Numerical Optimisation and Inverse Problems Project 3

Edward Small 10786391

April 2021

Contents

1 Linear and Nonlinear Least Squares

The average number of rabbits $y(t)$ (measure in units of hundreds) observed per square kilometre over time t (measured in units of years) in a national park is assumed to follow the exponential law

$$y(t) = a_1 e^{a_2 t} \tag{1}$$

with unknown constants $a_1 > 0$ and $a_2 \in \mathbb{R}$. Obeservations at 5 different times $t_1, ..., t_5$ yield the following data

k	t_k	y_k
1	1	3.2939
2	2	4.2699
3	4	7.1749
4	5	9.3008
5	8	20.259

Table 1: Table of observed values for rabbit population over 8 years

A standard approach is to estimate the two parameters a_1 and a_2 by solving a nonlinear unconstrained output least squares problem

$$\mathbf{a}^* = \text{argmin}_{\mathbf{a}} F(\mathbf{a}) \tag{2}$$

where $\mathbf{a} = [a_1, a_2]^T$ and

$$F(\mathbf{a}) = \frac{1}{2} \sum_{k=1}^{5} (f_k(\mathbf{a}))^2 \tag{3}$$

with

$$f_k(\mathbf{a}) = a_1 e^{a_2 t_k} - y_k, \quad k = 1, ..., 5 \tag{4}$$

An alternate approach for addressing this nonlinear least squares problem consists of linearising it and then solving the linear least squares problem instead. By taking natural logarithms on both sides of (1), we obtain the relationship

$$\ln(y(t)) = \ln(a_1) + a_2 t \tag{5}$$

By denoting

$$z(t) = \ln(y(t)) \quad b_1 = \ln(a_1) \quad b_2 = a_2 \tag{6}$$

we arrive at the linear representation of the problem

$$z(t) = b_1 + b_2 t \tag{7}$$

which gives the augmented table

k	t_k	y_k	$z_k = \ln(y_k)$
1	1	3.2939	1.1921
2	2	4.2699	1.4516
3	4	7.1749	1.9706
4	5	9.3008	2.2301
5	8	20.259	3.0086

Table 2: Table of observed values for rabbit population over 8 years with linearised values

The linearised problem can then be stated as

$$\mathbf{a}^* = \text{argmin}_{\mathbf{a}} \tilde{F}(\mathbf{a}), \qquad \text{with} \quad \tilde{F}(\mathbf{a}) = \frac{1}{2}\sum_{k=1}^{5}(\tilde{f}_k(\mathbf{a}))^2 \tag{8}$$

where

$$\tilde{f}_k(\mathbf{a}) = b_1 + b_2 t_k - z_k, \quad k = 1, ..., 5 \tag{9}$$

This is a linear least squares problem for finding the two unknown model parameters b_1 and b_2 from the data z_k, $k = 1, ..., 5$. The solution to the linear least squares problem can conveniently be obtained by solving the associated linear system of normal equations

$$B^T B\mathbf{x} = B^T \mathbf{d} \tag{10}$$

1.1 Expressions for B, x and d

Question: Calculate expressions for the matrix B and vectors $\mathbf{x}$ and $\mathbf{d}$ such that (10) represents the system of normal equations for problem.

We want to find the best line fit for an $n - 1 = 1$ (so first degree) polynomial, which is equation (7) with $m = 5$ data points. We therefore expect $B \in \mathbb{R}^{5\times 2}$, where

$$B_{ij} = t_i^{j-1}, \qquad \text{for} \quad i = 1, ..., 5, \quad j = 1, 2 \tag{11}$$

which gives

$$B = \begin{bmatrix} 1 & t_1 \\ 1 & t_2 \\ 1 & t_3 \\ 1 & t_4 \\ 1 & t_5 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 4 \\ 1 & 5 \\ 1 & 8 \end{bmatrix} \tag{12}$$

so (9) becomes

$$\tilde{f}_k(\mathbf{a}) = \left\{ \sum_{j=1}^{2} b_j B_{kj} \right\} - z_k \tag{13}$$

If we define $\mathbf{x}$ to be the vector of unknown coefficents, so

$$\mathbf{x} = [b_1, b_2]^T \tag{14}$$

and $\mathbf{d}$ to be the data vector, so

$$\mathbf{d} = [1.1921, 1.4516, 1.9706, 2.2301, 3.0086]^T \tag{15}$$

$d_k = z_k$, then we can use (13) to define a residuals vector $\mathbf{r}$, defined as

$$\begin{bmatrix} \tilde{f}_1(\mathbf{a}) \\ \tilde{f}_2(\mathbf{a}) \\ \tilde{f}_3(\mathbf{a}) \\ \tilde{f}_4(\mathbf{a}) \\ \tilde{f}_5(\mathbf{a}) \end{bmatrix} = B\mathbf{x} - \mathbf{d} = \mathbf{r} \tag{16}$$

From (8) we can cleary see that

$$\tilde{F}(\mathbf{a}) = \frac{1}{2}\mathbf{r}^T\mathbf{r} \tag{17}$$

which, when expanded out, becomes

$$\begin{aligned}\tilde{F}(\mathbf{a}) &= \frac{1}{2}(B\mathbf{x}-\mathbf{d})^T(B\mathbf{x}-\mathbf{d}) \\ &= \frac{1}{2}\mathbf{x}^T B^T B\mathbf{x} - \frac{1}{2}\mathbf{d}^T B\mathbf{x} - \frac{1}{2}\mathbf{x}^T B^T\mathbf{d} + \frac{1}{2}\mathbf{d}^T\mathbf{d}\end{aligned} \tag{18}$$

Since $\mathbf{d}^T B\mathbf{x} = \mathbf{x}^T B^T\mathbf{d}$, if we take $A = B^T B$, $\mathbf{b} = B^T\mathbf{d}$ and $c = \frac{1}{2}\mathbf{d}^T\mathbf{d}$ then we get

$$\tilde{F}(\mathbf{a}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{b}^T\mathbf{x} + c \tag{19}$$

which is the general form of a quadratic program. Not only this, but because A is made up of a known matrix that has been squared, it is therefore symmetric positive semi-definite (in this case it is symmetric positive definite, and must have a global minimum). We know that a necessary condition of an unconstrained minimum of a quadratic program is that

$$\nabla\tilde{F} = A\mathbf{x} - \mathbf{b} = 0 \tag{20}$$

Plugging back in the values for this problem yields

$$B^T B\mathbf{x} - B^T\mathbf{d} = 0 \tag{21}$$

which, rearranged, gives

$$B^T B\mathbf{x} = B^T\mathbf{d} \tag{22}$$

1.2 Solving the System

Question: Solve the system (10) for $\mathbf{x}$ and deduce expressions for the unknown parameters b_1 and b_2.

