Università della Svizzera italiana | Institute of Computing CI

Numerical Computing **2023**

Student: Hun Rim **Discussed with:** Georgy Batyrev

Bonus assignment

Due date: Wednesday, 22 November 2023, 11:59 PM

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.

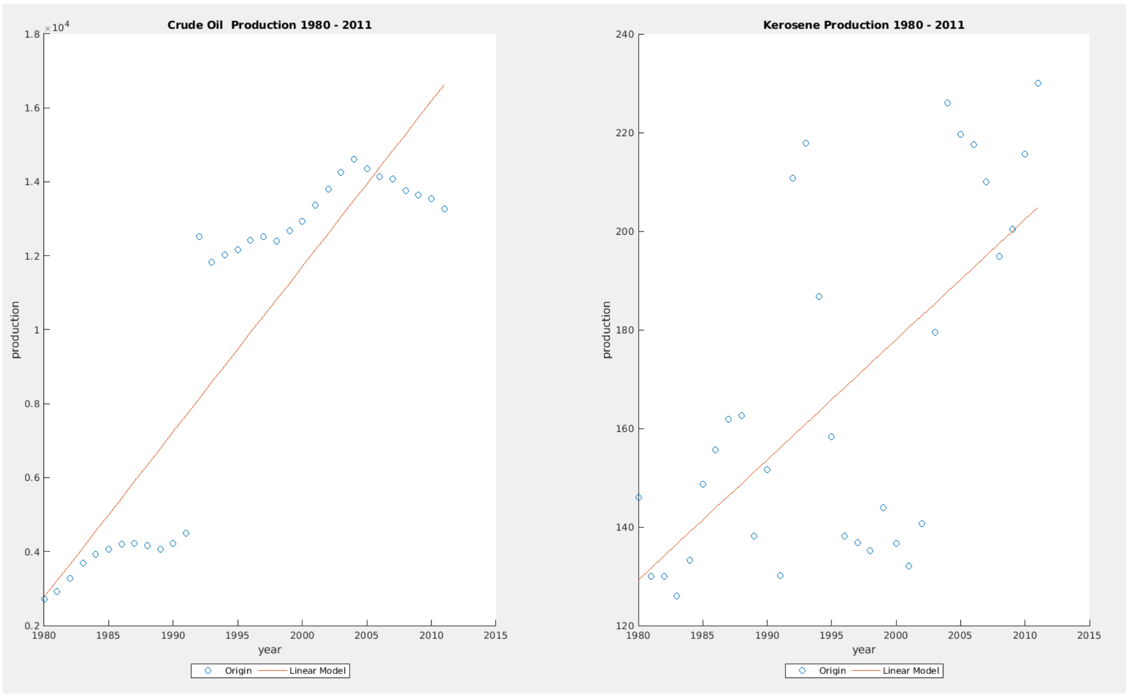

Figure 1: Linear Model and original data of crude oil and kerosene production from 1980 to 2011

(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.

