Università della Svizzera italiana

Institute of Computing CI

Numerical Computing **2023**

Student: Hun Rim **Discussed with:** FULL NAME

Bonus assignment **Due date:** Wednesday, 22 November 2023, 11:59 PM

Numerical Computing 2023 — Submission Instructions
(Please, notice that following instructions are mandatory: submissions that don't comply with, won't be considered)

- Assignments must be submitted to iCorsi (i.e. in electronic format).
- Provide both executable package and sources (e.g. C/C++ files, MATLAB). If you are using libraries, please add them in the file. Sources must be organized in directories called:

  Project_number_lastname_firstname

  and the file must be called:

  project_number_lastname_firstname.zip
  project_number_lastname_firstname.pdf
- The TAs will grade your project by reviewing your project write-up, and looking at the implementation you attempted, and benchmarking your code's performance.
- You are allowed to discuss all questions with anyone you like; however: (i) your submission must list anyone you discussed problems with and (ii) you must write up your submission independently.

Exercise 1: Inconsistent systems of equations [10 points]

Consider the following inconsistent systems of equations:

(a) $A_1 x = b_1$, where

$$A_1 = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \end{bmatrix} \; b_1 = \begin{bmatrix} 5 \\ 2 \\ 4 \end{bmatrix}$$

(b) $A_2 x = b_2$, where

$$A_2 = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 0 & 1 \end{bmatrix} \; b_2 = \begin{bmatrix} 2 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

Find the least squares solution x^* and compute the Euclidean norm of the residual, SE and RMSE.

solution:
Least Square solution x^* can be obtained by solving the following equation:

$$A^T A x = A^T b \tag{1}$$

Then from the x^* obtained, we can get the residual vector as following and from it, we can calculate the Euclidean norm and proceed to SE (Standard Error) and RMSE (Root Mean Squared Error):

$$r = Ax^* - b \tag{2}$$

$$EuclideanNorm = ||r||_2 \tag{3}$$

$$SE = ||r||_2^2 \tag{4}$$

$$RMSE = \sqrt{\frac{SE}{m}} \tag{5}$$

Where m is the number of rows in residual vector.

(a)

$$\begin{bmatrix}1 & 1 & 1\\0 & 0 & 0\end{bmatrix}\begin{bmatrix}1 & 0\\1 & 0\\1 & 0\end{bmatrix}x = \begin{bmatrix}1 & 1 & 1\\0 & 0 & 0\end{bmatrix}\begin{bmatrix}5\\2\\4\end{bmatrix}$$

$$\begin{bmatrix}3 & 0\\0 & 0\end{bmatrix}x = \begin{bmatrix}11\\0\end{bmatrix}$$

$$x^* = \begin{bmatrix}3.6667\\0\end{bmatrix}$$

$$r^* = \begin{bmatrix}1 & 0\\1 & 0\\1 & 0\end{bmatrix}\begin{bmatrix}3.6667\\0\end{bmatrix} - \begin{bmatrix}5\\2\\4\end{bmatrix} = \begin{bmatrix}-1.3333\\1.6667\\-0.3333\end{bmatrix}$$

$$EuclideanNorm = ||r||_2 = \sqrt{(-1.3333)^2 + (1.6667)^2 + (-0.3333)^2} = \sqrt{4.6667} \approx 2.1602$$

$$SE = ||r||_2^2 = (-1.3333)^2 + (1.6667)^2 + (-0.3333)^2 \approx 4.6667$$

$$RMSE = \sqrt{\frac{||r||_2^2}{m}} \approx \sqrt{\frac{4.6667}{3}} \approx 1.2472$$

(b)

$$\begin{bmatrix}1 & 0 & 1 & 1\\1 & 1 & 2 & 0\\0 & 1 & 1 & 1\end{bmatrix}\begin{bmatrix}1 & 1 & 0\\0 & 1 & 1\\1 & 2 & 1\\1 & 0 & 1\end{bmatrix}x = \begin{bmatrix}1 & 0 & 1 & 1\\1 & 1 & 2 & 0\\0 & 1 & 1 & 1\end{bmatrix}\begin{bmatrix}2\\2\\3\\4\end{bmatrix}$$

$$\begin{bmatrix}3 & 3 & 2\\3 & 6 & 3\\2 & 3 & 3\end{bmatrix}x = \begin{bmatrix}9\\10\\9\end{bmatrix}$$