Solving the system at this point is fairly straight forward. We first need to find $B^T B$, so

$$B^T = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ t_1 & t_2 & t_3 & t_4 & t_5 \end{bmatrix} \tag{23}$$

and therefore

$$\begin{aligned} B^T B &= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ t_1 & t_2 & t_3 & t_4 & t_5 \end{bmatrix} \begin{bmatrix} 1 & t_1 \\ 1 & t_2 \\ 1 & t_3 \\ 1 & t_4 \\ 1 & t_5 \end{bmatrix} \\ &= \begin{bmatrix} 1+1+1+1+1 & t_1+t_2+t_3+t_4+t_5 \\ t_1+t_2+t_3+t_4+t_5 & t_1^2+t_2^2+t_3^2+t_4^2+t_5^2 \end{bmatrix} \\ &= \begin{bmatrix} 5 & \sum_{k=1}^5 t_k \\ \sum_{k=1}^5 t_k & \sum_{k=1}^5 t_k^2 \end{bmatrix} \end{aligned} \tag{24}$$

which has determinant

$$|B^T B| = 5\sum_{k=1}^5 t_k^2 - \left\{\sum_{k=1}^5 t_k\right\}^2 \tag{25}$$

and so

$$(B^T B)^{-1} = \frac{1}{5\sum_{k=1}^5 t_k^2 - \left\{\sum_{k=1}^5 t_k\right\}^2} \begin{bmatrix} \sum_{k=1}^5 t_k^2 & -\sum_{k=1}^5 t_k \\ -\sum_{k=1}^5 t_k & 5 \end{bmatrix} \tag{26}$$

So, taking (22), we can reformulate to find $\mathbf{x}$ such that

$$\mathbf{x} = (B^T B)^{-1} B^T \mathbf{d} \tag{27}$$

So, if $\mathbf{d}$ is the data vector (containing each z_k) then

$$\begin{aligned}
\begin{bmatrix} b_1 \\ b_2 \end{bmatrix} &= \frac{1}{5\sum_{k=1}^{5} t_k^2 - \left\{\sum_{k=1}^{5} t_k\right\}^2} \begin{bmatrix} \sum_{k=1}^{5} t_k^2 & -\sum_{k=1}^{5} t_k \\ -\sum_{k=1}^{5} t_k & 5 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ t_1 & t_2 & t_3 & t_4 & t_5 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \end{bmatrix} \\
&= \frac{1}{5\sum_{k=1}^{5} t_k^2 - \left\{\sum_{k=1}^{5} t_k\right\}^2} \begin{bmatrix} \sum_{k=1}^{5} t_k^2 & -\sum_{k=1}^{5} t_k \\ -\sum_{k=1}^{5} t_k & 5 \end{bmatrix} \begin{bmatrix} \sum_{k=1}^{5} z_k \\ \sum_{k=1}^{5} z_k t_k \end{bmatrix}
\end{aligned} \tag{28}$$

This can then be used to find expression for each linearised coefficient

$$\begin{aligned}
b_1 &= \frac{1}{5\sum_{k=1}^{5} t_k^2 - \left\{\sum_{k=1}^{5} t_k\right\}^2} \left[\left\{ \sum_{k=1}^{5} t_k^2 \right\} \left\{ \sum_{k=1}^{5} z_k \right\} + \left\{ -\sum_{k=1}^{5} t_k \right\} \left\{ \sum_{k=1}^{5} z_k t_k \right\} \right] \\
b_2 &= \frac{1}{5\sum_{k=1}^{5} t_k^2 - \left\{\sum_{k=1}^{5} t_k\right\}^2} \left[\left\{ -\sum_{k=1}^{5} t_k \right\} \left\{ \sum_{k=1}^{5} z_k \right\} + \left\{ 5\sum_{k=1}^{5} z_k t_k \right\} \right]
\end{aligned} \tag{29}$$

Having the system organised this way makes it easy to alter the values of b_1 and b_2 if there is some kind of change to the data (for example, we add more data points).

1.3 Coefficients

Question: Use the relationship (6) for extracting values for a_1 and a_2 from b_1 and b_2.

Plugging in the data provided, we get

$$\begin{aligned}
b_1 &= 0.9326 \\
b_2 &= 0.2595
\end{aligned} \tag{30}$$

and using the relationship from (6), we find

$$\begin{aligned}
a_1 &= e^{b_1} \\
&= e^{0.9326} \\
&= 2.5411 \\
a_2 &= b_2 \\
&= 0.2595
\end{aligned} \tag{31}$$

to four decimal places (plots use a higher precision).

1.4 Linearised Solution vs Nonlinearised Solution

Question: Would you expect that, with exact arithmetic, the results for a_1, a_2 obtained from the linearised least squares approach should coincide with the results for a_1, a_2 obtained from the original nonlinear approach?

The answer, in truth, is dependant on the data, but in the vast majority of cases we will obtain different results. We can see this in practice with the example above.

