Computational Techniques and Numerical Calculations

Master Studies in Theoretical Chemistry and Computational Modelling

Numerical Linear Algebra

Evaluation exercises

Author:
José Antonio Quiñonero Gris

March 10, 2023

Part I

Polynomial regression analysis

In this part, a program has been written to perform a least square regression analysis to fit a set of points to a polynomial of degree m solving a linear system by the Gauss-Jordan method.

1 Theoretical background

The least squares regression polynomial of degree m for the set of points $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ is given by

$$y = a_0 + a_1 x + a_2 x^2 + \ldots + a_{m-1} x^{m-1} + a_m x^m, \tag{1}$$

where the coefficients are determined by the following system of $m+1$ linear equations

$$\begin{aligned}
n a_0 + \left(\sum x_i\right) a_1 + \left(\sum x_i^2\right) a_2 + \cdots + \left(\sum x_i^m\right) a_m &= \sum y_i, && (2)\\
\left(\sum x_i\right) a_0 + \left(\sum x_i^2\right) a_1 + \left(\sum x_i^3\right) a_2 + \cdots + \left(\sum x_i^{m+1}\right) a_m &= \sum x_i y_i, && (3)\\
\left(\sum x_i^2\right) a_0 + \left(\sum x_i^3\right) a_1 + \left(\sum x_i^4\right) a_2 + \cdots + \left(\sum x_i^{m+2}\right) a_m &= \sum x_i^2 y_i, && (4)\\
&\vdots\\
\left(\sum x_i^m\right) a_0 + \left(\sum x_i^{m+1}\right) a_1 + \left(\sum x_i^{m+2}\right) a_2 + \cdots + \left(\sum x_i^{2m}\right) a_m &= \sum x_i^m y_i. && (5)
\end{aligned}$$

In order to solve the system of linear equations, it can be written as

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{6}$$

where $\mathbf{A}$ is the coefficients (square) matrix

$$\mathbf{A} = \begin{pmatrix}
n & (\sum x_i) & (\sum x_i^2) & \cdots & (\sum x_i^m)\\
(\sum x_i) & (\sum x_i^2) & (\sum x_i^3) & \cdots & (\sum x_i^{m+1})\\
(\sum x_i^2) & (\sum x_i^3) & (\sum x_i^4) & \cdots & (\sum x_i^{m+2})\\
\vdots & \vdots & \vdots & \ddots & \vdots\\
(\sum x_i^m) & (\sum x_i^{m+1}) & (\sum x_i^{m+2}) & \cdots & (\sum x_i^{2m})
\end{pmatrix}, \tag{7}$$

$\mathbf{x}$ is the variables (column) vector to solve for and $\mathbf{b}$ the right-hand side (column) vector

$$\mathbf{x} = \begin{pmatrix} a_0\\ a_1\\ a_2\\ \vdots\\ a_m \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} \sum y_i\\ \sum x_i y_i\\ \sum x_i^2 y_i\\ \vdots\\ \sum x_i^m y_i \end{pmatrix}. \tag{8}$$

Sticking the column vector $\mathbf{b}$ to the right of $\mathbf{A}$, the augmented matrix $\mathbf{M} = \mathbf{A} \sqcup \mathbf{b}$ is constructed

$$\mathbf{M} = \mathbf{A} \sqcup \mathbf{b} = \left(\begin{array}{ccccc|c} n & \left(\sum x_i\right) & \left(\sum x_i^2\right) & \cdots & \left(\sum x_i^m\right) & \sum y_i \\ \left(\sum x_i\right) & \left(\sum x_i^2\right) & \left(\sum x_i^3\right) & \cdots & \left(\sum x_i^{m+1}\right) & \sum x_i y_i \\ \left(\sum x_i^2\right) & \left(\sum x_i^3\right) & \left(\sum x_i^4\right) & \cdots & \left(\sum x_i^{m+2}\right) & \sum x_i^2 y_i \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \left(\sum x_i^m\right) & \left(\sum x_i^{m+1}\right) & \left(\sum x_i^{m+2}\right) & \cdots & \left(\sum x_i^{2m}\right) & \sum x_i^m y_i \end{array}\right) \tag{9}$$

Then, reducing $\mathbf{A}$ from the augmented matrix either to a triangular form (with Gauss elimination) of fully reduced to diagonal form (with Gauss-Jordan or backsubstitution), the system is solved (except the case that the system is singular).

