11.1 Polynomial Least-Squares Curve Fit

A. Purpose

This subroutine determines a univariate polynomial that fits a given discrete set of data in the sense of minimizing the weighted sum of squares of residuals. The fitted polynomial can be constrained to match some data points exactly by appropriate setting of the a priori standard deviations of the data errors.

Auxiliary subroutines, described in Chapter 11.2, may be used to evaluate, differentiate, or integrate a polynomial produced by this fitting procedure.

B. Usage

B.1 Program Prototype, Single Precision

INTEGER M, NMAX, NDEG

REAL X(≥M),**Y**(≥M),**SD**(≥M or =1),**P**(≥NMAX+3), **SIGFAC, W**(≥(NMAX+3)×(NMAX+3))

LOGICAL SEEKN, COMTRN, CHBBAS

Assign values to M, X(), Y(), SD(), NMAX, SEEKN, COMTRN and CHBBAS. If COMTRN = .FALSE. values must also be assigned to P(1) and P(2).

CALL SPFIT (M, X, Y, SD, NMAX, SEEKN, COMTRN, CHBBAS, P, NDEG,SIGFAC, W)

Following the CALL to SPFIT one may wish to use SCPVAL or SMPVAL to evaluate the fitted polynomial at specific points, SCPDRV or SMPDRV to differentiate the polynomial, or SCPINT or SMPINT to integrate the polynomial. See Chapter 11.2 for descriptions of these subprograms.

B.2 Argument Definitions

M [in] Number of data points.

X() [in] Array of values of the independent variable. Values need not be ordered and may be repeated.

Y() [in] Array of values of the dependent variable indexed to correspond to the X() values.

SD() [in] If SD(1) > 0., each SD(i) must be positive and will be assumed to be the user's a priori estimate of the standard deviation of the uncertainty (observational errors, etc.) in the corresponding data value Y(i).

If SD(1) < 0., |SD(1)| will be used as the a priori standard deviation of each data value Y(i). In this case the array SD() can be dimensioned SD(1) since locations following SD(1) will not be used.

NMAX [in] Specifies the maximum degree polynomial to be considered. Require NMAX ≥ 0. If NMAX > M − 1 the subroutine will function as though NMAX = M − 1.

SEEKN [in] If SEEKN = .TRUE. the subroutine will determine the optimum value of NDEG ≤ NMAX in the sense described below in Section D.

If SEEKN = .FALSE. the subroutine will set NDEG = NMAX unless this is a singular or nearly singular problem, in which case NDEG will be reduced as necessary.

COMTRN [in] If COMTRN = .TRUE. the subroutine will determine the transformation parameters P(1) and P(2) as described in Section D.

If COMTRN = .FALSE. the subroutine will use the values of P(1) and P(2) that have been set by the user.

CHBBAS [in] If CHBBAS = .TRUE. the subroutine will use the Chebyshev basis. If CHBBAS = .FALSE. the subroutine will use the monomial basis. The Chebyshev basis is recommended for better numerical stability, when NMAX is greater than about six.

P() [inout] Parameter vector defining a polynomial according to Eqs. (1) and (2) or (3) and (4). Values of P(1) and P(2) may either be set by the user, or by the subroutine, depending upon the value of COMTRN. Values of P(j+3), j = 0, ..., NMAX will be computed by the subroutine. If NDEG < NMAX, the subroutine will set P(i+3) = 0, for i = NDEG + 1, ..., NMAX.

NDEG [out] Set by the subroutine to indicate the degree of the fitted polynomial. On return NDEG will satisfy 0 ≤ NDEG ≤ NMAX if the computation was successful and will be set to −1 if an error occurred.

SIGFAC [out] Number computed by the subroutine. See discussion of SIGFAC in Sections C and D following.

W() [scratch] Array of working storage.

B.3 Modifications for Double Precision

For double precision usage change the REAL type statements to DOUBLE PRECISION and change the subroutine name from SPFIT to DPFIT. Change the names of the auxiliary subroutines mentioned above to DCPVAL, DMPVAL, DCPDRV, DMPDRV, DCPINT, and DMPINT, respectively.

