail $= 7$

The polynomials $\mu(x)$ and/or $\nu(x)$ cannot be determined. The problem supplied is too ill-conditioned. This may occur when the constraint points are very close together, or large in number, or when an attempt is made to constrain high-order derivatives.

7 Accuracy

No complete error analysis exists for either the interpolating algorithm or the approximating algorithm. However, considerable experience with the approximating algorithm shows that it is generally extremely satisfactory. Also the moderate number of constraints, of low-order, which are typical of data fitting applications, are unlikely to cause difficulty with the interpolating function.

8 Further Comments

The time taken to form the interpolating polynomial is approximately proportional to n^3, and that to form the approximating polynomials is very approximately proportional to $m(k + 1)(k + 1 - n)$.

To carry out a least-squares polynomial fit without constraints, use e02ad. To carry out polynomial interpolation only, use e01ae.

9 Example

```
k = int32(4);
xmin = 0;
xmax = 4;
x = [0.5;
     1;
     2;
     2.5;
     3];
y = [0.03;
     -0.75;
     -1;
     -0.1;
     1.75];
w = [1;
     1;
     1;
     1;
     1];
xf = [0;
     4];
yf = [1;
     -2;
     9];
ip = [int32(1);
     int32(0)];
lwrk = int32(200);
[a, s, np1, wrk, ifail] = e02ag(k, xmin, xmax, x, y, w, xf, yf, ip, lwrk)
```

```
a =
         0         0         0         0         0
         0         0         0         0         0
         0         0         0         0         0
    3.9980    3.4995    3.0010    0.5005         0
    3.9980    3.4995    3.0010    0.5005   -0.0000
s =
         0
         0
         0
    0.0025
    0.0029
np1 =
           4
wrk =
     array elided
ifail =
           0
```