The Gauss elimination method is based in two elementary facts (facts about interchanging columns are omitted as only partial pivoting is going to be used in this case):

- The interchange of any two rows of $\mathbf{A}$ and the corresponding rows of $\mathbf{b}$ does not change the solution for $\mathbf{x}$, as it corresponds to writing the same set of linear equations in a different order.
- Likewise, the solution set is unchanged if any row in $\mathbf{A}$ is replaced by a linear combination of itself and any other row, as long as the same linear combination is done for $\mathbf{b}$.

Then, Gauss elimination is, basically, a method to reduce a matrix to a triangular matrix, with diagonal elements equal to one, by performing row operations. Essentially, the steps for the Gauss elimination are:

1. Check that the element $a_{i,i}$ (pivot) is not null. In case it is, the row is interchanged with other row with element $a_{j,i}$ $(j > i)$ not null (*partial pivoting*). In this case, the row with largest (in magnitude) available element is picked as the pivot.
2. Divide the i-th row by the pivot a_{ii}, so it becomes 1, $a_{i,i} \to 1$ (normalisation)
3. Substract the right amount of the i-th row, $\alpha A(i,:)$, from each other row to make all remaining $a_{j,1} = 0$ $(j > i)$. The *right amount*, α, can be trivially found, as

$$a_{j,i} - \alpha a_{i,i} = 0 \implies \alpha = \frac{a_{j,i}}{a_{i,i}}. \tag{10}$$

   Then, the linear combination to apply is

$$A(j,:) \leftarrow A(j,:) - \frac{a_{j,i}}{a_{i,i}} A(i,:). \tag{11}$$

4. Repeat for the rest of the columns.
5. Lastly, once all columns are transformed, check if $a_{n,n} = 0$. If this is the case, the system is singular.

The algorithm used in this program, based in this scheme, is explained in detail in the Description of the program section.

Then, after applying Gauss elimination over $\mathbf{A} \sqcup \mathbf{b} \to \mathbf{A'} \sqcup \mathbf{b'}$, $\mathbf{A'}$ is triangular. Then, $\mathbf{x}$ from eq. (6) can be solved by backsubstitution. The last element of $\mathbf{x}$, x_n, is given by

$$x_n = \frac{b'_n}{a'_{n,n}}, \tag{12}$$

and the next, by

$$x_{n-1} = \frac{1}{a'_{n-1,n-1}} \left(b'_{n-1} - x_n a'_{n-1,n}\right). \tag{13}$$

Then, the solution to the set of equations, x_i, is given by

$$x_i = \frac{1}{a'_{i,i}} \left(b'_i - \sum_{j=i+1}^{n} x'_{i,j} x_j \right). \tag{14}$$

Also, the triangular shape matrix $\mathbf{A'}$ can be further reduced up to a diagonal form, so it already yields the solution to the system of linear equations. It can be achieved as the Gauss elimination, but also performing row operations over the elements above the pivot, i.e. repeating steps 1 and 3 of the Gauss elimination but considering all $j \neq i$, not only $j > i$. It is called Gauss-Jordan elimination.

The result of the Gauss-Jordan elimination is, therefore, a matrix $\mathbf{A'} \sqcup \mathbf{b'}$ where $\mathbf{A'}$ is diagonal and, then, the solutions for $\mathbf{x}$ are simply the new elements of $\mathbf{b'}$

$$x_i = b'_i, \quad i = 1, 2, \ldots m + 1. \tag{15}$$

Finally, in eq. (1), the fitted polynomial can be written as

$$y = f(x) = b'_1 + b'_2 x + b'_3 x^2 + \ldots + b'_m x^{m-1} + b'_{m+1} x^m, \qquad b'_i = x_i = (\mathbf{x})_i. \tag{16}$$

