LAB 6 - LEAST SQUARES I

1. Finding the line of best fit

Suppose a satellite is attached to a rocket and is being launched into space. Once it leaves the atmosphere the satellite will separate from the rocket, and will be pushed by thrusters until it reaches its place of orbit. Let v_0 denote the initial velocity of the satellite at the time it separates from the rocket (in km/s), and assume that the acceleration caused by the thrusters is a constant a (in km/s^2). Then the velocity of the satellite at t seconds after separation is given by

$$v(t) = v_0 + at. \tag{1}$$

Suppose that we do not know the initial velocity of the satellite v_0, nor the acceleration a due to the thrusters, but we are able to measure its velocity $v(t)$ every 5 seconds, starting at $t = 5$ (see Table 1). Can we determine the values of v_0 and a from this information?

Time (seconds)	Velocity (kilometers per second)
5	3.33
10	4.43
15	4.39
20	5.23
25	5.67
30	6.06
35	7.01
40	7.16
45	8.03
50	8.78

Table 1. Velocity of an accelerating satellite.

Because of the equation $v(t) = v_0 + at$, we should expect that if we plot the points in Table 1 in the xy-plane they will all lie along a line with slope a and y-intercept v_0. Thus, if we want to determine the values of a and v_0, all we would need to do is plot the points, and determine the slope and y-intercept of the line that passes through them.

Unfortunately, when we plot the points in Figure 1 we see that although the data points look like they roughly lie along a line, there does not actually exist a single line which passes through all of the points. This is because of *noise* (or *error*) in our data, which results in part from the fact that it is not possible to measure the velocity of an object with complete accuracy and precision.

In Figure 2 we plot a few examples of lines which pass through some of the points (but not all of them). Each of these lines will give us a different slope and y-intercept, and hence will yield different values for a and v_0.

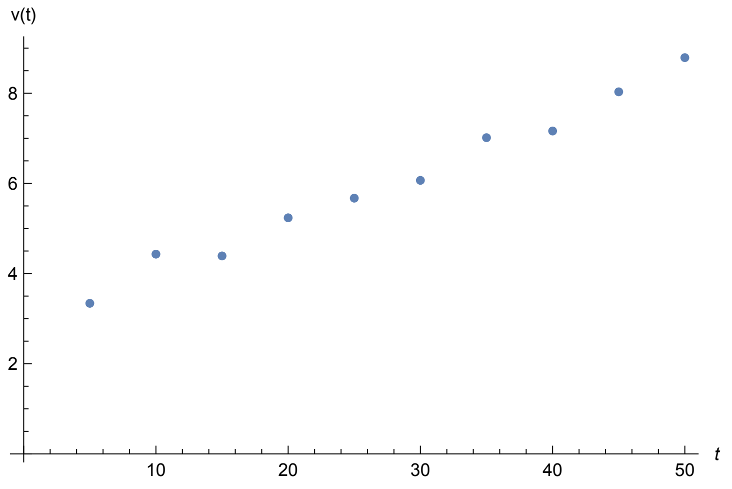

FIGURE 1. Velocity of the satellite from Table 1.

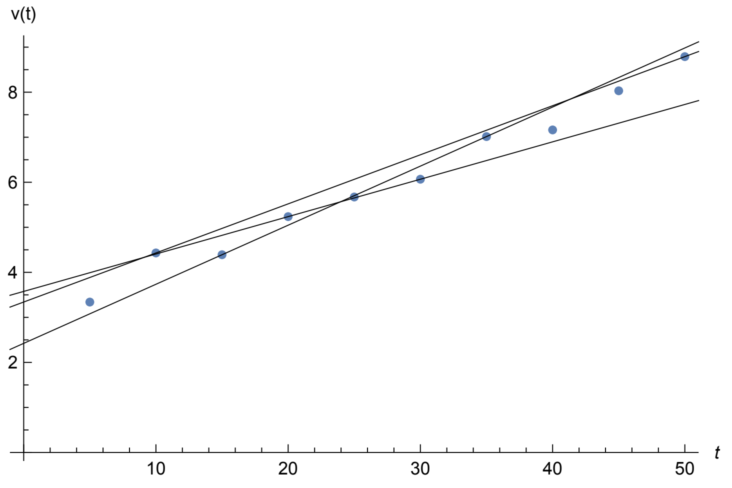

FIGURE 2. Velocity of the satellite from Table 1 with several candidate lines of best fit.