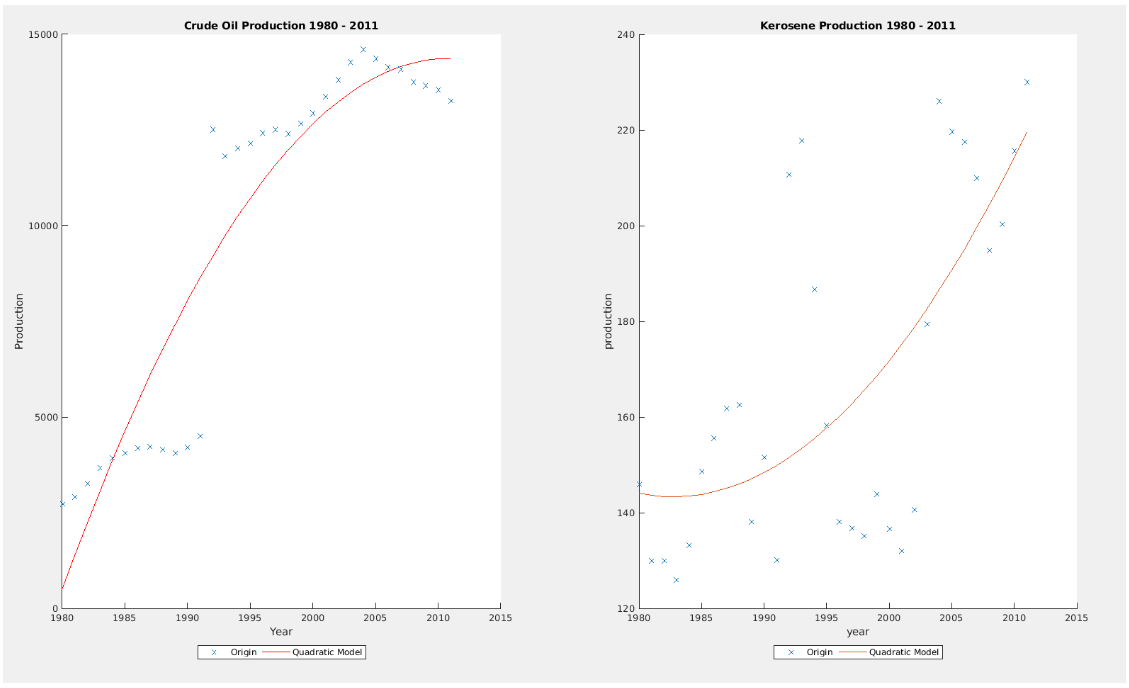

Figure 2: Quadratic Model and original data of crude oil and kerosene production from 1980 to 2011

(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.

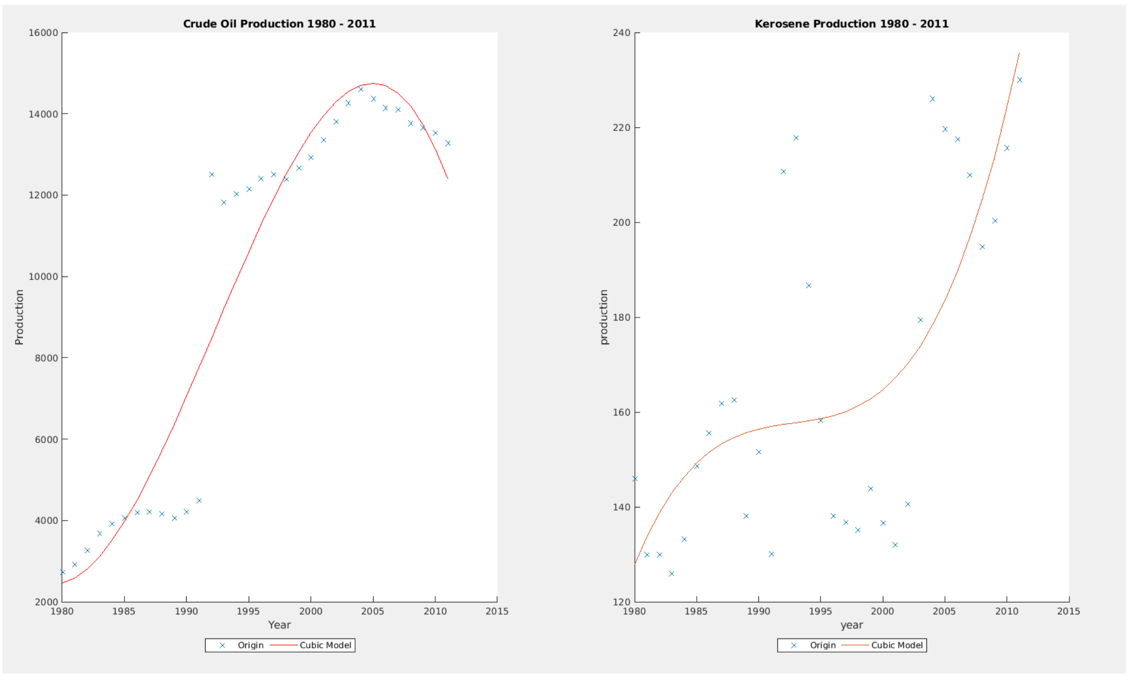

Figure 3: Cubic Model and original data of crude oil and kerosene production from 1980 to 2011

(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. The resulting graph is as following:

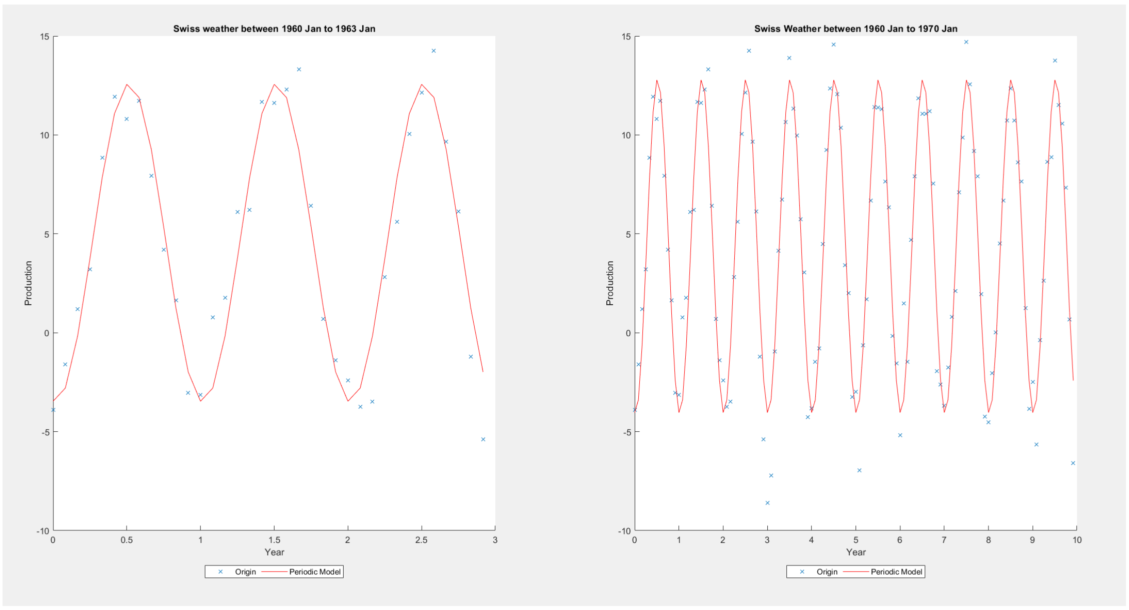

Figure 4: Periodic model of swiss weather 1960 Jan to 1963 Jan (left) and 1960 Jan to 1970 Jan (right)

(b) Repeat the same analysis and plots of the previous point for both time series, by using the periodic model $y_i = \alpha_1 + \alpha_2 cos(2\pi x_i) + \alpha_3 sin(2\pi x_i) + \alpha_4 cos(4\pi x_i)$ in the script *periodicB.m*.

Only things that need to be added are as following:

```
...
x = [ones(length(x1), 1), cos(2 * pi * x1),
      sin(2 * pi * x1), cos(4 * pi * x1)];
...
z = factors(1) +
    factors(2) * cos(2 * pi * x1(1:h)) +
    factors(3) * sin(2 * pi * x1(1:h)) +
    factors(4) * cos(4 * pi * x1(1:h));
...
```

Resulting graph is as following:

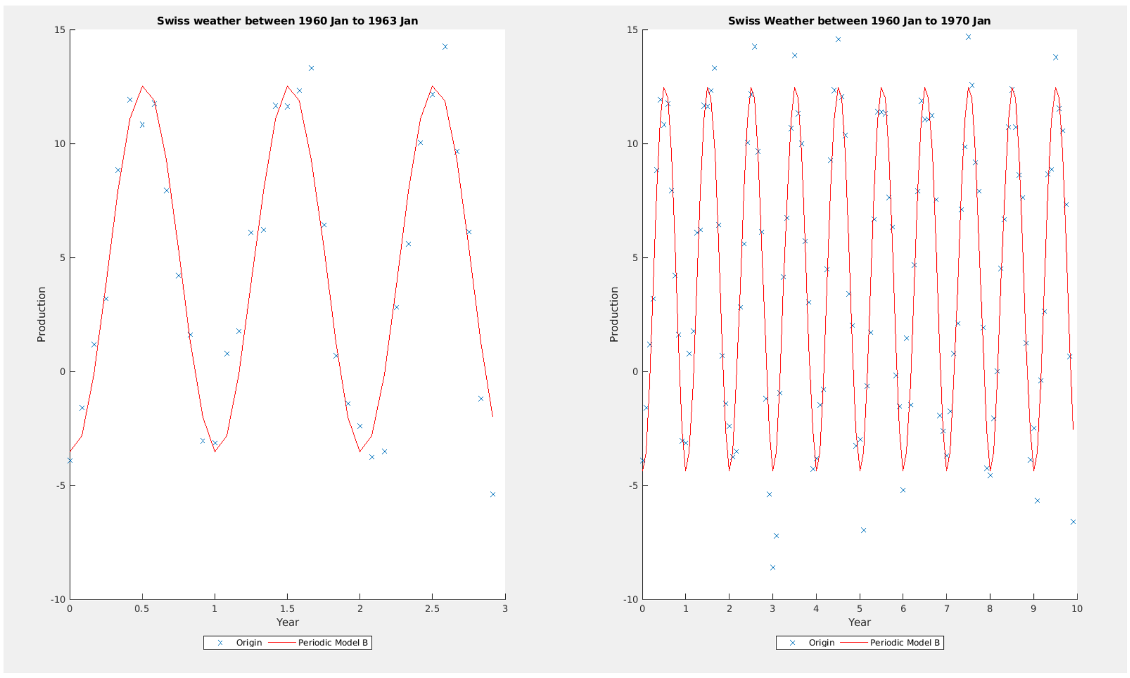

Figure 5: Periodic model 'B' of swiss weather 1960 Jan to 1963 Jan (left) and 1960 Jan to 1970 Jan (right)

(c) Compare the models of point (*a*) and (*b*). Was it beneficial to include more data? Which model would you prefer? Are you satisfied with the results obtained? If necessary, what would you suggest to improve your models? Motivate your answers.