The quality of the fit can be checked calculating the coefficient of determination, R^2

$$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}, \tag{17}$$

where SS_{res} is the residual sum of squares

$$SS_{\text{res}} = \sum_{i=1}^{n} \left(y_i - f(x_i)\right)^2, \tag{18}$$

and SS_{tot} is the total sum of squares

$$SS_{\text{tot}} = \sum_{i=1}^{n} \left(y_i - \overline{y}\right)^2. \tag{19}$$

2 Description of the program

The program is structured in

- Main file: `main.f90`
- Module containing procedures for input/output purposes: `modules/io.f90`

- Module containing the procedures needed to perform Gauss elimination, backsubstitution and Gauss-Jordan elimination: `module/gauss_jordan.f90`
- Module containing the procedures needed for the regression analysis: `modules/polynomial_regression.f90`
- Data files: input file, `data(input.dat`, and output file `data/output.dat`
- Plots: `graph/` folder

Ignoring the `main.f90` file and `io.f90` module, which content is senseless to explain, the modules containing the needed algorithms consist of:

2.1 Gauss, Backsubstitution and Gauss-Jordan

The procedures needed to solve for the linear system are contained in the `module/gauss_jordan.f90` module. These procedures are: Gauss elimination (procedure GaussElimination), Gauss-Jordan elimination (procedure Jordan) and backsubstitution (procedure BackSubstitution).

- **Gauss elimination**: the algorithm basically follows the steps given in the Theoretical background section. The algorithm is algorithm 1.

Algorithm 1 Gauss elimination

```
1: procedure GaussElimination($\mathbf{M}$, $\varepsilon$)
       Input. Matrix to reduce, $\mathbf{M}$; tolerance for the method, $\varepsilon$
       Output. Reduced to triangular form matrix, $\mathbf{M'} \leftarrow \mathbf{M}$
2:     Get number of rows of input matrix $\mathbf{M}$, $n_\mathrm{r}$, and number of columns, $n_\mathrm{c}$
3:     for $i \leftarrow 1, n_\mathrm{r}$ do
4:         if $|M_{i,i}| < \varepsilon$ then
5:             Swap the $i$-th row, $M(i,:)$, with the $j$-th row which element $M(j,i)$ $(\forall j > i)$ is the largest absolute value of the $i$-th column, $M(i,k) \leftarrow |M(:,i)|$
6:             $t(:) \leftarrow M(i,:)$
7:             $M(i,:) \leftarrow M(k,:)$
8:             $M(k,:) \leftarrow t(:)$
9:         Make the pivot $M_{i,i} = 1 \leftarrow M(i,:)/M(i,i)$
10:        for $j \leftarrow i+1, n_\mathrm{r}$ do                ▷ Make $M_{j,i} = 0$ $(\forall j > i)$
11:            $M_{j,i} \leftarrow M(j,:) = M(j,:) - (M(j,i)/M(i,i))\, M(i,:)$
12:    if $|M(n_\mathrm{r}, n_\mathrm{c})| < \varepsilon$ then write The system is singular
```

- **Backsubstitution**: the algorithm is algorithm 2.
- **Gauss-Jordan elimination**: the algorithm is algorithm 3.

2.2 Augmented matrix, fitted polynomial, R^2 coefficient

The procedures needed to perform the regression analysis are contained in the `module/polynomial_regression.f90` module. These procedures are: creation of the augmented matrix (procedure AugmentedMatrix), function for the fitted polynomial (procedure FittedPolynomial) and calculation of the determination coefficient R^2 (procedure RCoeff).

Algorithm 2 Backsubstitution

```
1: procedure BACKSUBSTITUTION($\mathbf{A}$, $\mathbf{b}$, $\mathbf{x}$)
     Input. Triangular form coefficient matrix, $\mathbf{A}$; right-hand side coefficients column vector, $\mathbf{b}$
     Output. Solution column vector, $\mathbf{x}$
2:   Get number of rows of input matrix $\mathbf{A}$, $n_{\mathrm{r},A}$
3:   Get size of column vector $\mathbf{b}$, $n_{\mathrm{r},b}$
4:   if $n_{\mathrm{r},A} \neq n_{\mathrm{r},b}$ then stop
5:   Initialize $x\left(n_{\mathrm{r},A}\right)$
6:   for $i \leftarrow n_{\mathrm{r}}, 1, -1$ do    ▷ Compute elements of $\mathbf{x}$, $x_i = 1/a_{i,i}\left(b_i - \sum_{j=i+1}^{N} a_{i,j}x_j\right)$
7:     $s = 0$
8:     for $j \leftarrow i+1, n_{\mathrm{r}}$ do
9:       $s = s + a_{i,j}x_j$
10:    $x_i = 1/a_{i,i}\left(b_i - s\right)$
```