- Which line should we pick to estimate a and v_0?
- Is it possible to find a line which best fits the data as a whole, and not just some of the data points?

We can answer these questions by restating them in terms of a matrix equation. Let (t_5, v_5), $(t_{10}, v_{10}), \ldots, (t_{50}, v_{50})$ be the data points we've gathered from our observations. Ideally, we want to find values of a and v_0 so that for each time t_i we have $v_i = v_0 + at_i$, but this system is overdetermined, and therefore likely to not have an exact solution. However, we can ask for the "best approximate" solution. This can be summarized as a vector equation by saying that we want to find a and v_0 so that

$$\begin{bmatrix} v_5 \\ v_{10} \\ v_{15} \\ v_{20} \\ v_{25} \\ v_{30} \\ v_{35} \\ v_{40} \\ v_{45} \\ v_{50} \end{bmatrix} = \begin{bmatrix} v_0 + at_5 \\ v_0 + at_{10} \\ v_0 + at_{15} \\ v_0 + at_{20} \\ v_0 + at_{25} \\ v_0 + at_{30} \\ v_0 + at_{35} \\ v_0 + at_{40} \\ v_0 + at_{45} \\ v_0 + at_{50} \end{bmatrix}$$

Plugging in values for t_i and v_i, and rewriting it as a matrix equation gives

$$\begin{bmatrix} 3.33 \\ 4.43 \\ 4.39 \\ 5.23 \\ 5.67 \\ 6.06 \\ 7.01 \\ 7.16 \\ 8.03 \\ 8.78 \end{bmatrix} = \begin{bmatrix} v_5 \\ v_{10} \\ v_{15} \\ v_{20} \\ v_{25} \\ v_{30} \\ v_{35} \\ v_{40} \\ v_{45} \\ v_{50} \end{bmatrix} = \begin{bmatrix} v_0 + 5a \\ v_0 + 10a \\ v_0 + 15a \\ v_0 + 20a \\ v_0 + 25a \\ v_0 + 30a \\ v_0 + 35a \\ v_0 + 40a \\ v_0 + 45a \\ v_0 + 50a \end{bmatrix} = \begin{bmatrix} 1 & 5 \\ 1 & 10 \\ 1 & 15 \\ 1 & 20 \\ 1 & 25 \\ 1 & 30 \\ 1 & 35 \\ 1 & 40 \\ 1 & 45 \\ 1 & 50 \end{bmatrix} \begin{bmatrix} v_0 \\ a \end{bmatrix}.$$

Letting

$$Y = \begin{bmatrix} 3.33 \\ 4.43 \\ 4.39 \\ 5.23 \\ 5.67 \\ 6.06 \\ 7.01 \\ 7.16 \\ 8.03 \\ 8.78 \end{bmatrix}, \qquad X = \begin{bmatrix} 1 & 5 \\ 1 & 10 \\ 1 & 15 \\ 1 & 20 \\ 1 & 25 \\ 1 & 30 \\ 1 & 35 \\ 1 & 40 \\ 1 & 45 \\ 1 & 50 \end{bmatrix}, \qquad \text{and} \qquad \beta = \begin{bmatrix} v_0 \\ a \end{bmatrix},$$

we can then say that finding a line passing through all of our data points is the same as finding values for the vector β, so that the matrix equation $Y = X\beta$ is satisfied. (We call the vector Y the *observation vector*, because it holds the values we've observed in our measurements.)

Problem 1. Create NumPy arrays `X1` and `Y1` using the values in the above matrices X and Y respectively.

Hint: recall that as `Y1` *is a vector, we create it in NumPy using a command of the form* `Y1=np.array([3.33,...,8.78])` *and not* `Y1=np.array([[3.33],...,[8.78]])`.