C. Examples and Remarks

C.1 Example

Given a set of 12 data pairs $(x_i,\ y_i)$, compute the weighted least-squares polynomial fit to these data letting the subroutine determine the preferred polynomial degree not to exceed 8. After computing the least-squares polynomial $p(x)$, compute and tabulate the quantities x_i, y_i, $p(x_i)$, and $r_i = y_i - p(x_i)$.

The program DRSPFIT carries out this computation using subroutines SPFIT and SCPVAL. The output is shown in ODSPFIT. Note that the value of NDEG selected by the subroutine is 7.

C.2 Transformation of the Independent Variable

For the best numerical accuracy, both in determining the best fitting polynomial and in the later evaluation of the polynomial, it is generally advisable to use transformation parameters P(1) and P(2) that cause the transformed variable u of Eqs. (1) or (3) to range from -1 to $+1$. This condition can be obtained by setting COMTRN = .TRUE.

When fitting with polynomials of degree higher than about six, the Chebyshev basis generally gives better numerical accuracy than the monomial basis. To obtain the potential numerical advantages of the Chebyshev basis it is essential to cause the transformed variable, u, to range from -1 to $+1$.

To force an identity transformation (*i.e.*, effectively no transformation) of the independent variable, set COMTRN = .FALSE., P(1) = 0., and P(2) = 1.

C.3 Interpretation of SIGFAC

If the user has set all SD(i) = 1.0, or equivalently, set SD(1) = -1.0, SIGFAC can be interpreted as an a posteriori estimate of the standard deviation of the error in each Y(i) value. More generally, whatever values the user has assigned to the SD(i)'s, the a posteriori estimate of the standard deviation of the error in Y(i) is SIGFAC $\times$ SD(i).

C.4 Equality Constraints

If the user wishes the fitted polynomial to agree to machine accuracy with one or more of the data points, this can be accomplished by setting the SD() value for such points much smaller than for the other points.

Let ϵ denote the machine precision, *i.e.*, ϵ = R1MACH(4) in single precision or ϵ = D1MACH(4) in double precision, see Chapter 19.1. For some value of c in the range $0.5 \le c \le 0.75$, we suggest setting SD(i) = ϵ^c for the points at which an exact fit is desired and SD(i) = 1.0 for all other points. For example if $\epsilon = 10^{-8}$ set SD(i) in the range 10^{-4} through 10^{-6} for the exact fit points.

Using $c < 0.5$ may not produce sufficiently small residuals at the desired points of exact fit, whereas setting $c > 0.75$ may trip the near-singularity test in these subroutines leading to an unwanted alteration of the problem.

Note that it is not mathematically reasonable to attempt to force exact fits at more than NMAX+1 points.

D. Functional Description

The user provides data $(x_i,\ y_i,\ s_i)$, for $i = 1$, ..., M, and an integer NMAX. If SEEKN = .FALSE. and the problem is not rank-deficient or extremely ill-conditioned, the subroutine simply determines the polynomial $p_n(x)$ of degree n = NMAX that minimizes the following weighted sum of squares of residuals:

$$\rho_n^2 = \sum_{i=1}^{M} \left[\frac{y_i - p_n(x_i)}{s_i} \right]^2$$

The subroutine also computes

$$\sigma_n = \left[\frac{\rho_n^2}{\max(1, M-n-1)} \right]^{1/2}$$

and returns this value as SIGFAC.

The subroutine makes use of subroutines SROTMG and SROTM (or DROTMG and DROTM) from the BLAS1 (See Chapter 6.3.) to implement a sequential Modified Givens orthogonal transformation method of reducing the matrix of the least-squares problem to triangular form [1], and then solves the triangular system. This method has excellent numerical stability.

After triangularizing the matrix of the problem for degree NMAX, the quantities ρ_n^2 and σ_n^2 may be computed inexpensively for all degrees n from zero through NMAX. Thus, if the user sets SEEKN = .TRUE. the subroutine computes these values of ρ_n^2 and computes

$$\text{CMIN} = \min\{\sigma_n^2 : 0 \le n \le \text{NMAX}\}.$$

The quantity ρ_n is a strictly nonincreasing function of n whereas σ_n typically decreases initially with increasing n but then oscillates and slowly increases as n becomes so large that the polynomial $p_n(x)$ is fitting noise rather than a true polynomial signal in the data.