If we re-arrange the formula for x^* we get:

$$x^* = (A_2^T A_2)^{-1} A_2^T b_2 \tag{6}$$

and the resulting x^* will be

$$x^* \approx \begin{bmatrix}2\\-0.3333\\2\end{bmatrix}$$

$$r = \begin{bmatrix}1 & 1 & 0\\0 & 1 & 1\\1 & 2 & 1\\1 & 0 & 1\end{bmatrix}\begin{bmatrix}2\\-0.3333\\2\end{bmatrix} - \begin{bmatrix}2\\2\\3\\4\end{bmatrix} \approx \begin{bmatrix}-0.3333\\-0.3333\\0.3333\\0\end{bmatrix}$$

$$EuclideanNorm = ||r||_2 \approx \sqrt{(-0.3333)^2 + (-0.3333)^2 + (0.3333)^2 + (0)^2} \approx \sqrt{0.3333} \approx 0.5774$$

$$SE = ||r||_2^2 = (-0.3333)^2 + (-0.3333)^2 + (0.3333)^2 + (0)^2 \approx 0.3333$$

$$RMSE = \sqrt{\frac{||r||_2^2}{m}} \approx \sqrt{\frac{0.3333}{4}} \approx 0.2887$$

Exercise 2: Polynomials models for least squares [20 points]

(a) Write *leastSquare.m* function which calculates least squares x^*, euclidean norm, SE and RMSE of a matrix A and vector b, and write a script $ex2a.m$ which computes the result of exercise 1.

```
function [x, EuclideanNorm, SE, RMSE] = leastSquares(A, b)
  x = (A' * A) \ A' * b;
  r = A * x - b;
  EuclideanNorm = norm(r);
  SE = EuclideanNorm ^ 2;
  MSE = SE / length(b);
  RMSE = sqrt(MSE);
end
```

The code above calculates least squares (*leastSquare.m*) using matlab library function and Euclidean norm is calculated by using the matlab norm function after calculating the residual vector. Standard Errors (SE) are calculated by directly squaring the Euclidean norm, and before the Root Mean Squared Error (RMSE) is calculated, the Mean Squared Error is calculated through dividing SE by length of vector b.

```
A_1 = [1, 0; 1, 0; 1, 0];
b_1 = [5; 2; 4];
A_2 = [1, 1, 0; 0, 1, 1; 1, 2, 1; 1, 0, 1];
b_2 = [2; 2; 3; 4];

[x_1, norm1, SE1, RMSE1] = leastSquares(A_1, b_1);
[x_2, norm2, SE2, RMSE2] = leastSquares(A_2, b_2);
```

This particular code ($ex2a.m$) defines the values of matrix A and vector b as given in the exercise 1, and it calculates and stores the least squares, Euclidean Norm, SE, RMSE using function in *leastSquare.m* as described above. Result of the calculation via the script is same as the calculation by hand for (b) of exercise 1, but it differs for the (a). Reason behind this is because script calculation solves for least squares through utilizing inverse like shown in equation 6, whereas when calculating by hand, an algebric approach is taken. As determinant of x is equal to 0, script generates NaN and Euclidean Norm, SE, RMSE which are calculated from it also generates NaN. However, algebric approach evades this problem, hence, it produces normal result. A solution to this problem would be checking if least squares generated are NaN and if true, we use built-in function such as $pinv(A) * b$ and otherwise we use the old inverse calculation approach.

Debugged code is as following:

```
determinant = det(A' * A);
if (determinant == 0)
    x = pinv(A) * b;
else
    x = (A' * A) \ A' * b;
end
r = A * x - b;
EuclideanNorm = norm(r);
SE = EuclideanNorm ^ 2;
MSE = SE / length(b);
RMSE = sqrt(MSE);
```