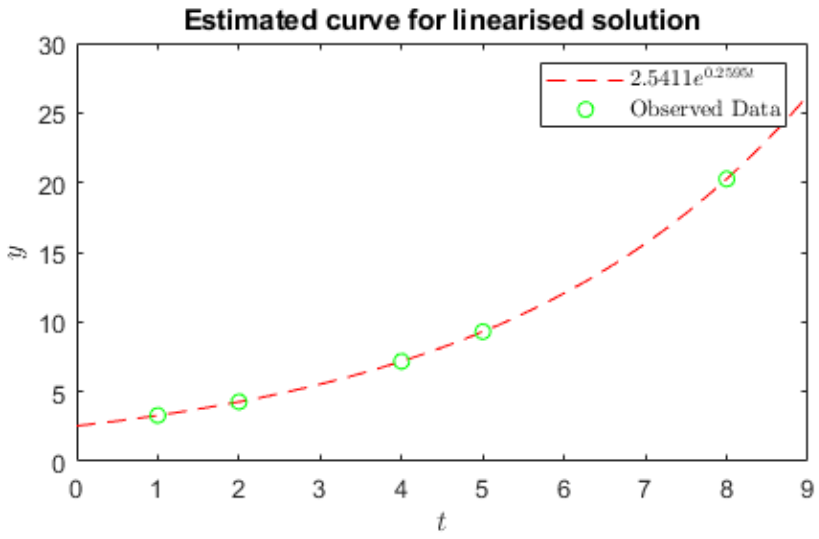

Figure 1: Curve fitted to the data using the linearised approach

Figure 1 clearly shows that the linearised approach, in this case, gives a satisfactory solution. The red dotted line is fitted very well to the data. However, if we really want to see know how well the linearised results for a_1 and a_2 fit the data we need to calculate the gradient of the minimisation problem of the nonlinear system given in (3).

$$\begin{aligned}\nabla F(\mathbf{a}) &= \sum_{k=1}^{5} \nabla f_k(\mathbf{a}) f_k(\mathbf{a}) \\ &= \begin{bmatrix} \sum_{k=1}^{5} e^{a_2 t_k}(a_1 e^{a_2 t_k} - y_k) \\ \sum_{k=1}^{5} t_k a_1 e^{a_2 t_k}(a_1 e^{a_2 t_k} - y_k) \end{bmatrix}\end{aligned} \tag{32}$$

If $\mathbf{a}$ from (31) is a true minimum, we would expect that $\nabla F(\mathbf{a}) \sim [0,0]^T$. However, even when using extremely high precision (16d.p.), for the solution obtained in (31)

$$\nabla F(\mathbf{a}) = \begin{bmatrix} 0.000476 \\ 0.011322 \end{bmatrix} \tag{33}$$

which is clearly nonzero. Whilst we may expect some rounding errors in the computation of this minimum, we would expect them to be at around the 14th decimal place, not the third or fourth. It is, however, certainly a minimum of the linearised problem, as (for the linearised problem)

$$\begin{aligned}\nabla \tilde{F}(\mathbf{b}) &= B^T B\mathbf{b} - B^T\mathbf{d} \\ &= \begin{bmatrix} 0.0533 \times 10^{-13} \\ 0.1421 \times 10^{-13} \end{bmatrix}\end{aligned} \tag{34}$$

which, accounting for computational rounding errors with small numbers, is fundamentally zero.

We can confirm this further by using a numerical technique to acquire a minimum for the nonlinear problem, such as the newton method. The code for applying one newton step (using the linear result as an initial guess) is in appendix A.2. Calculating the result gives $\hat{\mathbf{a}} = [2.541069, 0.2595019]^T$, so a

slightly different result to the linearised problem (but the same to 4 d.p.). The gradient at this from the nonlinear problem is

$$F(\hat{\mathbf{a}}) = \begin{bmatrix} -0.0109 \times 10^{-6} \\ -0.301 \times 10^{-6} \end{bmatrix} \tag{35}$$

which is significantly better than the linear solution.

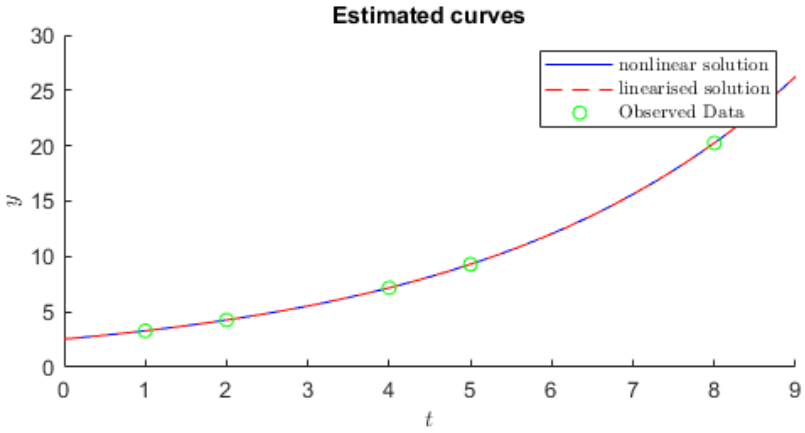

Figure 2: The linear and nonlinear solutions

The curves are almost indistinguishable, however zooming in on the data points reveals that they are certainly different

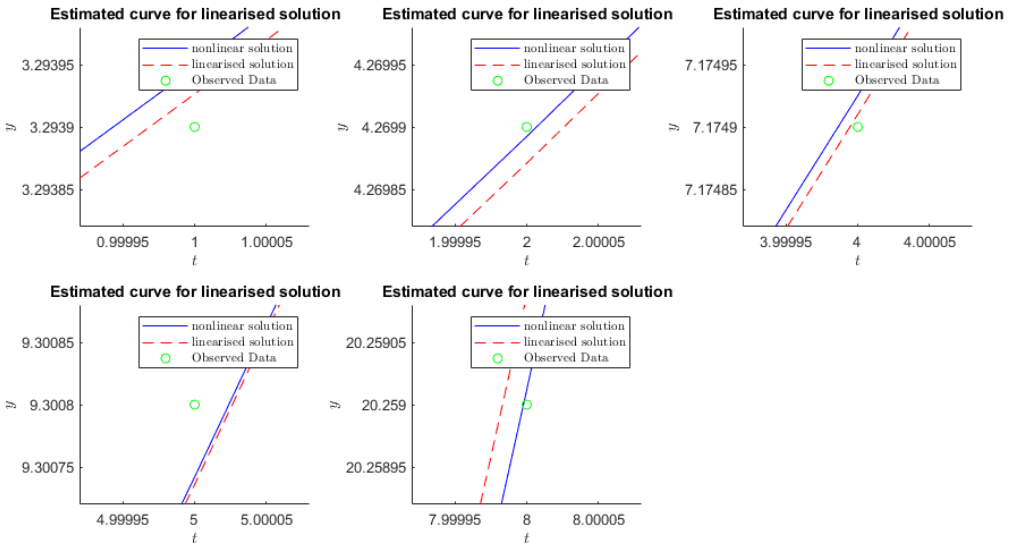

Figure 3: Close ups of the two different solutions against the data

The reason we have this discrepancy is due to how the errors are treated in the linearised model vs the nonlinearised model. The nonlinearised minmisation problem model wants to minimise