Algorithm 3 Gauss-Jordan elimination

```
1: procedure JORDAN($\mathbf{M}$, $\mathbf{x}$)
     Input. Augmented triangular matrix, $\mathbf{M} = \mathbf{A} \sqcup \mathbf{b}$
     Output. Solution column vector, $\mathbf{x}$
2:   Get number of rows of input matrix $\mathbf{M}$, $n_{\mathrm{r}}$, and columns $\mathbf{M}$, $n_{\mathrm{c}}$
3:   Initialize $x\left(n_{\mathrm{r}}\right)$
4:   for $i \leftarrow n_{\mathrm{r}}, 1, -1$ do    ▷ Make $\mathbf{A}$ diagonal
5:     for $j \leftarrow i-1, 1, -1$ do
6:       $M(j,:) \leftarrow M(j,:) - \left[M(j,i)/M(i,i)\right] M(i,:)$
7:   $\mathbf{x} \leftarrow M(:, n_{\mathrm{c}})$    ▷ Store coefficient vector $\mathbf{b}$ in the solution vector $\mathbf{x}$
```

- **Augmented matrix**: subroutine to create the augmented matrix by calculating all necessary sums for a polynomial of arbitrary degree, m, that fits into the data, inputted as the matrix $\mathbf{D}$. The creation of the augmented matrix from the set of (x, y) data is explained in algorithm 4.

Algorithm 4 Creation of the augmented matrix

```
1: procedure AUGMENTEDMATRIX($\mathbf{D}$, $m$, $\mathbf{M}$)
     Input. Data matrix, $\mathbf{D}$, such that $x \leftarrow D(:,1)$ and $y \leftarrow D(:,2)$; degree of the polynomial, $m$
     Output. Augmented matrix, $\mathbf{M}$
2:   Get number of data, number of rows of $\mathbf{D}$, $n$
3:   Initialize $M(m+1, m+1)$
4:   for $i \leftarrow 1, m+1$ do
5:     for $i \leftarrow 1, m+1$ do
6:       $M_{j,i} = \sum_{k=1}^{n} D_{j,1}^{i+j-2}$
7:       $M_{i,j} = M_{j,i}$
8:     $M_{i,m+2} = \sum_{k=1}^{n} D_{j,1}^{i-1} D_{j,2}$
```

- **Fitted polynomial**: the algorithm is algorithm 5.

- R^2 **determination coefficient**: the algorithm is algorithm 6.

Algorithm 5 Fitted polynomial function

```
1: function FITTEDPOLYNOMIAL($\mathbf{c}$, $x$)
       Input. Coefficients vector, $\mathbf{c}$; independent variable, $x$, at which the function is evaluated
       Output. Fitted polynomial, $f$, evaluated at $x$, $f(x)$
2:     Get number of coefficients vector, $m \leftarrow \text{size}(\mathbf{c})$
3:     Initialize $f = 0$
4:     for $i \leftarrow 1, m$ do
5:         $f = f + c_i x^{i-1}$
```