(b) Consider the linear model $y_i = \alpha_1 + \alpha_2 x_i$ and apply it to the crude oil and kerosene production data in the period 1980-2011. Write a script *linearModel.m* in which you use *leastSquares*() to compute the least squares solution x^* and the metrics of the residual. For each dataset, create a figure in which you plot the original data points and the linear model.

```
[year, production, ~] = readData(filePath);

duration = 2011 - 1980 + 1;

y = str2double(production);

x = [ones(duration, 1), year(:)];

[factors, ~, ~, ~] = leastSquares(x, y);

z = factors(1) + factors(2) * year(:); %based on y = mx + b;

% plot the line and scatter graph...
```

The code above simply follows the process of reading the data from the file, then converting 'x' and 'y' to viable format of matrix A and vector b, and recaclulating the linear model (line graph) according to the least squares returned and plotting them with the origin (scatter graph). During conversion, columns of the matrix A filled with $x_i^{columnNumber}$ (columnNumber starts from 0, and there are only 2 columns) and 'y' is placed as vector b.

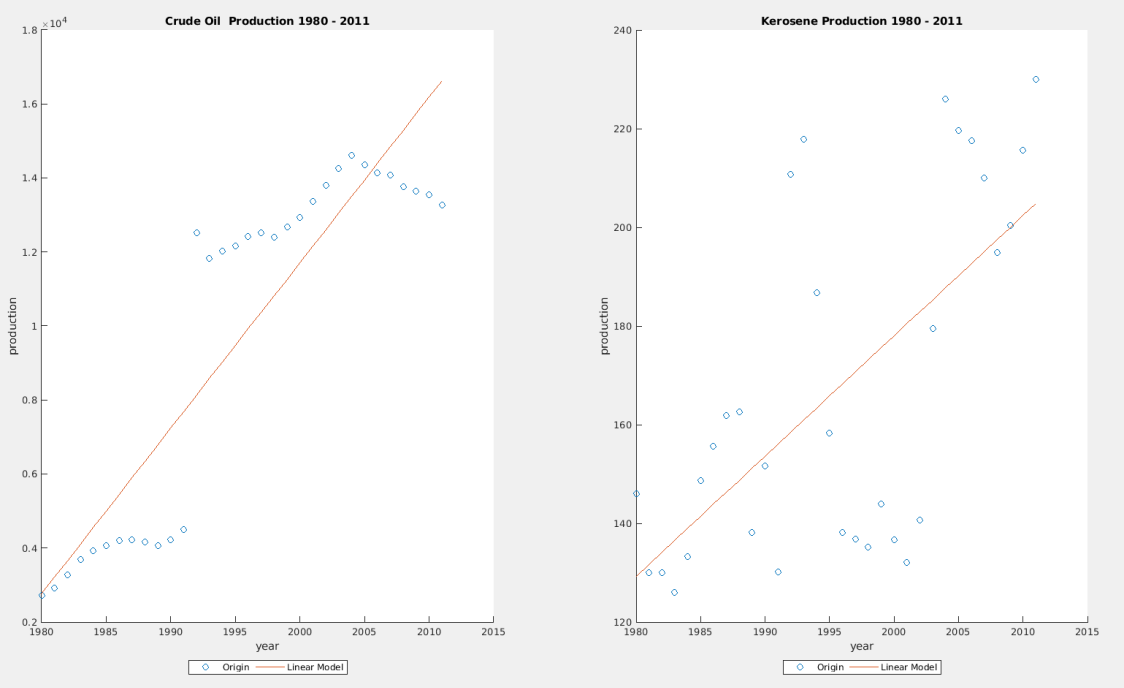

Figure 1: Linear Model and original data of crude oil and kerosene production from 1980 to 2011

(c) Consider the quadratic model $y_i = \alpha_1 + \alpha_2 x_i + \alpha_3 x_i^2$ and apply it to the crude oil and kerosene production data in the period 1980-2011. Write a script *quadraticModel.m* in which you use

leastSquares() to compute the least squares solution x^* and the metrics of the residual. For each dataset, create a figure in which you plot the original data points and the quadratic model.

The general flow of the logic is very similar, but the matrix A is extended via the same logic so it would produce bigger least square vector (there are 3 columns now). Then instead of $y_i = mx_i + b$, quadratic modelling equation $y_i = a + bx_i + cx_i^2$ is used.

```
same as before ...
x = [ones(duration, 1), year(:), year(:).^2];

[factors, ~, ~, ~] = leastSquares(x, y);

z = factors(1) + factors(2) * year(:) + factors(3) * year(:).^2;
same as before ...
```

As a result of that change, resulting graph (quadratic) became more fitting than the previous linear graph shown in figure 1.