$$\mathbf{a}^* = \text{argmin}_{\mathbf{a}} F(\mathbf{a}), \qquad \text{argmin} F(\mathbf{a}) = \frac{1}{2}\sum_{k=1}^{5}(f_k(\mathbf{a}))^2 \tag{36}$$

with

$$f_k(\mathbf{a}) = a_1 e^{a_2 t_k} - y_k, \quad k = 1, ..., 5 \tag{37}$$

If we call the fitted curve y (which is a function of t), then the above is trying to minimise the squares of the *absolute errors* of the system

$$f_k(\mathbf{a}) = y(t_k) - y_k \tag{38}$$

The linearised minimisation problem, however, wants to minimise

$$\mathbf{a}^* = \text{argmin}_{\mathbf{a}} \tilde{F}(\mathbf{a}), \qquad \text{with} \quad \tilde{F}(\mathbf{a}) = \frac{1}{2}\sum_{k=1}^{5}(\tilde{f}_k(\mathbf{a}))^2 \tag{39}$$

where

$$\tilde{f}_k(\mathbf{a}) = b_1 + b_2 t_k - z_k, \quad k = 1, ..., 5 \tag{40}$$

If we call the fitted curve y (which is a function of t), then, since $\ln(y_k) = z_k$ the linearised problem boils down to

$$\begin{aligned} \tilde{f}_k(\mathbf{a}) &= \ln(y(t_k)) - \ln(y_k) \\ &= \ln\left(\frac{y(t_k)}{y_k}\right) \\ &= \ln\left(\frac{y(t_k) + y_k - y_k}{y_k}\right) \\ &= \ln\left(\frac{y(t_k) - y_k}{y_k} + \frac{y_k}{y_k}\right) \\ &= \ln\left(\frac{y(t_k) - y_k}{y_k} + 1\right) \end{aligned} \tag{41}$$

which is more like minimising the squares of the *relative errors* of the system. Therefore, the larger the observed data wanders away from the "perfect curve" the more the two solutions will deviate. Conversely, in the case where $y(t_k) = y_k$ (so all the data lies perfectly on the line) the solutions will be the same. This is why the two curves are so similar - the data is incredibly close to fitting perfectly on the exponential curve. The result of the linearised solution, since it is relatively straightforward to obtain, is mostly useful to make an initial guess before using numerical methods to find the more accurate nonlinear solution.

The other thing to consider is that values do not move uniformly when applying a nonlinear transformation, such as a logarithm. As an example, we can see that y_1 becomes approximately 3 times smaller when the natural log is taken, where as y_5 becomes almost 7 times smaller. So if the errors are large (which they aren't in this case) then the errors will not change uniformly when the transformation is applied. We can see this in action by adding a catastrophically wrong data point.

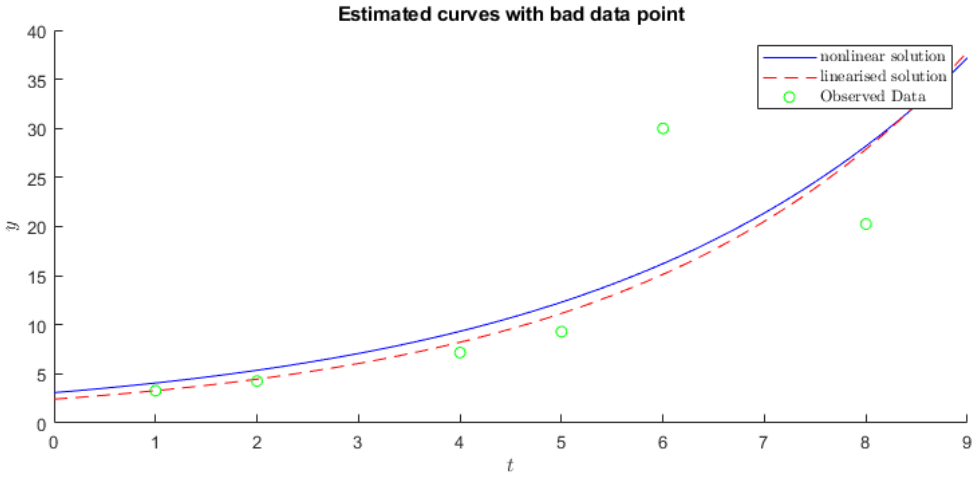

Figure 4: Estimated curves with a new "bad" data point (extreme error)

Adding a new data point $t = 6$, $y = 30$ gives us some better insight into how the curves adjust to this new data with an extreme error. The extreme error is weighted more heavily for the nonlinear approach, and so the curve is more shifted. The linear line is certainly the minimum of the the linear problem, as the minimum at this point gives

$$\nabla \tilde{F}\left(\begin{bmatrix} 0.887 \\ 0.3051 \end{bmatrix} \right) = \begin{bmatrix} -0.1776 \times 10^{-14} \\ 0 \end{bmatrix} \tag{42}$$

However, the same point for the nonlinear system yields

$$\nabla F\left(\begin{bmatrix} 2.4278 \\ 0.3051 \end{bmatrix} \right) = \begin{bmatrix} 7.2282 \\ 488.137 \end{bmatrix} \tag{43}$$

which is certainly not the minimum of the nonlinear problem. These differences make sense when we consider how the errors move when applying the transformation. Taking $a_1 = 2.4278$ and $a_2 = 0.3051$ for the "bad data", this means that the curve is

$$y(t) = 2.4278e^{0.3051t} \tag{44}$$

with the errors looking like

$$\epsilon_{\text{nonlinear}} = y(t_k) - y_k \qquad \epsilon_{\text{linear}} = \ln(y(t_k)) - \ln(y_k) \tag{45}$$

Taking t_1, the nonlinear and linear errors are both to the order $\times 10^{-5}$, so whilst they are different they are comparable. However, the nonlinear error at $t = 6$ (the new bad data point) is around 15, where as it is around 0.7 for the linearised error. These are incredibly different magnitudes, and will therefore effect the systems differently. This is why the nonlinear solutions is effected far more by the extreme errors than the linear problem. The tighter the data fits to the curve, the more similar the curves will become.