The subroutine sets NDEG to be the smallest value of n for which $\sigma_n^2 \le 1.01\times$CMIN, and it sets SIGFAC $= \sigma_{NDEG}$. This method of selecting NDEG is generally satisfactory when the ratio NMAX/M is reasonably small, say less than 1/3. If this ratio is larger there is a tendency for the NDEG selected by this method to be larger than one might pick if one viewed graphs of solution polynomials of different degrees.

Whatever the setting of SEEKN, if the problem for degree NMAX is rank-deficient, or very ill-conditioned, the subroutine will restrict consideration to lower degrees for which the problem is not extremely ill-conditioned. In particular this reduction of degree will certainly occur if NMAX+1 > M.

Parameterization of the polynomial, $p_n(x)$

In this subroutine an n^{th} degree polynomial $p_n(x)$ is represented in one of two special parametric forms (monomial basis or Chebyshev basis), both of which include the specification of a linear transformation of the independent variable in the parameterization. The parameters defining an n^{th} degree polynomial $p_n(x)$ are stored in the array P(i), $i = 1, ..., n + 3$.

Using the monomial basis these parameters define the polynomial $p_n(x)$ as follows:

$$u = \frac{x - P(1)}{P(2)} \tag{1}$$

$$p_n(x) = \sum_{i=0}^{n} P(i+3)u^i \tag{2}$$

whereas, if the Chebyshev basis is requested, $p_n(x)$ is defined as:

$$u = \frac{x - P(1)}{P(2)} \tag{3}$$

$$p_n(x) = \sum_{i=0}^{n} P(i+3)T_i(u) \tag{4}$$

where $T_i(u)$ denotes the Chebyshev polynomial of degree i. The Chebyshev polynomials may be defined as follows:

$$T_0(u) = 1, \quad T_1(u) = u, \quad T_2(u) = 2u^2 - 1$$
$$T_i(u) = 2u\ T_{i-1}(u) - T_{i-2}(u) \quad i = 3, 4, ...$$

The parameters P(1) and P(2) that occur in the transformation formula (1) or (3) may be set by the user (COMTRN = .FALSE.) or else computed by this subroutine (COMTRN = .TRUE.). In the latter case P(1) and P(2) will be computed as

$$\mathrm{P}(1) = (x_{\max} + x_{\min})/2$$
$$\mathrm{P}(2) = (x_{\max} - x_{\min})/2$$

where $x_{\max}$ and $x_{\min}$ are respectively the maximum and minimum values of x in the data array X(i), $i = 1, ...,$ M. This causes the transformed variable u of Eq. (1) or (3) to range from -1 to $+1$ as x ranges from $x_{\min}$ to $x_{\max}$.

References

1. Charles L. Lawson and Richard J. Hanson, **Solving Least-Squares Problems**, Prentice-Hall, Englewood Cliffs, N. J. (1974) 340 pages.

E. Error Procedures and Restrictions

The use of the Chebyshev basis is effective in improving the conditioning of the problem only if a transformation of the variable is used that maps the interval of interest for both fitting and evaluation to an interval approximately coincident with [−1., 1.].

The automatic procedure for selecting the degree of the fitted polynomial, used when SEEKN = .TRUE., tends to select the degree somewhat higher in some cases than one would choose by looking at plots of fits of different degrees.

This subroutine will issue an error message using the error processor in Chapter 19.2 at level 0, set NDEG = −1, and return if any of the following erroneous conditions exists:

1. SD(1) = 0, or M ≤ 0, or NMAX < 0.
2. COMTRN = .FALSE. and P(2) = 0.
3. SD(1) > 0. and some SD(i) ≤ 0. for $1 < i \leq$ M.

F. Supporting Information

The source language is Fortran 77.

Entry	Required Files
DPFIT	AMACH, DERM1, DERV1, DPFIT, DROTM, DROTMG, ERFIN, ERMSG, IERM1, IERV1
SPFIT	AMACH, ERFIN, ERMSG, IERM1, IERV1, SERM1, SERV1, SPFIT, SROTM, SROTMG

Initially designed and programmed by C. L. Lawson, JPL, 1970. Modified by Lawson and S. Y. Chiu, 1984, to use Modified Givens rather than Householder transformations to reduce the storage requirement for W() and the complexity of the program. Also, adapted the code to Fortran 77.