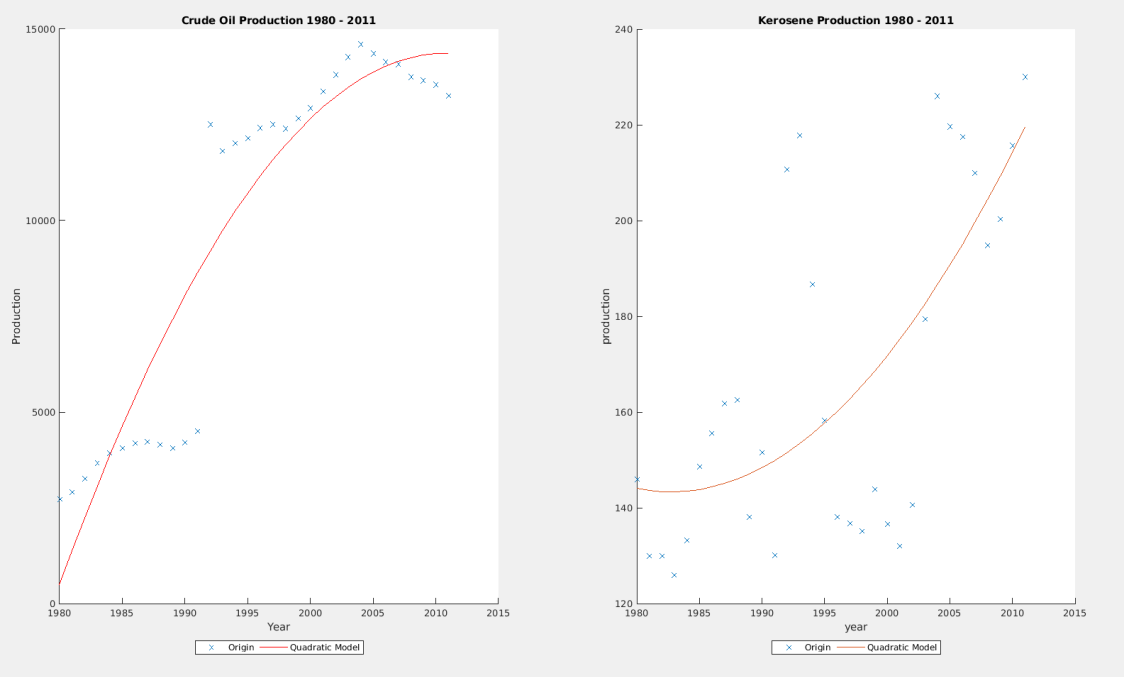

Figure 2: Quadratic Model and original data of crude oil and kerosene production from 1980 to 2011

(d) Consider the cubic model $y_i = \alpha_1 + \alpha_2 x_i + \alpha_3 x_i^2 + \alpha_4 x_i^3$ and apply it to the crude oil and kerosene production data in the period 1980-2011. Write a script *cubicModel.m* in which you use *leastSquares()* to com- pute the least squares solution x^* and the metrics of the residual. For each dataset, create a figure in which you plot the original data points and the cubic model.

We can apply the same approach as (c) and just increase the size of least squares vector and change the slope calculation from $y_i = a + bx_i + cx_i^2$, to $y_i = a + bx_i + cx_i^2 + dx_i^3$

Code can be altered as following:

```
x = [ones(duration, 1), year(1:duration),
                        year(1:duration).^2,
                        year(1:duration).^3];

[factors, ~, ~, ~] = leastSquares(x, y(1:duration));

z = factors(1) + factors(2) * year(1:duration)
               + factors(3) * year(1:duration).^2
               + factors(4) * year(1:duration).^3;
```

Finally, the resultant graph would be more fitted to the original data points than other approaches. Especially for the crude oil production.