Code for plots is in appendix A.1. Code for error calculation is in appendix A.4.

2 Ill-conditioned Systems and Moore-Penrose Inverse

2.1 Weights on a Triangular Element

Question: Suppose unknown weights a_1, a_2, a_3 are associated clockwise with the three vertices of a regular triangle as indicated in the figure. For each vertex the sum of the weights at the two adjacent vertices is measured. Show whether or not the weight at each vertex is uniquely determined from this data.

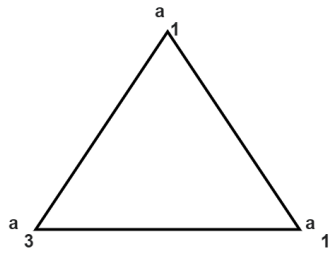

Figure 5: Physical model of triangular element with associated weights

Call v_n the sum of the two weights adjacent to the nth vertex. This creates a set of linear equations

$$\begin{aligned} a_2 + a_3 &= v_1 \\ a_3 + a_1 &= v_2 \\ a_1 + a_2 &= v_3 \end{aligned} \tag{46}$$

which can be represented by the following linear system

$$\begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \tag{47}$$

or $B\mathbf{a} = \mathbf{v}$. If the weight at each vertex can be uniquely determined, the matrix B must be of full rank (3 in this case, as $B \in \mathbb{R}^{3\times 3}$). The rank of B can be determined by looking at the linear independence of each row or column. If B has full rank, we expect every row/column to be linearly independent. This is to say that, given a set of scalars $\{\alpha_k\}_{k=1}^n$, the equation

$$\sum_{i=1}^{n} \alpha_i B_i = 0 \tag{48}$$

can only be satisfied if $\alpha_i = 0$ for all $i = 1, 2, ..., n$, where B_i is the ith column of B. For this example, $n = 3$. If B has rank 3, this means that $\{B_i\}_{i=1}^3$ form a basis of $\mathbb{R}^3$. If B has a rank lower than its dimension, this means that a row/column does that give any new information. This missing information can mean that multiple solutions can satisfy the system, as it can sometimes give us a "free choice" of one or more variables.

In this case, it is pretty clear that all the columns in B are linearly independent, as the only time

$$\alpha_1 \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} + \alpha_2 \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} + \alpha_3 \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = 0 \tag{49}$$

is satisfied is when $\alpha_1 = \alpha_2 = \alpha_3 = 0$, therefore B has rank 3. We can reorganise the system so that

$$\begin{aligned} a_1 &= \frac{v_2 - v_1 + v_3}{2} \\ a_2 &= \frac{v_1 - v_2 + v_3}{2} \\ a_3 &= \frac{v_2 + v_1 - v_3}{2} \end{aligned} \tag{50}$$

which means that if we are given the $\mathbf{v}$ vector then we can find the unique solution. The matrix B also has a well defined determinant ($\det(B) = 2$) and inverse, and so finding $\mathbf{a} = B^{-1}\mathbf{v}$ is also an option here, where

$$B^{-1} = \frac{1}{2}\begin{bmatrix} -1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & -1 \end{bmatrix} \tag{51}$$

which matches the set of linear equations above.

2.2 Weights on a Square Element

Question: Suppose unknown weights a_1, a_2, a_3, a_4 are associated clockwise with the four vertices of a regular quadrilateral as indicated in the figure. For each vertex the sum of the weights at the two adjacent vertices is measured.

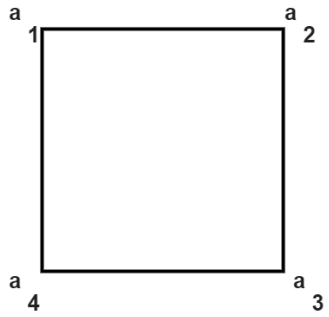

Figure 6: Physical model of square element with associated weights

Call v_n the sum of the two weights adjacent to the nth vertex. This creates a set of linear equations

$$\begin{aligned} a_2 + a_4 &= b_1 \\ a_1 + a_3 &= b_2 \\ a_2 + a_4 &= b_3 \\ a_1 + a_3 &= b_4 \end{aligned} \tag{52}$$

which can be represented by the following linear system

$$\begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} \tag{53}$$

or $D\mathbf{a} = \mathbf{b}$. The first red flag that the weight at each vertex may not be able to be uniquely determined is the fact that, in this case, the determinant of D is 0. This means that we cannot simply find the

value of D^{-1} and pre-multiply both sides of the linear system to find the value of $\mathbf{a}$, as D has no direct inverse.

As discussed in the previous section, we also need to concern ourselves with the linear independence of each of the vectors in D. If D is full rank it will have a rank of 4 as $D \in \mathbb{R}^{4\times 4}$. However, it is clear here that $D_1 = D_3$, $D_2 = D_4$, and there is no nonzero linear combinations that allow any of the odd columns to equal any of the even ones. Therefore rank(D)=2. This means that D is *rank deficient*. This is obvious when we examine the system of linear equations. The equations for b_1 and b_3 tell us the same information, as do the equations for b_2 and b_4. So, although at first it seems that we have four equations and four unknowns, in reality we fundamentally have two equations and four unknowns. We can double check this by reducing the matrix to row echlon form.

$$\left[\begin{array}{cccc|c} 0 & 1 & 0 & 1 & b_1 \\ 1 & 0 & 1 & 0 & b_2 \\ 0 & 1 & 0 & 1 & b_3 \\ 1 & 0 & 1 & 0 & b_4 \end{array}\right] \xrightarrow{r_1 \longleftrightarrow r_2} \left[\begin{array}{cccc|c} 1 & 0 & 1 & 0 & b_2 \\ 0 & 1 & 0 & 1 & b_1 \\ 0 & 1 & 0 & 1 & b_3 \\ 1 & 0 & 1 & 0 & b_4 \end{array}\right]$$

$$\xrightarrow{r_4 - r_1} \left[\begin{array}{cccc|c} 1 & 0 & 1 & 0 & b_2 \\ 0 & 1 & 0 & 1 & b_1 \\ 0 & 1 & 0 & 1 & b_3 \\ 0 & 0 & 0 & 0 & b_4 - b_2 \end{array}\right]$$

$$\xrightarrow{r_3 - r_2} \left[\begin{array}{cccc|c} 1 & 0 & 1 & 0 & b_2 \\ 0 & 1 & 0 & 1 & b_1 \\ 0 & 0 & 0 & 0 & b_3 - b_1 \\ 0 & 0 & 0 & 0 & b_4 - b_2 \end{array}\right]$$

which is the row echelon form of D. Since we have 4 rows, and 2 are all zeroes, we have

$$\text{rank}(D) = 4 - 2 = 2 \tag{54}$$

confirming the intuition above. Therefore, provided $b_1 = b_3$ and $b_2 = b_4$, there are an infinite number of solutions that can satisfy the system (assuming that there is no restriction on the values of a_i). If $b_1 \neq b_3$ or $b_2 \neq b_4$ then there are no solutions for $\mathbf{a}$ that can satisfy the entire system.