Since we've already observed that there isn't a line that passes through all of the data points, this tells us that there will not be a vector β which satisfies the equation $Y = X\beta$ exactly. We can, however, try to find a vector $\widehat{\beta}$ which will make $X\widehat{\beta}$ *as close as possible to* Y. This will correspond to finding the slope a and intercept v_0 of a line which fits our data set as closely as possible.

To find such a vector $\widehat{\beta}$, we don't try to solve our original equation $X\beta = Y$, but rather we find a vector $\widehat{\beta}$ which satisfies

$$X^T X\widehat{\beta} = X^T Y.$$

We call this new equation the *normal equation of the system* $X\beta = Y$. It turns out that even though we can't find a vector β which satisfies our original equation $X\beta = Y$, we will *always* be able to find a vector $\widehat{\beta}$ which satisfies the normal equation $X^T X\widehat{\beta} = X^T Y$. Moreover, such a vector $\widehat{\beta}$ will make it so that $X\widehat{\beta}$ is as close as possible to Y. In other words, *a solution to the normal equation will be the best possible approximation to a solution for the original system!* We call the vector $\widehat{\beta}$ a *least-squares solution* to the system $X\beta = Y$.

Problem 2.
(1) Compute the coefficient matrix $X^T X$ for the normal equations, and save its value as `normal_coef1`. Recall, the transpose of a NumPy array `A` can be obtained by `np.transpose(A)`, while the matrix produce of two (appropriately-sized) NumPy arrays `A` and `B` is obtained by `np.matmul(A,B)` or `A@B`.
(2) Compute the right-hand side $X^T Y$ of the normal equations, and save its value as `normal_vect1`.

Returning to our original problem, we observe that finding a vector

$$\widehat{\beta} = \begin{bmatrix} v_0 \\ a \end{bmatrix}$$

which satisfies the normal equation $X^T X\widehat{\beta} = X^T Y$ will tell us the y-intercept v_0 and slope a of the line that best fits our data. These values give us the best approximation to the initial velocity and acceleration of the satellite that we can obtain from our data.

In problems that follow, we will need to solve a matrix equation of the form $A\mathbf{x} = \mathbf{b}$, where A is an invertible square matrix. To do this, we can use the function `np.linalg.solve(A,b)` which will return the vector `x` as a NumPy array. Notice that `A` must be square in order to use this function.

Problem 3.
1. Using the NumPy function `np.linalg.solve` find a solution $\widehat{\beta}$ to the normal equations $X^TX\widehat{\beta} = X^TY$. Save the value of your solution as `beta1`. This is the least squares approximation to the original system $X\beta = Y$.
2. Interpreting the entries in `beta1` as the y-intercept v_0 and slope a of a line in the xy-plane, use `beta1` to define a function `ls1_line(x)` whose input is an x-value `x`, and whose output `y` is the corresponding y-coordinate on this line. For example, `ls1_line(12.5)` should return `4.3030000000000035`.
3. Plot the function `ls1_line` along with the data points in Table 1.
4. Use the function `ls1_line` to predict the velocity of the satellite at time $t = 60$. Save your answer as the variable `pred1`.

It should be noted that the normal equations can be used in other more general situations (not just to find lines of best fit), because of the following theorem.

Theorem 1. *Let A be an $m\times n$ matrix, and $\mathbf{b}$ a vector in $\mathbb{R}^m$. Then the equation $A^TA\mathbf{x} = A^T\mathbf{b}$ will be consistent, and any solution $\widehat{\mathbf{x}}$ to $A^TA\mathbf{x} = A^T\mathbf{b}$ will have the property that*

$$\|A\widehat{\mathbf{x}} - \mathbf{b}\| \leq \|A\mathbf{x} - \mathbf{b}\|$$

for any $\mathbf{x}$ in $\mathbb{R}^n$.