Algorithm 6 R^2 determination coefficient

```
1: procedure RCOEFF($\mathbf{x}$, $\mathbf{y}$, $\mathbf{c}$, $R^2$)
Require: FITTEDPOLYNOMIAL($\mathbf{c}$, $x$)
       Input. Data vectors, $\mathbf{x}$ and $\mathbf{y}$; polynomial coefficients vector, $\mathbf{c}$
       Output. Determination coefficient, $R^2$
2:     Get sizes of data vectors $n_x \leftarrow \text{size}(\mathbf{x})$, $n_y \leftarrow \text{size}(\mathbf{y})$,
3:     if $n_x \neq n_y$ then stop
4:     else$n \leftarrow n_x$
5:     Calculate mean value of $y$ data, $\overline{y} \leftarrow \left(\sum_{i=1}^{n} y_i\right)/n$
6:     Initialize sum of residual squares, $SS_{\text{res}} = 0$
7:     Initialize total sum of squares, $SS_{\text{tot}} = 0$
8:     for $i \leftarrow 1, n$ do
9:         $SS_{\text{res}} = SS_{\text{res}} + (y_i - f(\mathbf{c}, x_i))^2$    ▷ Calculate the residual sum of squares, $SS_{\text{res}}$
10:        $SS_{\text{tot}} = SS_{\text{tot}} + (y_i - \overline{y})^2$    ▷ Calculate total sum of squares, $SS_{\text{tot}}$
11:    Calculate $R^2 \leftarrow 1 - (SS_{\text{res}}/SS_{\text{tot}})$
```

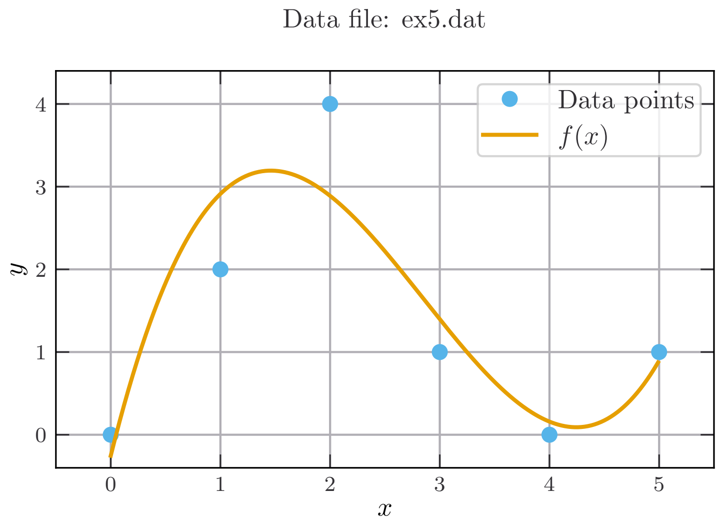

Figure 1: Data points and fitted polynomial of exercise 5.

3 Results

3.1 Exercise 5

The (x, y) data points are plotted along with the fitted polynomial in fig. 1.

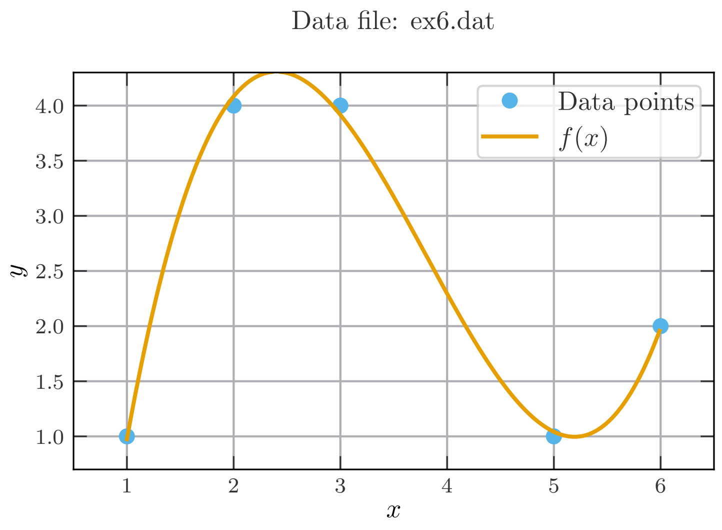

Figure 2: Data points and fitted polynomial of exercise 6.

