NAG Library Function Document

nag_1d_cheb_fit (e02adc)

1 Purpose

nag_1d_cheb_fit (e02adc) computes weighted least squares polynomial approximations to an arbitrary set of data points.

2 Specification

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

void nag_1d_cheb_fit (Integer m, Integer kplus1, Integer tda, 
                       const double x[], const double y[], const double w[], double a[], 
                       double s[], NagError *fail)
```

3 Description

nag_1d_cheb_fit (e02adc) determines least squares polynomial approximations of degrees 0, 1, ..., k to the set of data points \((x_r, y_r)\) with weights \(w_r\), for \(r = 1, 2, \ldots, m\).

The approximation of degree \(i\) has the property that it minimizes \(\sigma_i\), the sum of squares of the weighted residuals \(\epsilon_r\), where

\[
\epsilon_r = w_r(y_r - f_r)
\]

and \(f_r\) is the value of the polynomial of degree \(i\) at the \(r\)th data point.

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

\[
\tilde{x} = \frac{2x - x_{\max} - x_{\min}}{x_{\max} - x_{\min}}.
\]

Here \(x_{\max}\) and \(x_{\min}\) are respectively the largest and smallest values of \(x_r\). The polynomial approximation of degree \(i\) is represented as

\[
\frac{1}{2}(a_{i+1,0}T_0(\tilde{x}) + a_{i+1,1}T_1(\tilde{x}) + a_{i+1,2}T_2(\tilde{x}) + \cdots + a_{i+1,i+1}T_i(\tilde{x})),
\]

where \(T_j(\tilde{x})\) is the Chebyshev polynomial of the first kind of degree \(j\) with argument \(\tilde{x}\).

For \(i = 0, 1, \ldots, k\), the function produces the values of \(a_{i+1,j+1}\), for \(j = 0, 1, \ldots, i\), together with the value of the root mean square residual \(s_i = \sqrt{\sigma_i/(m - i - 1)}\). In the case \(m = i + 1\) the function sets the value of \(s_i\) to zero.

The method employed is due to Forsythe (1957) and is based upon the generation of a set of polynomials orthogonal with respect to summation over the normalized dataset. The extensions due to Clenshaw (1960) to represent these polynomials as well as the approximating polynomials in their Chebyshev series forms are incorporated. The modifications suggested by Reinsch and Gentleman (Gentleman (1969)) to the method originally employed by Clenshaw for evaluating the orthogonal polynomials from their Chebyshev series representations 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 Chebyshev series representations of the polynomial approximations should be carried out using nag_1d_cheb_eval (e02aec).
4 References


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


5 Arguments

1: \( m \) – Integer \hspace{1cm} \text{Input}

\( m \) – the number \( m \) of data points.

\text{Constraint: } m \geq \text{mdist} \geq 2, \text{ where mdist is the number of distinct } x \text{ values in the data.}

2: \( kplus1 \) – Integer \hspace{1cm} \text{Input}

\( kplus1 \) – integer \( k+1 \), where \( k \) is the maximum degree required.

\text{Constraint: } 0 < kplus1 \leq \text{mdist}, \text{ where mdist is the number of distinct } x \text{ values in the data.}

3: \( tda \) – Integer \hspace{1cm} \text{Input}

\( tda \) – integer \( tda \), the stride separating matrix column elements in the array \( a \).

\text{Constraint: } tda \geq kplus1.

4: \( x[m] \) – const double \hspace{1cm} \text{Input}

\( x[m] \) – the values \( x_r \) of the independent variable, for \( r = 1, 2, \ldots, m \).

\text{Constraint: } \text{the values must be supplied in nondecreasing order with } x[m - 1] > x[0].

5: \( y[m] \) – const double \hspace{1cm} \text{Input}

\( y[m] \) – the values \( y_r \) of the dependent variable, for \( r = 1, 2, \ldots, m \).