2.3 Moore-Penrose Inverse

Question: Assume that the correct (but unknown) four weights of the quadrilateral are

$$\mathbf{a}^T = [a_1, a_2, a_3, a_4]^T = [1, 2, 3, 4]^T \tag{55}$$

In this particular case, the data are given as $\mathbf{b} = [6, 4, 6, 4]^T$ (assuming that no noise is present). For this data set $\mathbf{b}$, calculate the Moore-Penrose (or 'generalized' or 'minimum norm least squares') solution $\mathbf{a}^+$ to the corresponding inverse problem (assuming that you do not know the 'correct' solution $\mathbf{a}$). Provide sufficient details on how you have calculated your result for $\mathbf{a}^+$.

In this case, because D is rank deficient, we are interested in characterising the general form of all possible solutions. This then makes it possible for us to select one which is of the most interest for us (for example, the solution that minimises $||\mathbf{a}||$). In this situation, instead of solving the classic $D\mathbf{a} - \mathbf{b}$ system we solve the least squares problem

$$\min_{\mathbf{a}} ||D\mathbf{a} - b||^2 \tag{56}$$

which has a unique minimiser of minimal norm $\mathbf{a}^+$. This unique minimiser is found by solving

$$\mathbf{a}^+ = D^+\mathbf{b} \tag{57}$$

where $D^+ = V\Sigma^+U^T$ is the Moore-Penrose inverse of D. In the case where $||D\mathbf{a} - b||^2$ has mutliple minimisers, we select the minimum of $||\mathbf{a}||$ as the unique minimiser. Here $D \in \mathbb{R}^{4\times4}$. If $D \in \mathbb{R}^{m\times n}$

1. $V \in \mathbb{R}^{n\times n}$ is orthogonal, and contain the right singular vectors of D as its columns
2. $U \in \mathbb{R}^{m\times m}$ is orthogonal, and contain the left singular vectors of D as its columns
3. $\Sigma \in \mathbb{R}^{m\times n}$ has the singular values of D down the leading diagonal, zeroes elsewhere
4. $\Sigma^+ \in \mathbb{R}^{n\times m}$ has the inverse of $\text{diag}(\sigma_1, ..., \sigma_r)$ as its upper left block matrix, zeroes elsewhere

and $D = U\Sigma V^T$ is the singular decomposition of D, and each $\sigma_1 > \sigma_2 > ... > \sigma_r > 0 = \sigma_{r+1} = ... = \sigma_n$ are the singular values of D. To find the values of U and V we will have to solve the eigenvalue problem

$$A\mathbf{x} = \lambda\mathbf{x} \qquad \text{or} \qquad (A - \lambda I)\mathbf{x} = 0 \tag{58}$$

where $\mathbf{x}$ is an eigenvector of A and λ is the corresponding eigenvalue (a scalar).

2.3.1 Σ^+

The first step we need to take is to calculate the singular values of the matrix D. To begin, calculate the matrix $W = D^TD$, so

$$W = \begin{bmatrix} 2 & 0 & 2 & 0 \\ 0 & 2 & 0 & 2 \\ 2 & 0 & 2 & 0 \\ 0 & 2 & 0 & 2 \end{bmatrix} \tag{59}$$

We then want to calculate the eigenvalues of this matrix (of which there should be 4) and sort them into ascending order. Applying the eigenvalue equation, we therefore need

$$|(W - \lambda I)| = 0 \tag{60}$$

which gives

$$(-(\lambda^2) + 4\lambda)^2 = 0 \tag{61}$$

This means that $\lambda = 4$ or $\lambda = 0$ (twice, as we have two repeated roots here, giving us four values). This gives us the singular values of $\sigma_1 = \sigma_2 = \sqrt{4} = 2$, the rest are 0's. This then allows us to construct Σ such that

$$\Sigma = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{62}$$

We then need to calculate the inverse of the upper left hand matrix (the matrix where the singular values are nonzero)

$$\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}^{-1} = \frac{1}{4}\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \tag{63}$$

Σ^+ is

$$\Sigma^+ = \begin{bmatrix} \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{64}$$

2.3.2 V

To calculate V we need to find the right singular vectors of D. Some of the work has already been done in the previous section. We need to use the eigenvalues from W to find the corresponding eigenvectors.

Starting with $\lambda = 4$, we have

$$W - \lambda I = \begin{bmatrix} 2-\lambda & 0 & 2 & 0 \\ 0 & 2-\lambda & 0 & 2 \\ 2 & 0 & 2-\lambda & 0 \\ 0 & 2 & 0 & 2-\lambda \end{bmatrix} = \begin{bmatrix} -2 & 0 & 2 & 0 \\ 0 & -2 & 0 & 2 \\ 2 & 0 & -2 & 0 \\ 0 & 2 & 0 & -2 \end{bmatrix} \tag{65}$$

for which the corresponding eigenvectors satisfy

$$(W - \lambda I)\mathbf{x} = 0 \tag{66}$$

where $\mathbf{x}$ is the eigenvector. For $\lambda = 4$ (the first two eigenvectors), we get

$$\mathbf{x}_1 = \begin{bmatrix} -\frac{1}{\sqrt{2}} \\ 0 \\ -\frac{1}{\sqrt{2}} \\ 0 \end{bmatrix} \qquad \mathbf{x}_2 = \begin{bmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix} \tag{67}$$

since the equations require that $x_1 = x_3$ and $x_2 = x_4$. The eigenvectors have been normalised.

