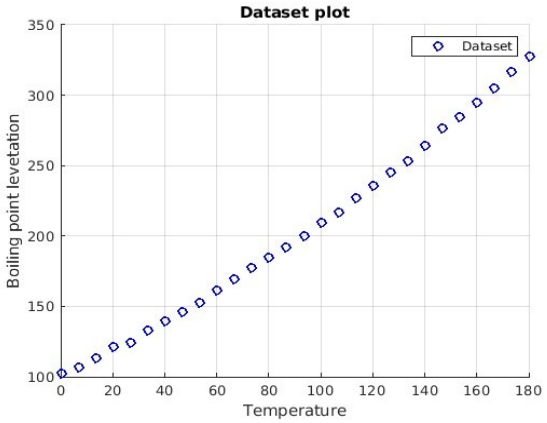
1. Downloaded the “MyAppInstaller_Con.exe” on my Linux PC and sent it to the “Saved messages” in the telegram web application, and from a Windows virtual machine downloaded the installer and run it then after finishing the installation, run the program as administrator and follow the instructions in the exam file, provided my student number and created the dataset, once again uploaded the file -the dataset this time- to telegram and downloaded it on my Linux PC to start working on it using MATLAB.

2. Read the data from the excel file using the “readtable” function since it is one of the recommended according to the matlab docs. Then plotted the dataset using the “plot” function by inputting the Temperature_C as a system input and the Boiling_pointElevation_K as a system output.
   The figure shows the plot of the data:



3. Given that the data consists of (x,y) data pairs not a time series, and they are read from a file not in real time, and we have a few number of data points, so decide to use black-box modeling and the mathematical first order model:
   $y = \beta_0 + \beta_1 x + \epsilon$
   for an ordinary linear least square. And it could be rewritten in matrix notation for N points as:
   $y = \mathbf{X}p + \epsilon$
   where:

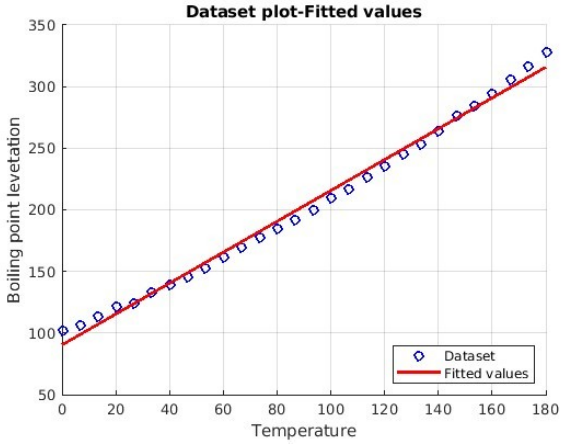
$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}, \mathbf{p} = \begin{bmatrix} a \\ b \end{bmatrix}, \mathbf{X} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_N \end{bmatrix}, \epsilon = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_N \end{bmatrix}$$

4. Constructing the design matrix **X** is done by including the “x” data points from the (x,y) data pairs as a column which represent “ β_1 ” and the y-axis intercept of the points which is considered as = 1, for all points, and it represent the “ β_0 ” from the mathematical model in the point “3”. and it will be as follows:

$$\mathbf{X} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_N \end{bmatrix}$$

where:
N = number of points.

5. To find the least squares estimator, need to solve the matrix mathematical model in '3' for the parameters 'p' and by applying matrix algebra, and ignoring the noise 'ϵ' we start by multiplying the equation by X^T to get a M *M matrix where M is the number of parameters as a result we get:
$(X^T \cdot X) \cdot p = X^T \cdot y$
now by left-multiplying both sides by $(X^T \cdot X)^{-1}$:
$(X^T \cdot X)^{-1} \cdot (X^T \cdot X) \cdot p = (X^T \cdot X)^{-1} \cdot (X^T \cdot y)$
and since $A^{-1} \cdot A = I$ and the identity matrix is like 1 in matrix algebra then the solution is:
$\hat{p} = (X^T X)^{-1} X^T y$
and from solving the equation in MATLAB the result is:
$\hat{P} = \begin{bmatrix} 90.2729 \\ 1.2530 \end{bmatrix}$
finding the fitted values of the model according to the following equation:
$y = \beta_0 + \beta_1 \cdot x$
where x here is an independent variable (non-random).
The resulting plot for the fitted values is:



6. The probable uncertainty in the estimation of the parameters (β_0 , β_1) is called the variance, and the covariance is another measure that is important to determine the probable uncertainty in the estimation of the parameters since it represent the relationship between two variables.
When the variances get written on the diagonal of a matrix and the rest of the matrix elements are the covariances then the matrix is called the variance-covariance matrix or varcov, and it could be found from the following equation:
varcov = $(X^T X)^{-1}$
running the above matrix in MATLAB gives the following result:
varcov = $\begin{bmatrix} 0.1355 & -0.0011 \\ -0.0011 & 0.0000 \end{bmatrix}$

7. First we explain homoskedasticity which refers to the case when the variance is constant for all values in the matrix y and hence for all the residuals 'ϵ', and the residuals are the noise in the

data or in other words the difference between the measured values ‘y’ and the predicted model function values.
To find the residuals we use the equation in ‘3’ that is:
y = **X**p + ϵ
where:
ϵ = y - **X**p
now to find the homoskedastic variance we use the equation:

$$\hat{\sigma}_e^2 = \frac{\epsilon^T \epsilon}{N - M}$$

and it gives the result:

$$\hat{\sigma}_e^2 = 31.9867$$

8. The variances of the parameters can be found from multiplying the variance-covariance matrix by the homoskedastic variance, and as the following derivation shows:
starting from the equation:

$$\hat{\boldsymbol{p}} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{y}$$

sub y = **X**p + ϵ in:

$$\hat{p} = (X^T X)^{-1} X^T (Xp + \epsilon)$$

and

$$\hat{\boldsymbol{p}} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{X}\boldsymbol{p} + (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{\epsilon}$$

$$\hat{p} = p + (X^T X)^{-1} X^T \epsilon$$

and moving the P to the other side of the equal sign:

$$\hat{p} - p = (X^T X)^{-1} X^T \epsilon$$

now multiplying each side by the transpose and taking the expectation of them we get the variance-covariance matrix:

$$E[(\hat{p} - p)(\hat{p} - p)^T] = E[((X^T X)^{-1} X^T \epsilon)((X^T X)^{-1} X^T \epsilon)^T]$$

to simplify:

$$E[(\hat{p} - p)(\hat{p} - p)^T] = E[(X^T X)^{-1} X^T \epsilon \epsilon^T X (X^T X)^{-1}]$$

$$E[(\hat{p} - p)(\hat{p} - p)^T] = (X^T X)^{-1} X^T E[\epsilon \epsilon^T] X (X^T X)^{-1}$$

given that:

$$E[\epsilon \epsilon^T] = \sigma^2 I$$

substitute in the variance-covariance matrix:

$$E[(\hat{\boldsymbol{p}} - \boldsymbol{p})(\hat{\boldsymbol{p}} - \boldsymbol{p})^T] = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T (\boldsymbol{\sigma}^2 \boldsymbol{I}) \boldsymbol{X} (\boldsymbol{X}^T \boldsymbol{X})^{-1}$$

we arrive at:

$$\boldsymbol{E}[(\hat{\boldsymbol{p}} - \boldsymbol{p})(\hat{\boldsymbol{p}} - \boldsymbol{p})^T] = (\boldsymbol{\sigma}^2 \boldsymbol{I})(\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{X} (\boldsymbol{X}^T \boldsymbol{X})^{-1} = (\boldsymbol{\sigma}^2 \boldsymbol{I})(\boldsymbol{X}^T \boldsymbol{X})^{-1}$$