Model Type	EuclideanNorm	Standard Error	RMSE
Periodic A short	10.0838	101.6826	1.6806
Periodic A long	18.0011	324.0413	1.6433
Periodic B short	10.0807	101.6201	1.6801
Periodic B long	17.8216	317.6092	1.6269

From the graph generated alone, it becomes very obvious that there is barely any difference between the two models. The negligence of 2 model's difference becomes more apprent through the display of quality metrics through a table. Naturally, the necessity to add more data becomes also negligible and almost pointless. Hence, periodic model A will be more preferable as it would be easier to calculate with lower time complexity due to smaller input and residual vector resulting from it.

Generally, it seems like periodic models are fitting the original datapoints quite well. However, if there has To make an improvement to it, maybe values other than 2π can be used with more data to increase accuracy, for example, π.

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

The file *nuclear.txt* contains the data on the nuclear electric power consumption by year in China in the period 1999-2006. We consider the power law model, expressed as:

$$y_i = \alpha_i x_i^{\alpha_2} \tag{7}$$

(a) Find the least squares best fit by using data linearization and compute the RMSE both of the log-linearized model and of the original exponential model. Include in your report all the computations and the necessary steps, as explained in the slides of the tutorial.

Log-Linearization can be done using the power law model shown in the beginning of the question. To convert it into linear function, $ln()$ can be applied to both of the sides, then due to the properties of logarithms, the power law model can be expressed as following:

$$ln(y_i) = ln(\alpha_1) + \alpha_2 ln(x_i) \tag{8}$$

For original exponential model, it can be expressed as the following:

$$y_i = \alpha_i e^{\alpha_2 x_i} \tag{9}$$

Which then can be converted applying $ln()$ to both sides again and get the following linear expression:

$$ln(y_i) = ln(\alpha_1) + \alpha_2 x_i \tag{10}$$

Equation 8 and 10 can be used to build the original exponential model and the log-linearization model using the previously defined $leastSquare$ function. Important factor to remember during the implementation would be that the calculated result of the model would be equivalent to $ln(y_i)$ not the y_i, hence, the result would have to be used as the power of e and not be graphed directly in relation to the original data.

```
% Getting x = year and y = consumption data.
[year, consumption, change] = readData('../data/nuclear.txt');

% Manipulate x-vals so it starts from 1
xaxis = year - min(year) + 1;
% Convert x and y values into log forms
logX = log(xaxis);
logY = log(consumption);

h = height(logX);

% Create a coefficient matrix with 1s and log form of x values to
% pas it onto the leastSquare calculation to get 2d residual
% vector (alphas) according to eq8
X = [ones(h, 1), logX];

% Call least square function with predefined matrix and log form of Y
% to obtain residual vectors (factors) and RMSE for this model
[factors, ~, ~, rmse_log] = leastSquares(X, logY);

% Calculate the log-linearization model fitting the given data
% We are taking the logX as multiplicand, the same form as the
% coefficient passed for leastSquares function
% the result is powered by 'e' because the result without it will
% be equivalent to 'y' values used in least square function
% which were logged.
z = exp(factors(1) + factors(2) * logX);

... print the residual vector and the rmse of the model ...

% Just in case, re-calculate the log for of the y-axis
logYO = log(consumption);

hO = height(xaxis);

% Create a coefficient matrix with 1s and x-axis values
% to get 2d residual vector (alphas) for original exponential model
XO = [ones(hO, 1), xaxis];

% Calculate the residual vector (factorsO) and RMSE value
% (rmse_original)
% of the original exponential method through leastSquares function
[factorsO, ~, ~, rmse_original] = leastSquares(XO, logYO);

% Calculate original exponential model values fitting to this data
% exp() function called for same reason as calculation of 'z'
zO = exp(factorsO(1) + factorsO(2) * xaxis);

... print the residual vector and the rmse of the model ...
```

The full script can be viewed at *ex4b.m* in *src* folder.