for $\lambda = 0$ (the last two eigenvectors), we get $x_1 = -x_3$ and $x_2 = -x_4$. We require that the length of the eigenvector is one, calculated by

$$\sqrt{x_1^2 + x_2^2 + x_3^2 + x_4^2} = 1 \tag{68}$$

Therefore, either $x_1 = -x_3 = -\frac{1}{2}$ and $x_2 = -x_4 = -\frac{1}{2}$ or $x_1 = -x_3 = \frac{1}{2}$ and $x_2 = -x_4 = -\frac{1}{2}$, which gives the final two eigenvectors as

$$\mathbf{x}_3 = \begin{bmatrix} -\frac{1}{2} \\ -\frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{bmatrix} \qquad \mathbf{x}_4 = \begin{bmatrix} \frac{1}{2} \\ -\frac{1}{2} \\ -\frac{1}{2} \\ \frac{1}{2} \end{bmatrix} \tag{69}$$

Now that we have all the eigenvectors, we can form V to be

$$V = [\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4] = \begin{bmatrix} -\frac{1}{\sqrt{2}} & 0 & -\frac{1}{2} & \frac{1}{2} \\ 0 & \frac{1}{\sqrt{2}} & -\frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{\sqrt{2}} & 0 & \frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \tag{70}$$

2.3.3 U

From here, we can use the fact that $U\Sigma V^T = D$ to calculate U. Since

$$\begin{aligned} \Sigma V^T &= \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -\frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \end{bmatrix} \\ &= \begin{bmatrix} -\frac{2}{\sqrt{2}} & 0 & -\frac{2}{\sqrt{2}} & 0 \\ 0 & \frac{2}{\sqrt{2}} & 0 & \frac{2}{\sqrt{2}} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{aligned} \tag{71}$$

therefore

$$U\begin{bmatrix} -\frac{2}{\sqrt{2}} & 0 & -\frac{2}{\sqrt{2}} & 0 \\ 0 & \frac{2}{\sqrt{2}} & 0 & \frac{2}{\sqrt{2}} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \tag{72}$$

We can then split this into a system of equations, such as

$$U_{1,1}\left(-\frac{2}{\sqrt{2}}\right) + U_{1,2}(0) + U_{1,3}(0) + U_{1,4}(0) = 0 \tag{73}$$

which gives $U_{1,1} = 0$, or

$$U_{1,1}(0) + U_{1,2}\left(\frac{2}{\sqrt{2}}\right) + U_{1,3}(0) + U_{1,4}(0) = 1 \tag{74}$$

which gives $U_{1,2} = \frac{1}{\sqrt{2}}$. Continuing on in this manner gives

$$U = \begin{bmatrix} 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ -\frac{1}{\sqrt{2}} & 0 & 0 & -\frac{1}{\sqrt{2}} \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ -\frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix} \tag{75}$$

We can do a quick sanity check by inspecting $U\Sigma V^T$

$$U\Sigma V^T = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} = D \tag{76}$$

and therefore this formulation is a SVD of D.

2.3.4 D^+ and $\mathbf{a}^+$

The Moore-Penrose inverse is defined as $D^+ = V\Sigma^+U^T$, which are now all known matrices. This gives

$$\begin{aligned} V\Sigma^+U^T &= \begin{bmatrix} -\frac{1}{\sqrt{2}} & 0 & -\frac{1}{2} & \frac{1}{2} \\ 0 & \frac{1}{\sqrt{2}} & -\frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{\sqrt{2}} & 0 & \frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & -\frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \\ -\frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & -\frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \end{bmatrix} \\ &= \begin{bmatrix} 0 & \frac{1}{4} & 0 & \frac{1}{4} \\ \frac{1}{4} & 0 & \frac{1}{4} & 0 \\ 0 & \frac{1}{4} & 0 & \frac{1}{4} \\ \frac{1}{4} & 0 & \frac{1}{4} & 0 \end{bmatrix} \\ &= D^+ \end{aligned} \tag{77}$$

This can then be applied to the system to find

$$\begin{aligned} \mathbf{a}^+ &= D^+\mathbf{b} \\ &= \begin{bmatrix} 0 & \frac{1}{4} & 0 & \frac{1}{4} \\ \frac{1}{4} & 0 & \frac{1}{4} & 0 \\ 0 & \frac{1}{4} & 0 & \frac{1}{4} \\ \frac{1}{4} & 0 & \frac{1}{4} & 0 \end{bmatrix} \begin{bmatrix} 6 \\ 4 \\ 6 \\ 4 \end{bmatrix} \\ &= \begin{bmatrix} 2 \\ 3 \\ 2 \\ 3 \end{bmatrix} \end{aligned} \tag{78}$$

which satisfies the linear system.

It is clear that solution $\mathbf{a}^+$ does not match the official weights given in $\mathbf{a}$ (in this case). This is because the Moore-Penrose inverse finds the smallest possible value that will satisfy the linear system (or, in the case of there being no solution, the closest thing to a solution) in a Euclidean norm sense. The Euclidean norm is defined as

$$||\mathbf{x}||_2 = \sqrt{x_1^2 + x_2^2 + ... + x_n^2} \qquad \text{for} \quad \mathbf{x} \in \mathbb{R}^n \tag{79}$$

and so we expect that $||\mathbf{a}^+||_2 \leq ||\mathbf{a}||_2$. Calculating these values gives

1. $||\mathbf{a}||_2 = \sqrt{1^2 + 2^2 + 3^2 + 4^2} = \sqrt{30}$
2. $||\mathbf{a}^+||_2 = \sqrt{2^2 + 3^2 + 2^2 + 3^2} = \sqrt{26}$

So the relationship holds, as $\sqrt{26} < \sqrt{30}$. We also expect the difference between two solutions to be in the null space of D, so

$$D(\mathbf{a} - \mathbf{a}^+) = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ -1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{80}$$

which also holds. This answer was checked against Matlab, code for which is in appendix A.3.

