Cubic Least-Squares Technique

Homework 12
Due 26 April 2023

The data (filled squares) shown in the Figure 6 can be well fitted with a 3rd-order polynomial of the form $f(x) = a_1 + a_2 x + a_3 x^2 + a_4 x^3$. The coefficients a_1, a_2, a_3, a_4 are determined by the set of linear equations

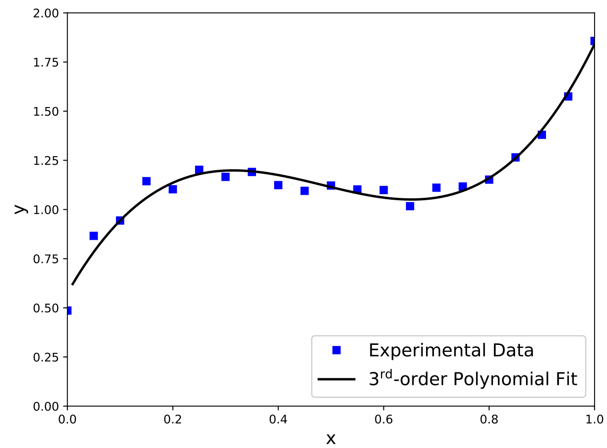

Figure 6: Experimental data fitted by a 3rd-order polynomial.

given by Eqs. (1) through (4):

$$m\ a_1 + \sum_{j=1}^{m} x_j\ a_2 + \sum_{j=1}^{m} x_j^2\ a_3 + \sum_{j=1}^{m} x_j^3\ a_4 = \sum_{j=1}^{m} y_j\,, \tag{1}$$

$$\sum_{j=1}^{m} x_j\ a_1 + \sum_{j=1}^{m} x_j^2\ a_2 + \sum_{j=1}^{m} x_j^3\ a_3 + \sum_{j=1}^{m} x_j^4\ a_4 = \sum_{j=1}^{m} y_j\ x_j\,, \tag{2}$$

$$\sum_{j=1}^{m} x_j^2\ a_1 + \sum_{j=1}^{m} x_j^3\ a_2 + \sum_{j=1}^{m} x_j^4\ a_3 + \sum_{j=1}^{m} x_j^5\ a_4 = \sum_{j=1}^{m} y_j\ x_j^2 \tag{3}$$

$$\sum_{j=1}^{m} x_j^3\ a_1 + \sum_{j=1}^{m} x_j^4\ a_2 + \sum_{j=1}^{m} x_j^5\ a_3 + \sum_{j=1}^{m} x_j^6\ a_4 = \sum_{j=1}^{m} y_j\ x_j^3\,. \tag{4}$$

The quantity m (=21) denotes the number of data points. In matrix notation, Eqs. (1) - (4) are given by $\mathbf{A}\,\mathbf{a} = \mathbf{b}$, where $\mathbf{a} = (a_1, a_2, a_3, a_4)^T$ contains the unknown coefficients, which we want to compute. They follow from

$$\mathbf{a} = \mathbf{A}^{-1}\,\mathbf{b}\,, \tag{5}$$

where $\mathbf{A}^{-1}$ denotes the inverse of matrix $\mathbf{A}$. Both $\mathbf{A}$ and $\mathbf{A}^{-1}$ are $n \times n = 4 \times 4$ matrices.

Tasks

Write a structured and well commented Fortran program which computes the coefficients a_1, a_2, a_3, and a_4. To do that you may use the Fortran subroutine `INVERSE` which computes the inverse of matrix A and returns the result to the calling program. You can download this subroutine and the experimental data set from the class website. The subroutine is called with

```
CALL INVERSE(A, C, n)
```

where the first argument is the coefficient matrix $\mathbf{A}$ of the system of linear equations (1) through (4). The second argument, $\mathbf{C}$, is an auxiliary matrix which, on return to the calling program, contains the elements of the inverse matrix, i.e., $\mathbf{C} = \mathbf{A}^{-1}$. The matrices $\mathbf{A}$ and $\mathbf{C}$ must be declared as `REAL` in the calling program.

Code Design

- The program determines dynamically how much space is to be allocated for all arrays and matrices.
- The data from `nonlinear.dat` is read with a `DO` loop to determine how much storage space must be allocated for the arrays and matrices.
- Do not forget to use the `REWIND` command to reposition the data file to the beginning of the file so that the next `READ` statement will read the first record.
- Use the `ALLOCATE` statement to assign the actual bounds to the arrays and matrices.
- The best-fit cubic model data are to be written to an output file for x values $0\,(0.01)\,1$.
- The screen output produced by your program should be as follows:

```
Vector b:
   24.118000   13.234500    9.468365    ...

Matrix A:
   21.000000   10.500000    7.175000    ...
   10.500000    7.175000    5.512500    ...
    7.175000    5.512500    4.516663    ...
    5.512500    4.516663    3.854156    ...

Inverse matrix A^{-1}:
    0.544011   -3.960025    7.715984    ...
   -3.960029   42.864491  -97.344635    ...
    7.716001  -97.344681  238.249985    ...
   -4.391153   60.141766 -154.079041    ...

Solution vector (a_1,a_2,a_3,a_4):
    0.574694        ...       ...       ...
```

- Plot the original data and the best-fit cubic model data, as shown in Fig. 6.

The program structure shown below should help you to write your code.

```
  PROGRAM CubicFit

  IMPLICIT none


! Date I/O
      OPEN(unit=10, file='nonlinear.dat', status='old')
      OPEN(unit=20, file='CubicLeastSquaresFit.dat', status='unknown')
```

```
! Determine how much space needs to be allocated for arrays and matrices


! Determine actual bounds for arrays and matrices


! Read data from data file


! Compute all sums


! Compute all elements of matrix A


! Compute all elements of vector b


! Print a header and elements of vector b


! Print a header and the original matrix


! Compute inverse matrix C = A^{-1}
      CALL INVERSE(A,C,n)


! Print a header and the inverse matrix C


! Compute all components of the solution vector a


! Print a header and all elements of the solution vector


! Write best-fit data to output file
      x_A =0.0; x_B=1.0
      Nint= 100
      DO i=1,Nint
         x_i = x_A + (x_B - x_A)/float(Nint) * float(i)
         f_i = a(1) + a(2)*x_i + a(3)*x_i**2 + a(4)*x_i**3
         WRITE(20,*) x_i, f_i
      END DO

! Release arrays and matrice from memory


! Close access to data files

END PROGRAM CubicFit
```