DRSPFIT

```
c      program DRSPFIT
c>> 1996-06-21 DRSPFIT Krogh Special code for C conversion.
c>> 1996-05-28 DRSPFIT Krogh Added external statement.
c>> 1995-09-15 DRSPFIT Krogh Declare all variables.
c>> 1995-09-15 DRSPFIT Krogh Remove '0' in format (again?)
c>> 1994-10-19 DRSPFIT Krogh  Changes to use M77CON
c>> 1994-08-09 DRSPFIT WVS Remove '0' in format
c>> 1991-11-20 DRSPFIT CLL Edited for Fortran 90
c>> 1987-12-09 DRSPFIT  Lawson  Initial Code.
c--S replaces "?": DR?PFIT, ?PFIT, ?CPVAL
c      Demonstration driver for SPFIT.
c++ Code for .C. is inactive
c%%   long int j;
c++ End
      external SCPVAL
      real               X(12),Y(12),P(11),SIGFAC,W(121),SCPVAL
      real               SIG(1), R, YFIT
      integer I, M, NDEG, NP3
c
      data X / 2.E0, 4.E0, 6.E0, 8.E0, 10.E0, 12.E0, 14.E0,
     *         16.E0, 18.E0, 20.E0, 22.E0, 24.E0 /
      data Y / 2.2E0, 4.0E0, 5.0E0, 4.6E0, 2.8E0, 2.7E0,
     *         3.8E0, 5.1E0, 6.1E0, 6.3E0, 5.0E0, 2.0E0 /
      data SIG(1) / -1.E0 /
      data M      / 12 /
c
 1001 format(1X/'  I     X         Y         YFIT   R=Y-YFIT'/1X)
 1002 format(1X,I2,F6.0,2F9.3,F10.3)
c
      call SPFIT(M,X,Y,SIG,8, .TRUE. , .TRUE. , .TRUE. ,P,NDEG,SIGFAC,W)
      NP3 = NDEG+3

c++ Code for ~.C. is active
      print
     *'(/'' NDEG ='',I2,10X,''SIGFAC ='',F8.4//
     *'' P(1),P(2) ='',9X,2F15.8//'' P(3),...,P(NDEG+3) ='',3F15.8/
     *(21X,3F15.8))', NDEG,SIGFAC,(P(I),I=1,NP3)
c++ Code for .C. is inactive
c%%   printf(
c%%      "\n NDEG =%2ld           SIGFAC =%8.4f",ndeg,sigfac );
c%%   printf(
c%%   "\n\n P(1),P(2) =          %15.8f%15.5f\n\n P(3),...,P(NDEG+3) =",
c%%   p[0], p[1]);
c%%   for (i = 2; i < (np3); i+=3){
c%%      for (j = i; j < (i < np3-2 ? i+3 : np3); j++)
c%%           printf("%15.8f", p[j] );
c%%           if (i < np3-2) printf("\n                          ");}
c%%   printf( "\n" );
c++ End
      write(*,1001)
      do 20 I = 1,M
        YFIT = SCPVAL(P,NDEG,X(I))
        R = Y(I) - YFIT
        write(*,1002)I,X(I),Y(I),YFIT,R
   20 continue
      stop
      end
```

ODSPFIT

NDEG = 7 SIGFAC = 0.2216

P(1),P(2) = 13.00000000 11.00000000

P(3),...,P(NDEG+3) = 3.99472594 0.57358360 −0.82918429
−0.58353752 −1.42390406 0.20219161
0.35689130 −0.29838806

I	X	Y	YFIT	R=Y–YFIT
1	2.	2.200	2.205	−0.005
2	4.	4.000	3.959	0.041
3	6.	5.000	5.147	−0.147
4	8.	4.600	4.333	0.267
5	10.	2.800	3.028	−0.228
6	12.	2.700	2.699	0.001
7	14.	3.800	3.651	0.149
8	16.	5.100	5.156	−0.056
9	18.	6.100	6.196	−0.096
10	20.	6.300	6.187	0.113
11	22.	5.000	5.048	−0.048
12	24.	2.000	1.992	0.008