	Original Exponential Model	Log-Linearization Model
RMSE	5.4672	5.0953
α_1	2.3554	2.3705
α_2	0.2257	0.7549

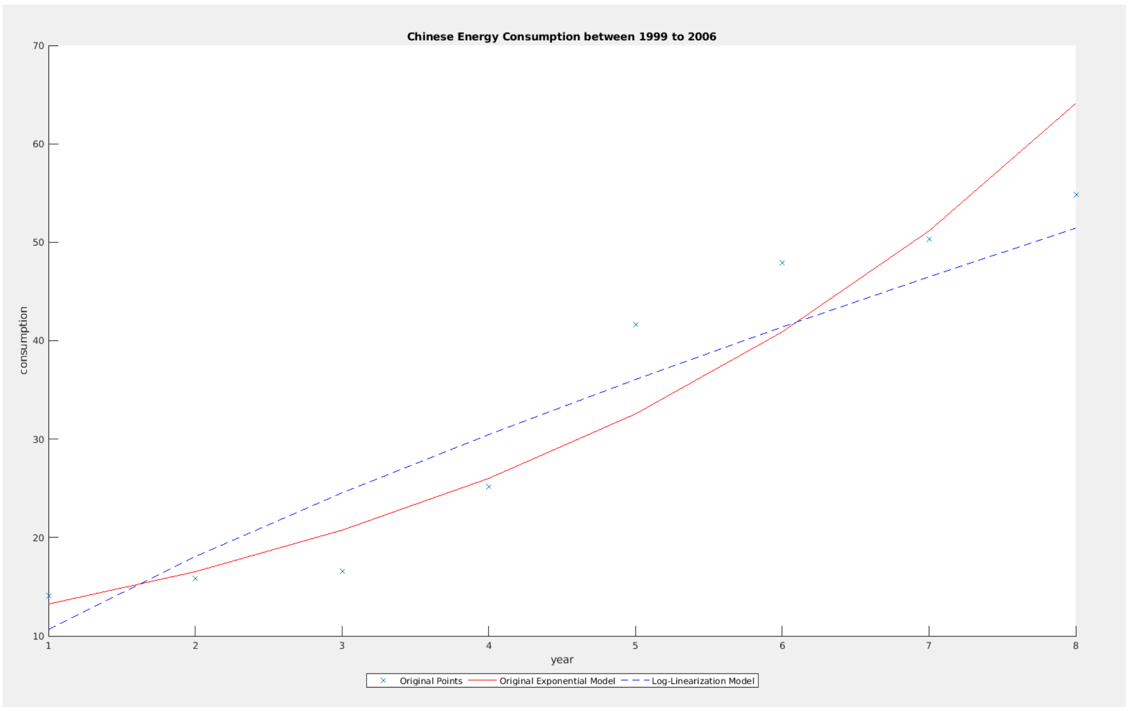

Figure 6: Original exponential model and Log-linearization model with original data of Chinese electricity consumption between 1999 to 2006

(b) Write a function *levenbergMarquardt()* in which you implement the Levenberg-Marquardt algorithm for solving nonlinear least squares problems. Following again the slides of the tutorial, show how you can formulate the problem in order to solve it with Levenberg-Marquardt method and compute analytically all the necessary quantities. Finally, write a script *ex4b.m* in which you use the function *levenbergMarquardt()* to fit the data points and compute the RMSE.

Logic behind the Levenberg Marquardt algorithm is an alteration on Gauss-Newton method using λ (Gaussian-Newton method is equivalent to Levenberg Marquardt algorithm when $\lambda = 0$). The difference is that this algorithm tackles eventual ill-conditioning of Jacobian matrix by introducing a regularization.

```
Given a random initial guess x^0 and lambda
    for i = 0, 1, 2, . . . do
        A = Jr(x(i))
        v(i) = (-1) * (A^T * A + lambda * diag(A^T * A))^(-1)
                * A^T * r(x(i))
        x(i+1) = x(i) + v(i)
        if (norm(v(i)) < tolerance) {
            return 1; // exit with success
        }
    end for
    return 0; exit with failure
```

Important aspect of implementing this algorithm is the definition of Jacobian matrix and calculation of residuals ($r(x_i)$). Jacobian matrix can be expressed as following for the power law model which is our basis equation:

$$J_r(\alpha_1, \alpha_2) = \begin{bmatrix} x_1^{\alpha_2} & \alpha_1 x_1^{\alpha_2} log(x_1) \\ ... & ... \\ x_i^{\alpha_2} & \alpha_1 x_i^{\alpha_2} log(x_i) \end{bmatrix} \tag{11}$$

Residual calculation can be expressed as following for the power law model:

$$r(\alpha_1, \alpha_2) = \begin{bmatrix} \alpha_1 x_1^{\alpha_2} - y_1 \\ ... \\ \alpha_1 x_i^{\alpha_2} - y_i \end{bmatrix} \tag{12}$$

With these factors in mind, the *levenbergMarquardt()* is implemented in *levenbergMarquardt.m* script and resultant graph is plotted with other least squares best fit model graphs. The result is as following:

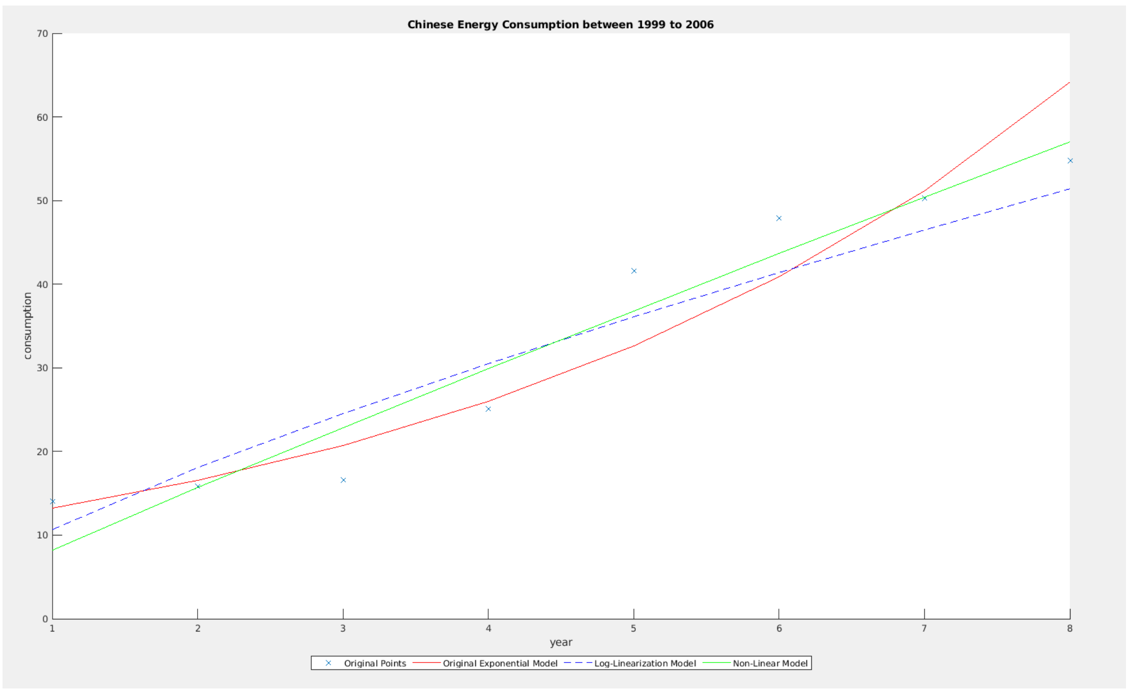

Figure 7: Original exponential model and Log-linearization model and non-linear model with original data of Chinese electricity consumption between 1999 to 2006

For calculation of non-linear model, there are various parameters that needs to be passed.

```
function [estimatedAlphas, residuals, rmse, exitFlag] =
levenbergMarquardt(X, Y, initialGuess, maxIterations, tolerance, lambda)
% X -> Original X-axis vector
% Y -> Original Y-axis vector
% initialGuess -> Intuition based initial value of alphas
% maxIterations -> Iteration limit
% tolerance -> epsilon value
% lambda -> regularization term
```

(c) Compare the results obtained in points (a) and (b), by extending the script *ex4b.m* to produce a plot of the original data points together with the two models. Which model would you choose?

The root mean square error (RMSE) for Original Exponential Model and Log-Linearization Model were calculated before which were 5.4672 and 5.0953. Newly calculated RMSE for Non-Linear Model using Levenberg Marquardt is 4.2394. Like explained before, lower RMSE indicates better fit, hence, out of 3 models Non-Linear Model has the best fit to the original data points. Which was also obvious through the graph in figure 7 itself. Naturally, it would be the wisest to choose Non-Linear Model to predict the data in this scenario.

Exercise 5: Tikhonov regularization [15 points]

Given the ill-posed problem $Ax = b$, with $A \in R^{n\times m}$ and $b \in R^m$, we aim to solve it using linear least squares with Tikhonov regularization, which is formulated as

$$min_x ||Ax - b||^2 + \alpha||x||^2 \tag{13}$$

where $\alpha \in R$ is a regularization parameter.

(a) Derive analytically the optimal solution x^* of equation13. What is the purpose of the parameter α and how should we proceed to choose its value?