A Matlab Code

A.1 Least Squares Plotting

```
%% system values
B = [1, 1;
    1, 2;
    1, 4;
    1, 5;
    1, 8];

d = [1.1921;1.4516;1.9706;2.2301;3.0086];

%% solution to the system
b = inv(B'*B)*B'*d;

disp('b1')
b(1)
disp('b2')
b(2)
disp('corresponding a1')
exp(b(1))
disp('corresponding a2')
b(2)

%% plot stuff
% plot observed data
tk = [1, 2, 4, 5, 8];
yk = [3.2939, 4.2699, 7.1749, 9.3008, 20.259];
figure(1)
for n=1:length(tk)
    subplot(2, 3, n)
    hold on
    t = [0:0.1:9];
    y = @(a1, a2) a1*exp(a2.*t);
    %plot value of nonlinear solution
    lecr = y(2.541069136837578, 0.259501850448539);
    plot(t, lecr, '-b')
    %plot value of linear solution
    cour = y(exp(b(1)), b(2));
    plot(t, cour, '--r');
    hold on
    plot(tk, yk, 'og')
    % label axis
    xlabel('$t$', 'interpreter', 'Latex')
    ylabel('$y$', 'interpreter', 'Latex')
    title('Estimated curve for linearised solution')
    legend('nonlinear solution', 'linearised solution', 'Observed Data', ...
    'interpreter', 'Latex')

    %limit plot to data point
    xlim([tk(n)-0.00008, tk(n)+0.00008]);
    ylim([yk(n)-0.00008, yk(n)+0.00008]);
end

%% Just checking some values
detB = 5*(1^2+2^2+4^2+5^2+8^2) - (1+2+4+5+8)^2;
coef1= (1^2+2^2+4^2+5^2+8^2)*(1.1921+1.4516+1.9706+2.2301+3.0086);
coef2 = -(1+2+4+5+8)*(1.1921+2*1.4516+4*1.9706+5*2.2301+8*3.0086);

(1/detB)*(coef1+coef2);

%% plot entire curve
figure(2)
hold on
```

```
t = [0:0.1:9];
y = @(a1, a2) a1*exp(a2.*t);
lecr = y(2.541069136837578, 0.259501850448539);
plot(t, lecr, '-b')
exp(b(1))
b(2)
ans = [exp(b(1)), b(2)];
cour = y(exp(b(1)), b(2));
plot(t, cour, '--r');
hold on
plot(tk, yk, 'og')
xlabel('$t$', 'interpreter', 'Latex')
ylabel('$y$', 'interpreter', 'Latex')
title('Estimated curves')
legend('nonlinear solution', 'linearised solution', 'Observed Data', ...
    'interpreter', 'Latex')
```

A.2 Newton Method (one step)

```
tk = [1, 2, 4, 5, 8];
yk = [3.2939, 4.2699, 7.1749, 9.3008, 20.259];
a=[2.5, -0.75]
GNnab2Fex(a(1), a(2), tk, yk) + GNnab2F(a(1), a(2), tk)
nabF(a(1), a(2), tk, yk)
for n=1:1
    a=[2.5411; 0.2595];
    p = (GNnab2Fex(a(1), a(2), tk, yk) + GNnab2F(a(1), a(2), tk))...
        \(-nabF(a(1), a(2), tk, yk));
    a = a + 1*p;
end
disp('estimated minimiser for nonlinear systen')
a
disp('gradient for newton step')
nabF(a(1), a(2), tk, yk)
disp('gradient at linear solution')
nabF(2.541107475549933, 0.2595, tk, yk)

% find grad for a1 and a2
function [F] = nabF(a1, a2, tk, yk)
    F = zeros(2,1);
    for n=1:length(tk)
        F(1) = F(1) + exp(a2*tk(n))*(a1*exp(a2*tk(n)) - yk(n));
        F(2) = F(2) + tk(n)*a1*exp(a2*tk(n))*(a1*exp(a2*tk(n))-yk(n));
    end
end

% find Gauss newton approx of hessian
function [F] = GNnab2F(a1, a2, tk)
    F = zeros(2,2);
    for n=1:length(tk)
        F(1, 1) = F(1, 1) + (exp(a2*tk(n)))^2;
        F(1, 2) = F(1, 2) + (exp(a2*tk(n)))^2*tk(n)*a1;
        F(2, 1) = F(2, 1) + (exp(a2*tk(n)))^2*tk(n)*a1;
        F(2, 2) = F(2, 2) + (exp(a2*tk(n)))^2*tk(n)^2*a1^2;
    end
end

% extension to full hessian
function [F] = GNnab2Fex(a1, a2, tk, yk)
    F = zeros(2,2);
    for n=1:length(tk)
        F(1, 2) = F(1, 2) + tk(n)*exp(a2*tk(n))*(a1*exp(a2*tk(n))-yk(n));
        F(2, 1) = F(2, 1) + tk(n)*exp(a2*tk(n))*(a1*exp(a2*tk(n))-yk(n));
        F(2, 2) = F(2, 2) + tk(n)^2*exp(a2*tk(n))*(a1*exp(a2*tk(n))-yk(n));
    end
```

```
end
```

A.3 Moore-Penrose Inverse

```
syms a1 a2 a3 a4 v1 v2 v3 v4

%% triangular system
disp('--- triangular weights system ---')
A=[0, 1, 1;
    1, 0, 1;
    1, 1, 0];
a=[a1;a2;a3];
v=[v1;v2;v3];
disp('Determinant of A')
det(A)
disp('Rank of A')
rank(A)
disp('Unique solution for system')
invA=inv(A);
invA*v

%% square system
disp('--- square weights system ---')
A = [0, 1, 0, 1;
    1, 0, 1, 0;
    0, 1, 0, 1;
    1, 0, 1, 0];
a = [a1;a2;a3;a4];
v=[6;4;6;4];
disp('Determinant of A')
det(A)
disp('Rank of A')
rank(A)
disp('Moore-Penrose psuedo inverse of A')
Am = pinv(A)
disp('System solution')
b = Am*v
disp('singular value decomposition of A')
format rational
[U S V] = svd(A)
%% determ
syms L  L1 L2 L3 L4
W = A'*A;
det(W - L*eye(4));

[a, b] = eigs(W);

%%
```

A.4 Error Calculation

```
%% calculate errors for linear/nonlinear system
y = @(t) 2.4278*exp(0.3051*t);
y5 = 30;
y1 = 3.2939;
t1 = 1;
t5 = 6;
y(t5) - y5
log(y(t5)) - log(y5)
y(t1) - y1
log(y(t1)) - log(y1)
```