6: \( w[m] \) – const double \hspace{1cm} \text{Input}

\( w[m] \) – the set of weights, \( w_r \), for \( r = 1, 2, \ldots, m \). For advice on the choice of weights, see the e02 Chapter Introduction.

\text{Constraint: } w[r] > 0.0, \text{ for } r = 0, 1, \ldots, m - 1.

7: \( a[kplus1 \times tda] \) – double \hspace{1cm} \text{Output}

\( a[kplus1 \times tda] \) – contains the coefficient \( a_{i+1,j+1} \times tda + j \) for \( i = 0, 1, \ldots, k \) and \( j = 0, 1, \ldots, i \).

8: \( s[kplus1] \) – double \hspace{1cm} \text{Output}

\( s[kplus1] \) – contains the root mean square residual \( s_i \), for \( i = 0, 1, \ldots, k \), as described in Section 3. For the interpretation of the values of the \( s_i \) and their use in selecting an appropriate degree, see the e02 Chapter Introduction.
9:  fail – NagError *
    The NAG error argument (see Section 3.6 in the Essential Introduction).

6  Error Indicators and Warnings

NE_2_INT_ARG_GT
    On entry, kplus1 = \langle value\rangle while the number of distinct x values, mdist = \langle value\rangle. These arguments must satisfy kplus1 ≤ mdist.

NE_2_INT_ARG_LT
    On entry, tda = \langle value\rangle while kplus1 = \langle value\rangle. The arguments must satisfy tda ≥ kplus1.

NE_ALLOC_FAIL
    Dynamic memory allocation failed.

NE_INT_ARG_LT
    On entry, kplus1 must not be less than 1: kplus1 = \langle value\rangle.

NE_NO_NORMALISATION
    On entry, all the x[r] in the sequence x[r], r = 0, 1, ..., m - 1 are the same.

NE_NOT_NON_DECREASING
    On entry, the sequence x[r], r = 0, 1, ..., m - 1 is not in nondecreasing order.

NE_WEIGHTS_NOT_POSITIVE
    On entry, the weights are not strictly positive: w[\langle value\rangle] = \langle value\rangle.

7  Accuracy

No error analysis for the method has been published. Practical experience with the method, however, is generally extremely satisfactory.

8  Parallelism and Performance

Not applicable.

9  Further Comments

The time taken by nag_1d_cheb_fit (e02adc) is approximately proportional to m(k + 1)(k + 11).

The approximating polynomials may exhibit undesirable oscillations (particularly near the ends of the range) if the maximum degree k exceeds a critical value which depends on the number of data points m and their relative positions. As a rough guide, for equally spaced data, this critical value is about 2 × \sqrt{m}. For further details see page 60 of Hayes (1970).

10  Example

Determine weighted least squares polynomial approximations of degrees 0, 1, 2 and 3 to a set of 11 prescribed data points. For the approximation of degree 3, tabulate the data and the corresponding values of the approximating polynomial, together with the residual errors, and also the values of the approximating polynomial at points half-way between each pair of adjacent data points.
The example program supplied is written in a general form that will enable polynomial approximations of degrees \(0, 1, \ldots, k\) to be obtained to \(m\) data points, with arbitrary positive weights, and the approximation of degree \(k\) to be tabulated. \texttt{nag_1d_cheb_eval} (e02aec) is used to evaluate the approximating polynomial. The program is self-starting in that any number of datasets can be supplied.

**10.1 Program Text**

\[
/* \text{nag_1d_cheb_fit (e02adc) Example Program.} */
\]

\[
* \text{Copyright 2014 Numerical Algorithms Group.}
* \text{Mark 5, 1998.}
* \text{Mark 8 revised, 2004.}
*/
\]

\[
#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nage02.h>
\]

\[
int main(void)
\{
    \#define A(I, J) a[(I) *tda + J]

    Integer exit_status = 0, i, iwght, j, k, m, r, tda;
    NagError fail;
    double *a = 0, *ak = 0, d1, fit, *s = 0, *w = 0, *x = 0, x1, xarg, xcapr,
            xm, *y = 0;

    INIT_FAIL(fail);

    printf("nag_1d_cheb_fit (e02adc) Example Program Results \n");

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

    \#ifdef _WIN32
    while ((scanf_s("%NAG_IFMT", &m)) != EOF)
    \#else
    while ((scanf("%NAG_IFMT", &m)) != EOF)
    \#endif
    \{
        if (m >= 2)
        {
            if (!