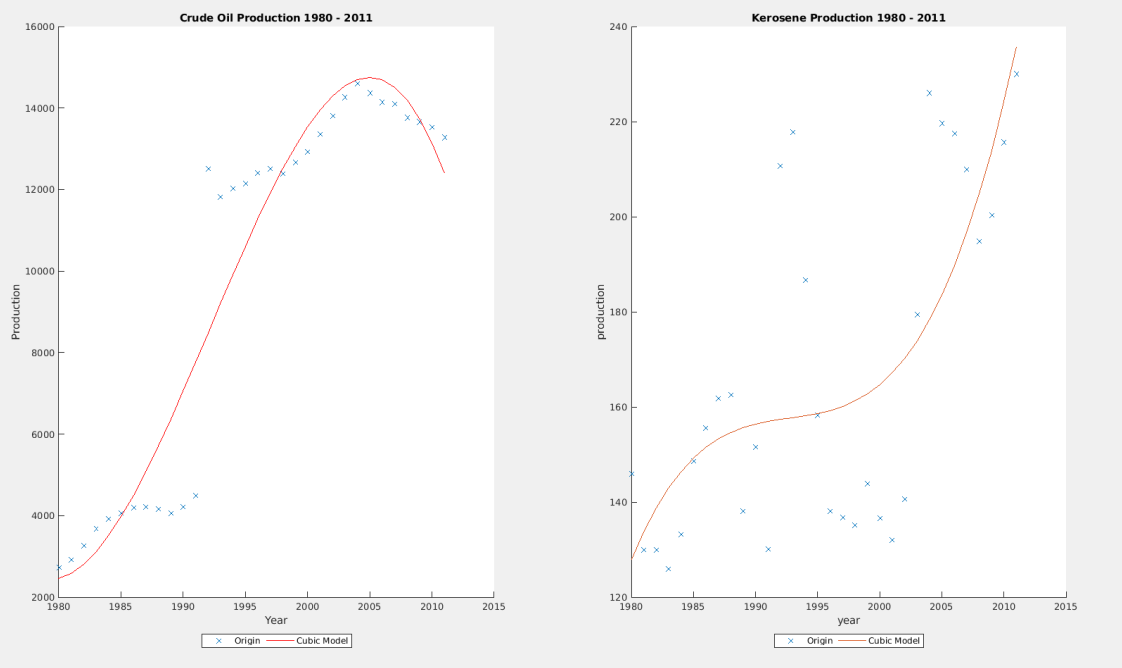

Figure 3: Cubic Model and original data of crude oil and kerosene production from 1980 to 2011

(e) Compare the linear, quadratic and cubic models on the basis of the quality metrics computed above, by creating a table containing the results for the two models. Which one of the three models would you pick for the crude oil data? And for the kerosene? Provide an estimate of the crude oil and kerosene production in 2012 by using the three models and compare the values obtained with the real values reported in the data source. Comment on your results.

Model Type	EuclideanNorm	Standard Error	RMSE
Linear Crude	11,471	131,590,000	2,027.8
Quadratic Crude	9,582.6	91,827,000	1,694
Cubic Crude	7,917.9	62,692,000	1,399.7
Linear Kerosene	151.0222	22,808	26.6972
Quadratic Kerosene	145.3013	21,112	25.6859
Cubic Kerosene	138.476	19,175	24.4793

Energy Type	Linear Estimation	Quadratic Estimation	Cubic Estimation	Real Data
Crude Oil	17,082(30.3%)	14,344(+9.4%)	11,473 (-12.5%)	13,111
Kerosene	207.2856(-23.6%)	225.1621 (-16.9%)	248.4740 (-7.3%)	267.89

To make a simple analysis, having a smaller Euclidean Norm suggests better fit of the model to the original data as smaller euclidean norm indicates smaller residuals, and naturally bigger would indicate the opposite. Similarly, Standard Error (SE) and Root Mean Squared Error (RMSE) indicates better fit of the model when they are small.

Under this assumption, the data on the table suggests for all the cases, cubic model is the best, quadratic model is the second, and linear model is the worst model to fit the original data.

However, the prediction made for 2012 from these models suggest, even though cubic model proved to be the best for kerosene production trend prediction, in terms of prediction for crude oil production, quadratic model showed more accurate result. However, prediction of a singular value is not enough to make a conclusion that quadratic model will always perform better for kerosene production prediction or cubic model will always perform better for crude oil production prediction as the 2012 data might have been just a singular anormaly in the data.