What the above theorem says is that even though the original system $A\mathbf{x} = \mathbf{b}$ may be inconsistent, the normal equations $A^TA\mathbf{x} = A^T\mathbf{b}$ will always have a solution $\widehat{\mathbf{x}}$, and that solution will make $A\widehat{\mathbf{x}}$ as close as possible to $\mathbf{b}$.

This theorem is very useful for computations in Python. The function `np.linalg.solve` can only be used to solve equations $A\mathbf{x} = \mathbf{b}$ when A is both square and invertible. When A is either not invertible, or not square, we can use `np.linalg.lstsq(A,b)` to find a least-squares solution to $A\mathbf{x} = \mathbf{b}$. When $A\mathbf{x} = \mathbf{b}$ has a solution, then the least-squares solution will be an exact solution. When $A\mathbf{x} = \mathbf{b}$ does not have solution, then `np.linalg.lstsq(A,b)` will return a closest possible approximation to an exact solution. While we won't use `np.linalg.lstsq(A,b)` directly in this lab, you can try using it to check your work.

2. Finding curves of best fit

In the previous example we saw how to fit lines to our data when the data looked (or was assumed to be) linear. In this next example we will see that we can apply similar techniques to find more complicated curves to fit data which doesn't necessarily follow a linear relation.

Suppose in our above example of the satellite separating from the rocket that instead of measuring the velocity of the satellite every 5 seconds, we measure the distance the satellite has traveled from the point of separation. If the satellite is traveling at an initial velocity of v_0 kilometers per second when it separates, and undergoes a constant acceleration of a kilometers per second squared, then by integrating equation (1) the distance d it has traveled at time t is given by

$$d(t) = v_0t + \frac{a}{2}t^2. \tag{2}$$

To simplify our calculations, we will set $A = \frac{a}{2}$, and equation (2) becomes

$$d(t) = v_0t + At^2. \tag{3}$$

Time (seconds)	Distance (kilometers)
5	20.57
10	87.48
15	197.45
20	347.67
25	546.12
30	784.35
35	1066.02
40	1390.97
45	1761.85
50	2177.34

Table 2. Distance traveled by an accelerating satellite.

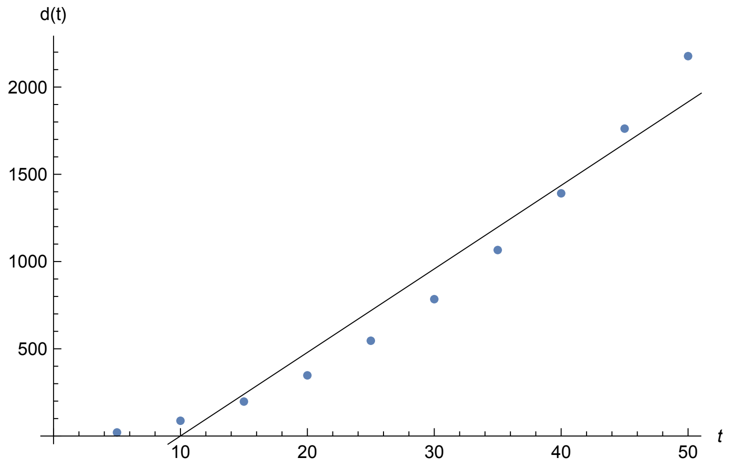

Figure 3. Distance traveled by the satellite from Table 2.

The data gathered by our measurements is summarized in Table 2.

We can plot the data in Table 2, and use the method of least squares as above to find a line of best fit to the data. In this case the line that best fits our data has the equation

$$d = -478.43 + 47.87t.$$

We plot this line, along with the data from Table 2 in Figure 3. Notice that this line seems to fit the data reasonably well. However, if we track the satellite's position for an additional 50 seconds, and plot the resulting data in Figure 4, we see that our choice of line doesn't seem so ideal anymore.

Why does our line seem to fit so poorly when we continue tracking the progress of the satellite? If we look back at equation (3) we notice that it doesn't describe a line at all, but rather a parabola! So instead of looking for a line of best fit, we should be looking for a parabola that best fits our data.