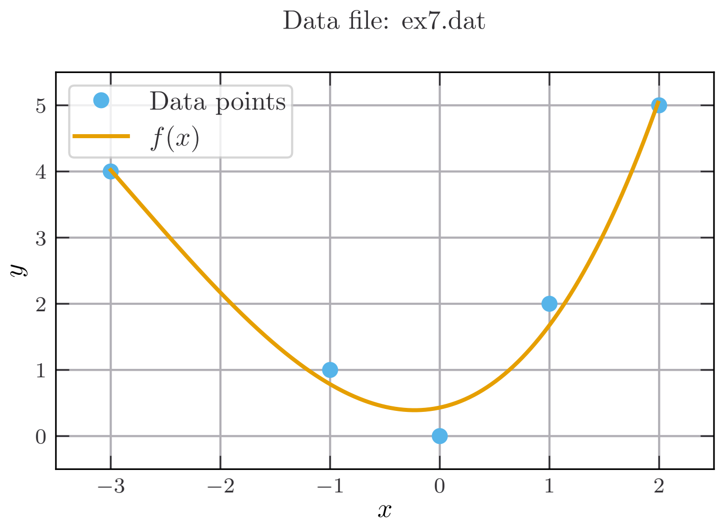

Figure 3: Data points and fitted polynomial of exercise 7.

3.2 Exercise 6

The (x, y) data points are plotted along with the fitted polynomial in fig. 2.

3.3 Exercise 7

The (x, y) data points are plotted along with the fitted polynomial in fig. 3.

3.4 Exercise 8

The (x, y) data points are plotted along with the fitted polynomial in fig. 4.

3.5 Exercise 9

The (x, y) data points are plotted along with the fitted polynomial in fig. 5.

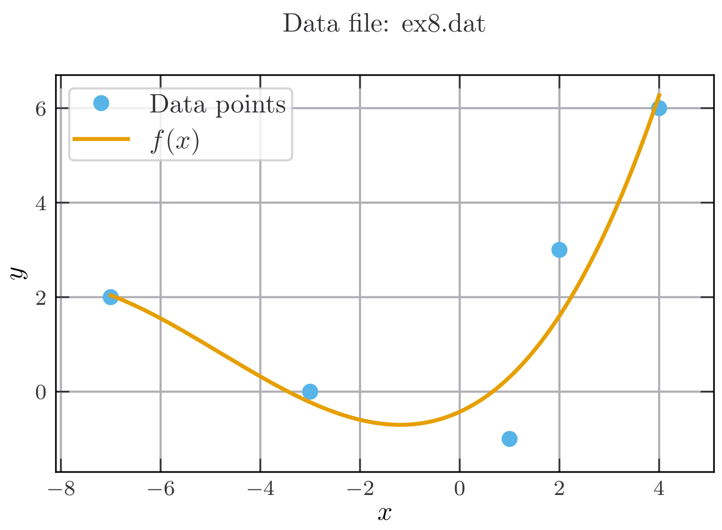

Figure 4: Data points and fitted polynomial of exercise 8.

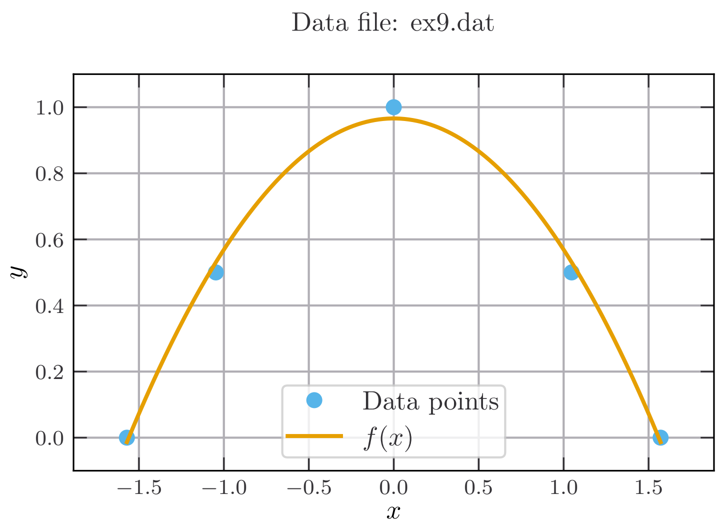

Figure 5: Data points and fitted polynomial of exercise 9.

Part II

4 Theoretical background

5 Description of the program

6 Results