From the equation 13, optimal solution x^* can be found by setting it as the Lagrangian and differentiating the equation in respect to x:

$$L(x) = ||Ax - b||^2 + \alpha||x||^2 \tag{14}$$

$$\frac{dL}{dx} = 2A^T(Ax - b) + 2\alpha x = 0 \tag{15}$$

The optimal solution or x^* can be expressed by solving equation 15:

$$x^* = (A^TA + \alpha I)^{-1}A^Tb \tag{16}$$

Purpose behind α in this context is to adjust the fit of the model to the given data. It is particularly useful in case matrix A is poorly conditioned or nearly singular and it needs to be adjusted to prevent overfitting. α is very important and it is often chosen over cross-validation or based-on given knowledge of level of fluctuation in the data. When choosing α, the value has to be chosen optimaly to minimize the residual of $||Ax - b||$ while keeping the solution of x small.

(b) We now consider the Hilbert matrix $H \in R^{n\times n}$, with entries defined as follows:

$$H_{ij} = \frac{1}{i + j - 1} \tag{17}$$

for every i, j = 1, . . . , n. Write a Matlab script *illposedHilbert.m* in which you generate H for n = 50, 100, 200, 300, 400, 500, 1000 and solve the problem $Hx = b$, for $b = Hx$ exact and x exact generated through the function *rand(n,1)*. To make your results reproducible, reinitialize the random number generator to its startup configuration by adding rng('default') at the beginning of your script. Produce also two figures in which you plot: (I) the condition number of H (use *cond()* in Matlab) against n; (II) the norm of the error $||x_{exact} - x_2||$ against n.

```
rng('default');

n = [50, 100, 200, 300, 400, 500, 1000];
condNum = zeros(length(n), 1);
errorNorms = zeros(length(n), 1);

for i = 1:length(n)
    H = hilb(n(i));
    x_exact = rand(n(i), 1);
    b = H * x_exact;
    x = H \ b;
    condNum(i) = cond(H);
    errorNorms(i) = norm(x_exact - x);
end

... graph the result ...
```

The values of n are stored in a predefined array of the same name. Then the array is looped through and each element is used to create a hilbert matrix using library function $hilb(matrixDimension)$, and x_{exact}. Which then is used to calculate individual conditional number and error norm. The results are then plotted both as a scatter and line graph. Resulting graph is as the following:

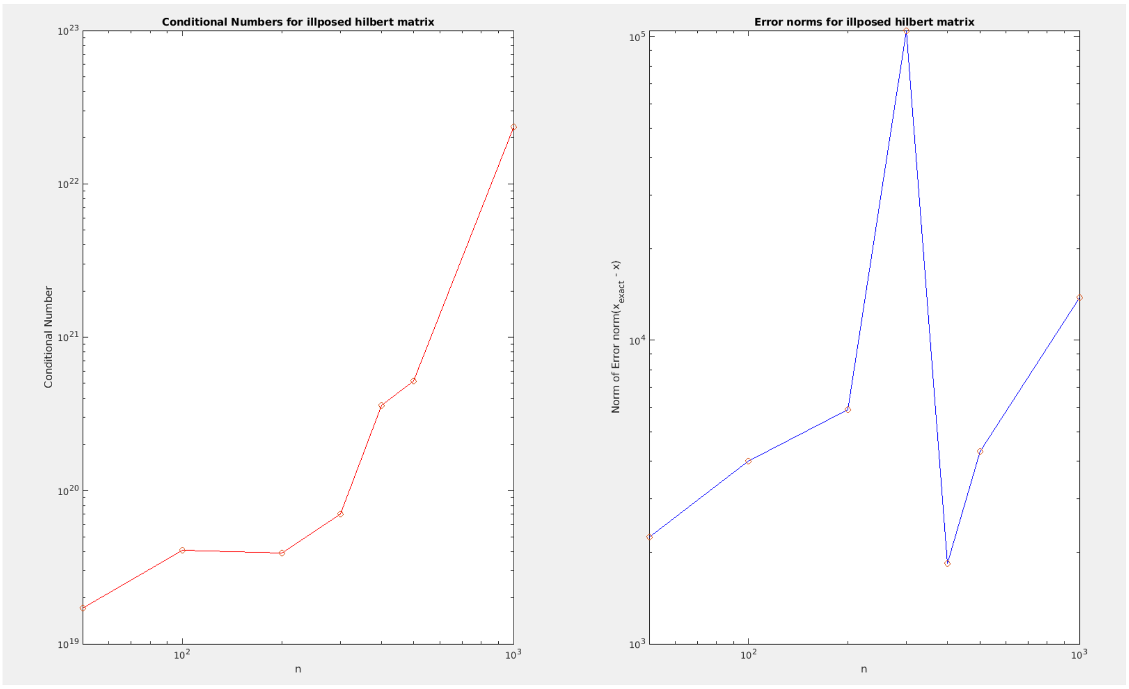

Figure 8: Illposed Hilbert Matrix's conditional numbers and error norms, for clarity the graph is logged

(c) We now focus our attention on the case n = 100. Write a Matlab script regularizedHilbert.m in which you estimate the regularized solution x reg according to Eq. (1) by using at least 10 different values of the parameter α (explain your choice of the values). To visualize the results, produce two figures in which you plot: (I) the norm of the error $||x_{exact} - x_2||_2$ against the values of α; (II) $||Hx = b||_2$ against $||x||_2$ for the different values of α. Comment your results.