To find such a parabola, we want to find v_0 and A so that each of our data points (t_5, d_5), $(t_{10}, d_{10}), \ldots, (t_{50}, d_{50})$ satisfies the equation

$$d_j = v_0 t_j + At_j^2.$$

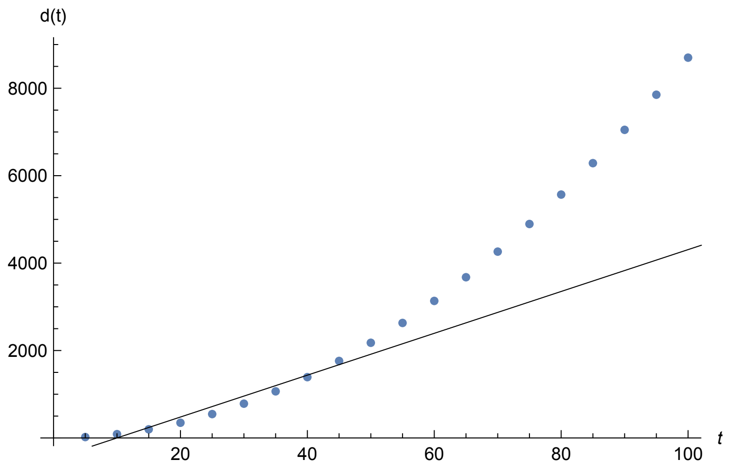

FIGURE 4. More distance data on our satellite.

We can again express these ten equations as a single vector equation

$$\begin{bmatrix} d_5 \\ d_{10} \\ \vdots \\ d_{50} \end{bmatrix} = \begin{bmatrix} v_0 t_5 + At_5^2 \\ v_0 t_{10} + At_{10}^2 \\ \vdots \\ v_0 t_{50} + At_{50}^2 \end{bmatrix}$$

which we can rewrite as a matrix equation

$$\begin{bmatrix} d_5 \\ d_{10} \\ \vdots \\ d_{50} \end{bmatrix} = \begin{bmatrix} t_5 & t_5^2 \\ t_{10} & t_{10}^2 \\ \vdots & \vdots \\ t_{50} & t_{50}^2 \end{bmatrix} \begin{bmatrix} v_0 \\ A \end{bmatrix}.$$

Define the matrices Y, X, and β as follows:

$$Y = \begin{bmatrix} d_5 \\ d_{10} \\ \vdots \\ d_{50} \end{bmatrix} = \begin{bmatrix} 20.57 \\ 87.48 \\ \vdots \\ 2177.34 \end{bmatrix}$$

$$X = \begin{bmatrix} t_5 & t_5^2 \\ t_{10} & t_{10}^2 \\ \vdots & \vdots \\ t_{50} & t_{50}^2 \end{bmatrix} = \begin{bmatrix} 5 & 5^2 \\ 10 & 10^2 \\ \vdots & \vdots \\ 50 & 50^2 \end{bmatrix}, \qquad \text{and} \qquad \beta = \begin{bmatrix} v_0 \\ A \end{bmatrix}.$$

Then finding a parabola $d = v_0 t + At^2$ which passes through all of our points is equivalent to finding a solution β to the matrix equation $Y = X\beta$.

Problem 4. Create NumPy arrays `X2` and `Y2` using the values in the above matrices X and Y respectively.

Since we can't find an exact solution to $X\beta = Y$, we can instead look for an *approximate* solution to $X\beta = Y$, and thus find a parobola which *approximates* our data. The best approximation to a solution of $X\beta = Y$ is again found by computing the normal equation

$$X^T X\widehat{\beta} = X^T Y$$

and finding a solution to this equation (recall that the normal equations are always guaranteed to have a solution, but the solution may not be unique). If $X^T X$ is invertible, then the unique solution is given by $\widehat{\beta} = (X^T X)^{-1}X^T Y$.