solving the matrix in MATLAB we get the variances of the parameters on the diagonal:

$$(\boldsymbol{\sigma}^2 \boldsymbol{I})(\boldsymbol{X}^T \boldsymbol{X})^{-1} = \begin{bmatrix} 4.3332 & -0.0355 \\ -0.0355 & 0.0004 \end{bmatrix}$$

which shows that var(β_0) = 4.3332, and var(β_1) = 0.0004
now finding the standard deviation by taking the square root of the variance:
std_dev(β_0) = 2.0816,
and
std_dev(β_1) = 0.0198

9. Singular value decomposition is a factorization of a matrix into a product of three matrices starting with a rotation then rescaling and another rotation, and the design matrix can be decomposed according to the following equation:

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T$$

to find the singular value decomposition of the matrix using MATLAB we use the command svd(**X**). Now knowing that 'S' matrix is diagonal but not square, then the following rule holds:

$$\boldsymbol{S^T = S}$$

and the matrices U an V are orthogonal matrices that means that they are square and invertible and:

$$\boldsymbol{U^{-1} = U^T}$$

$$\boldsymbol{V^{-1} = V^T}$$

10. To find the parameters for the linear least square problem using singular value decomposition. Starting from the equation:

$$\hat{p} \;=\; \boldsymbol{(X^T X)^{-1} X^T y}$$

rearrange:

$$\boldsymbol{(X^T \cdot X)\, p = X^T y}$$

sub $\boldsymbol{X = USV^T}$ in:

$$\boldsymbol{(USV^T)^T (USV^T)\, p = (USV^T)^T}\, y$$

applying the transpose of the brackets:

$$\boldsymbol{(VSU^T)(USV^T)}\, p = \boldsymbol{(VSU^T)}\, y$$

rearranging

$$\boldsymbol{VS(U^T U)SV^T}\, p = \boldsymbol{(VSU^T)}\, y$$

since **U** is an orthogonal matrix and from the equation $A^{-1} \cdot A = I$:

$$\boldsymbol{VSSV^T}\, p = \boldsymbol{(VSU^T)}\, y$$

which equals:

$$\boldsymbol{SV^T p = U^T y}$$

now multiplying both sides of the equation by $\boldsymbol{(SV^T)^{-1}}$:

$$\boldsymbol{(SV^T)^{-1} SV^T p = (SV^T)^{-1} U^T y}$$

and we arrive at:

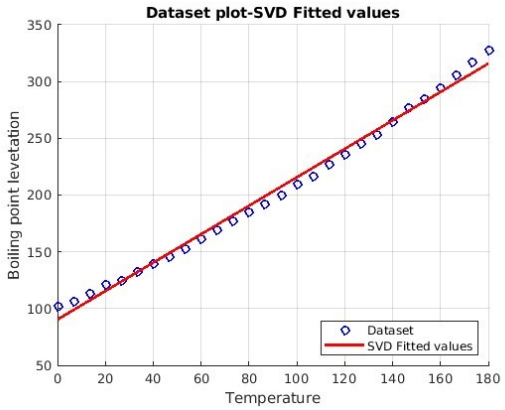
$$\boldsymbol{p = (VS^{-1} U^T)\, y}$$

now solving the the result in MATLAB shows that the equation can not be solved because the **S** matrix does not have the correct size, so we use the command svd(X, 'econ') to find the singular values decomposition for the design matrix which will give an economy-size decomposition differs from the original size by removing the extra rows or columns of zero from the **S** matrix and the columns in either **U** or **V** that multiplies those zeros.

Now after finding the economy-size **USV** matrices and trying again to solve the equation in MATLAB we get the parameters:

P = $\begin{bmatrix} 90.2729 \\ 1.2530 \end{bmatrix}$

and when plot the fitted values for the model we get:



which is showing the same result as the plot earlier.