Tikhonov regularization introduces a regularization parameter α, which controls the trade-off between the least squares fit $||Hx - b||_2$ and the magnitude of the solution vector $||x||_2$, as described in the equation 13.

```
alphas = logspace(-19, 1, 10);
for i = 1:length(alphas)
    x_reg = (H' * H + alphas(i) * eye(n)) \ H' * b;

    errorNorms(i) = norm(x_exact - x_reg);
    solutionNorms(i) = norm(x_reg);
    residualNorms(i) = norm(H * x_reg - b);
end

... graph the result ...
```

The x_{reg} value is calculated according to simplified version of the equation 13 as demonstrated in equation 16. From the x_{reg} calculated, the rest can be calculated using pre-defined values. The resulting graph is as the following:

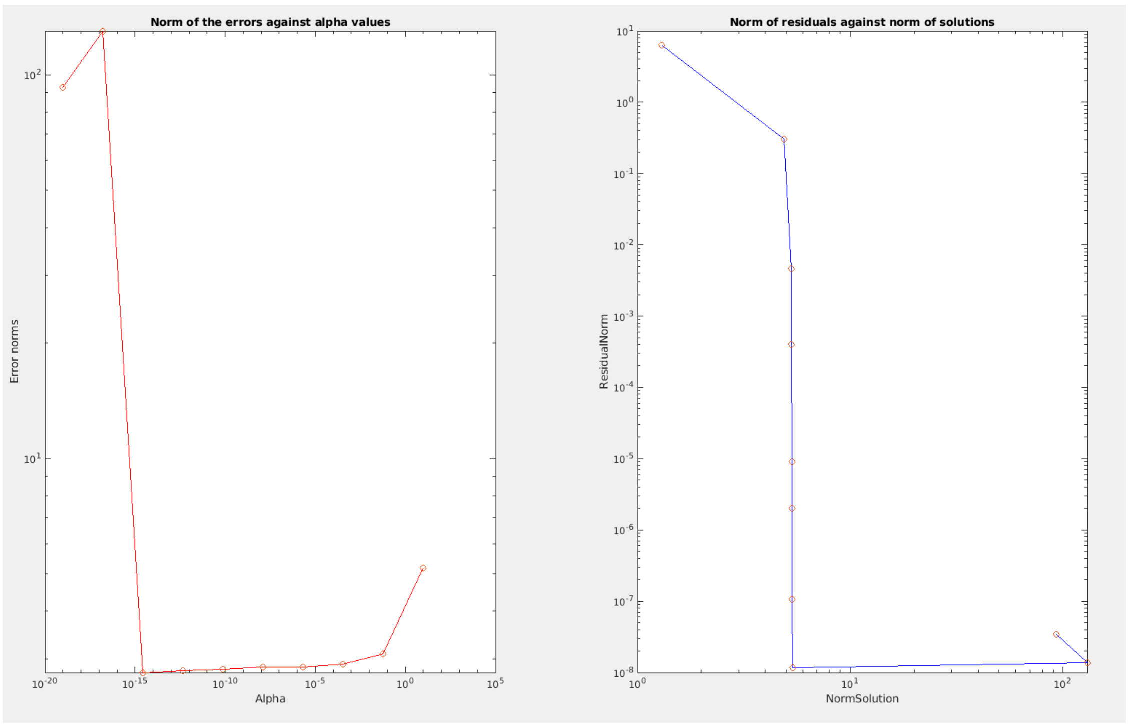

Figure 9: Regularized Matrix, the norm of the error $||x_{exact} - x_2||$ against α (left), and $||Hx - b||_2$ against $||x||_2$ (right), graphs are logged for better visualization

The graph on the left showing the alpha's relation against norm of errors shows that as the value of alpha increases, the norm of error increase in the beginning and fall drastically in the middle, and start increasing exponentially. It is an indication that there is an optimal range of values for α.

Unlike the graph on the right, graph on the left shows a downshift trend and a bit of a rise at the end. Which means as solution norm increases, the residual norm decreases to certain extent and trend is broken after a certain point. Which indicates that as solution norm increases, the graph fits better to the original data and after certain point it starts to overfit.

In conclusion, it would be important to find range of α values which minimizes the norm of error without underfitting or overfitting the graph to the original datapoints.