Problem 5.
1. Using `X2` and `Y2` compute the coefficient matrix $X^T X$ for the normal equations above, and save its value as `normal_coef2`.
2. Using `X2` and `Y2` compute the right-hand side $X^T Y$ of the normal equations above, and save its value as `normal_vect2`.
3. Solve the normal equations $X^T X\widehat{\beta} = X^T Y$ to find the least squares solution $\widehat{\beta}$ for the system $X\beta = Y$. Save your solution as a variable named `beta2`.

Problem 6.
1. Use your solution `beta2` to define a function `ls2_par(x)` whose input is an x-value `x`, and whose output `y` is the corresponding y-coordinate on this parabola of best fit. *Remember,* $A = \frac{a}{2}$*!* For example, executing the command `ls2_par(13.1)` should return `149.73834158597967`.
2. Plot the resulting parabola of best fit along with the data points from Table 2.
3. Use the function `ls2_par` to predict the position of the satellite at time $t = 60$ seconds. Save your answer as a variable named `pred2`.

3. Kepler's Third Law

According to Kepler's third law of planetary motion, the semi-major axis r of the orbit of a planet around the sun (see Figure 5) is proportional to a power of the period T of the orbit (i.e. the time it takes for a planet to complete one full orbit around the sun).

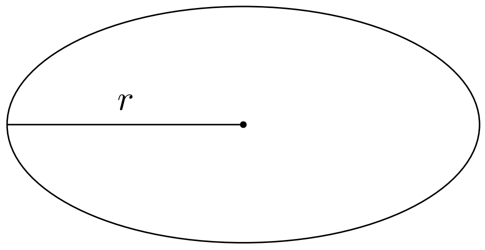

FIGURE 5. The semi-major axis of an ellipse.

We thus have

$$r = cT^a, \tag{4}$$

for some exponent a and some constant of proportionality c. In 1618 Kepler had estimates available for the period and semi-major axes of six planets orbiting around the sun. Kepler's data is included in Table 3.

	Semi-major axis r (in AU)	Period T (in Earth days)
Mercury	0.389	87.77
Venus	0.724	224.70
Earth	1	365.25
Mars	1.524	686.95
Jupiter	5.2	4332.62
Saturn	9.510	10759.2

TABLE 3. Estimates from 1618 for the semi-major axes and periods of six planets orbiting the sun.

We will use the data in Table 3 to estimate the constant of proportionality c and exponent a in equation (4). First we must convert it to a linear equation, which we do by taking the natural logarithm of both sides of equation (4) and using properties of logarithms to give

$$\begin{aligned}\ln(r) &= \ln(cT^a)\\ &= \ln(c) + a\ln(T).\end{aligned}$$

We can think of this as being a linear equation $y = \ln(c) + ax$, in the variables $x = \ln(T)$ and $y = \ln(r)$, with slope a and y-intecept $\ln(c)$.

As in the above example, we will be looking to find a parameter vector

$$\beta = \begin{bmatrix}\ln(c)\\ a\end{bmatrix}$$

which most closely fits the model in equation (4) to our data. We will construct the observation vector Y and matrix X as above. However, because our linear model is $\ln(r) = \ln(c) + a\ln(T)$, we will use the logarithms of r and T instead of using their values directly.

Problem 7. (1) Find the matrix X and the observation vector Y as above to express the planetary orbit data in Table 3 using the linear model $\ln(r) = \ln(c) + a\ln(T)$. Save the values of these matrices as `X3` and `Y3` respectively. (Enter the data in the same order it is displayed in Table 3.)

(2) Compute the coefficient matrix X^TX and right-hand side X^TY for the normal equations above, and save their values as `normal_coef3` and `normal_vect3`, respectively.

(3) Solve the normal equations $X^TX\widehat{\beta} = X^TY$ to find the least squares solution $\widehat{\beta}$ for the system $X\beta = Y$. Save your solution as a variable named `beta3`.

Problem 8. Using your least-squares solution `beta3` to $X\beta = Y$ and the model $r = cT^a$, predict the semi-major axes of the orbits of Uranus and Neptune, given that their periods are 30687.15 and 60190.03 respectively. Save these predictions as variables named `pred_Uran` and `pred_Nept` respectively.