                (x = NAG_ALLOC(m, double)) ||
                (y = NAG_ALLOC(m, double)) ||
                (w = NAG_ALLOC(m, double)))
            {
                printf("Allocation failure\n");
                exit_status = -1;
                goto END;
            }
        }
        else
        {
            printf("Invalid m.\n");
            exit_status = 1;
            return exit_status;
        }
    \#ifdef _WIN32
    scanf_s("%NAG_IFMT", &k);
    \#else
scanf("%"NAG_IFMT"", &k);
#endif
if (k >= 1)
{
    if (!a = NAG_ALLOC((k+1)*(k+1), double)) ||
    !(s = NAG_ALLOC(k+1, double)) ||
    !(ak = NAG_ALLOC(k+1, double))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }
    tda = k+1;
} else
{
    printf("Invalid k.\n");
    exit_status = 1;
    return exit_status;
}
#else
    scanf("%"NAG_IFMT"", &iwght);
#endif
for (r = 0; r < m; ++r)
{
    if (iwght != 1)
    {
        scanf("%lf", &x[r]);
        #ifdef _WIN32
            scanf_s("%lf", &x[r]);
        #else
            scanf("%lf", &x[r]);
        #endif
        scanf("%lf", &y[r]);
        #ifdef _WIN32
            scanf_s("%lf", &y[r]);
        #else
            scanf("%lf", &y[r]);
        #endif
        scanf("%lf", &w[r]);
        #ifdef _WIN32
            scanf_s("%lf", &w[r]);
        #else
            scanf("%lf", &w[r]);
        #endif
    } else
    {
        scanf("%lf", &x[r]);
        #ifdef _WIN32
            scanf_s("%lf", &x[r]);
        #else
            scanf("%lf", &x[r]);
        #endif
        scanf("%lf", &y[r]);
        #ifdef _WIN32
            scanf_s("%lf", &y[r]);
        #else
            scanf("%lf", &y[r]);
        #endif
        w[r] = 1.0;
    }
}
/* nag_1d_cheb_fit (e02adc).
* Computes the coefficients of a Chebyshev series
* polynomial for arbitrary data
*/
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_1d_cheb_fit (e02adc).\n", fail.message);
    exit_status = 1;
}
for (i = 0; i <= k; ++i)
{
    printf("\n");
    printf("%s%4"NAG_IFMT"%s%12.2e\n", "Degree", i,
            " R.M.S. residual =", s[i]);
    printf("\n J Chebyshev coeff A(J) \n");
    for (j = 0; j < i+1; ++j)
        printf("%s%4"NAG_IFMT"%15.4f\n", j+1, A(i, j));
}

for (j = 0; j < k+1; ++j)
    ak[j] = A(k, j);
x1 = x[0];
xa = x[m-1];
printf("\n %s%4"NAG_IFMT"\n\n", "Polynomial approximation and residuals for degree", k);
printf(" R Abscissa Weight Ordinate Polynomial Residual \n");
for (r = 1; r <= m; ++r)
{
    xcapr = (x[r-1] - x1 - (xa - x[r-1])) / (xa - x1);
    /* nag_1d_cheb_eval (e02aec).