To get a more concrete conclusion, we would need more historical data and also make more predictions based on the obtained model. However, statistically it is undeniable that cubic model will show the best predictions.

Exercise 3: Analysis of periodic data [20 points]

The file *temperature.txt* contains the area mean-temperatures of Switzerland between January 1864 and March 2021 included. Temperature data exhibit a periodic behaviour and we will try to capture it by using periodic models. You will need the function *leastSquares()* implemented in Exercise 2.

(a) Consider the periodic model $y_i = \alpha_1 + \alpha_2 cos(2\pi x_i) + \alpha_3 sin(2\pi x_i)$ and apply it to the temperature data: (I) between January 1960 and January 1963; (II) between January 1960 and January 1970. Write a script *periodicA.m* in which you compute the least squares solutions and the metrics of the residual, and plot the outputs of the model against the original data in both cases.

```
[x1, y1] = readTemperatureData(filepath, startYear, endYear);
x = [ones(length(x1), 1), cos(2 * pi * x1), sin(2 * pi * x1)];
[factors, ~, ~, ~] = leastSquares(x, y1);
h = height(x1);

z = factors(1) +
    factors(2) * cos(2 * pi * x1(1:h)) +
    factors(3) * sin(2 * pi * x1(1:h));
generate graphs...
```

The *temperature.txt* file is read in a bit of a different way as the structure of the data is different. It can be seen in *readTenoeratureData.m* file. Then the x-axis data is converted into the periodic model format given in the question. Then the residual vector is calculated using *leastSquares.m* which was completed in the previous question. Then periodic model graph is calculated and graphed along with original data points.

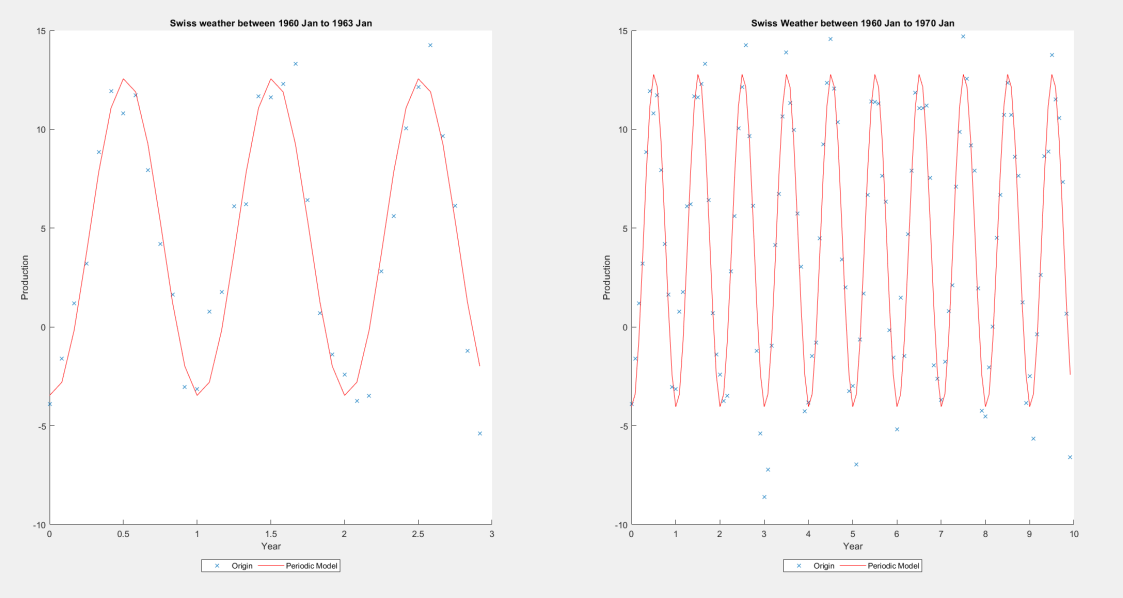

Figure 4: Periodic model of swiss weather 1960 Jan to 1963 Jan (left) and 1960 Jan to 1970 Jan (right)

Exercise 4: Data linearization and Levenberg-Marquardt method for the exponential model [20 points]

Exercise 5: Tikhonov regularization [15 points]