     * Evaluates the coefficients of a Chebyshev series
     * polynomial */
    nag_1d_cheb_eval(k+1, ak, xcapr, &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;
    }
    d1 = fit - y[r-1];
    printf(" %3"NAG_IFMT"%11.4f%11.4f%11.4f%11.2e\n", r, x[r-1],
           w[r-1], y[r-1], fit, d1);
    if (r < m)
    {
        xarg = (x[r-1] + x[r]) * 0.5;
        xcapr = (xarg - x1 - (xa - xarg)) / (xa - x1);
        /* nag_1d_cheb_eval (e02aec), see above. */
        nag_1d_cheb_eval(k+1, ak, xcapr, &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(" %11.4f %11.4f\n", xarg, fit);
    }
}

END:
NAG_FREE(a);
NAG_FREE(x);
NAG_FREE(y);
NAG_FREE(w);
NAG_FREE(s);
NAG_FREE(ak);
return exit_status;
10.2 Program Data

nag_1d_cheb_fit (e02adc) Example Program Data

<table>
<thead>
<tr>
<th>R</th>
<th>Abscissa</th>
<th>Weight</th>
<th>Ordinate</th>
<th>Polynomial</th>
<th>Residual</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0000</td>
<td>1.0000</td>
<td>10.4000</td>
<td>10.4461</td>
<td>4.61e-02</td>
</tr>
<tr>
<td></td>
<td>1.5500</td>
<td></td>
<td></td>
<td>9.3106</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2.1000</td>
<td>1.0000</td>
<td>7.9000</td>
<td>7.7977</td>
<td>-1.02e-01</td>
</tr>
<tr>
<td></td>
<td>2.6000</td>
<td></td>
<td></td>
<td>6.2555</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3.1000</td>
<td>1.0000</td>
<td>4.7000</td>
<td>4.7025</td>
<td>2.52e-03</td>
</tr>
<tr>
<td></td>
<td>3.5000</td>
<td></td>
<td></td>
<td>3.5488</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3.9000</td>
<td>1.0000</td>
<td>2.5000</td>
<td>2.5533</td>
<td>5.33e-02</td>
</tr>
<tr>
<td></td>
<td>4.4000</td>
<td></td>
<td></td>
<td>1.6435</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4.9000</td>
<td>1.0000</td>
<td>1.2000</td>
<td>1.2390</td>
<td>3.90e-02</td>
</tr>
<tr>
<td></td>
<td>5.3500</td>
<td></td>
<td></td>
<td>1.4257</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>5.8000</td>
<td>0.8000</td>
<td>2.2000</td>
<td>2.2425</td>
<td>4.25e-02</td>
</tr>
<tr>
<td></td>
<td>6.1500</td>
<td></td>
<td></td>
<td>3.3803</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>6.5000</td>
<td>0.8000</td>
<td>5.1000</td>
<td>5.0116</td>
<td>-8.84e-02</td>
</tr>
<tr>
<td></td>
<td>6.8000</td>
<td></td>
<td></td>
<td>6.8400</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>7.1000</td>
<td>0.7000</td>
<td>9.2000</td>
<td>9.0982</td>
<td>-1.02e-01</td>
</tr>
<tr>
<td></td>
<td>7.4500</td>
<td></td>
<td></td>
<td>12.3171</td>
<td></td>
</tr>
</tbody>
</table>

10.3 Program Results

nag_1d_cheb_fit (e02adc) Example Program Results

Degree 0 R.M.S. residual = 4.07e+00

J Chebyshev coeff A(J)
1  12.1740

Degree 1 R.M.S. residual = 4.28e+00

J Chebyshev coeff A(J)
1  12.2954
2   0.2740

Degree 2 R.M.S. residual = 1.69e+00

J Chebyshev coeff A(J)
1  20.7345
3   8.1876

Degree 3 R.M.S. residual = 6.82e-02

J Chebyshev coeff A(J)
1  24.1429
2   9.4065
3  10.8400
4  3.0589

Polynomial approximation and residuals for degree 3
<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>7.8000</td>
<td>0.5000</td>
<td>16.1000</td>
<td>16.2123</td>
</tr>
<tr>
<td>10</td>
<td>8.1000</td>
<td>20.1266</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>8.4000</td>
<td>24.5000</td>
<td>24.6048</td>
<td>1.05e-01</td>
</tr>
<tr>
<td>11</td>
<td>8.7000</td>
<td>29.6779</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>9.0000</td>
<td>35.3000</td>
<td>35.3769</td>
<td>7.69e-02</td>
</tr>
</tbody>
</table>