3 Least squares problems

In this chapter, we consider problems where there are more linear equations than unknowns, or equivalently, more constraints than variables. Such problems arise, for example, when we try to find parameters that 'best fit' a model to a data set.

To help with the discussion, consider a representative example of finding a line of best fit for given data points (x_1, y_1), (x_2, y_2), ..., (x_n, y_n). To determine such a line, we want to determine a_0 and a_1 such that $l(x) = a_0 + a_1 x$ 'best fits' the data. For this problem, we have 2 unknowns, a_0 and a_1, but we have n equations we wish to satisfy:

$$y_i = l(x_i), \quad i = 1, 2, 3, \ldots, n$$

Combining these equations into a matrix equation, we obtain

$$\begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \iff \mathbf{Ax} = \mathbf{b}.$$

It is highly unlikely that there can be a solution to these n equations, unless the n points happened to all lie on a single line. So instead of looking for an exact solution, we will look for a 'least squares' solution in the sense that we want to find a_0 and a_1 so that the *least squares error*

$$e(a_0, a_1) = \sum_{i=1}^{n} (y_i - (a_0 + a_1 x_i))^2$$

is minimized. Note there are different ways to minimize, but typically for data fitting problems, least squares error minimization is both intuitive and has good mathematical properties.

Since equality is not expected here, when writing least squares problems, we typically use the notation $\cong$ instead of $=$, and so write the system as

$$\mathbf{Ax} \cong \mathbf{b},$$

with the understanding that this means minimizing the sum of squares error. A nice mathematical property of using least squares error is that it is the same as minimizing the Euclidean norm of the residual $\mathbf{Ax}-\mathbf{b}$. In fact simply expanding out $\|\mathbf{Ax}-\mathbf{b}\|_2^2$ reveals

$$\|\mathbf{Ax}-\mathbf{b}\|_2^2 = \sum_{i=1}^{n}(y_i-(a_0+a_1x_i))^2.$$

Thus, by defining the least squares (LSQ) problem as

$$\mathbf{Ax}\cong\mathbf{b} \iff \mathbf{x} \text{ minimizes } \|\mathbf{Ax}-\mathbf{b}\|_2,$$

we have that it is the exact same thing as minimizing the sums of squares error:

$$\begin{aligned}\mathbf{Ax}\cong\mathbf{b} &\iff \mathbf{x} \text{ minimizes } \|\mathbf{Ax}-\mathbf{b}\|_2\\ &\iff \mathbf{x} \text{ is the solution of } \min_{\mathbf{x}}\|\mathbf{Ax}-\mathbf{b}\|_2^2\\ &\iff (a_0,a_1) \text{ is the solution of } \min_{a_0,a_1}\sum_{i=1}^{n}(y_i-(a_0+a_1x_i))^2.\end{aligned}$$

Hence, we observe that solving the LSQ problem is exactly the same as minimizing the squared distances of the data points from the predicted line.

3.1 Solving LSQ problems with the normal equations

We consider now a linear system of n equations and m unknowns that can be written as

$$\mathbf{A}_{n\times m}\mathbf{x}_{m\times 1}\cong\mathbf{b}_{n\times 1}.$$

We assume that $n>m$, and $\mathbf{A}$ has full column rank (i.e., rank$(\mathbf{A})=m$). This assumption corresponds to the parameters being independent of each other, which is typically a safe assumption. For example, in a line of best fit, one wants to find a_0 and a_1 that best fit a line

$$l(x)=a_0+a_1x$$

to a data set. Here $m=2$ and we would have full column rank (see the matrix $\mathbf{A}$ above). But if we changed the problem to instead look for coefficients of 1, x, and also $(x+1)$, then we would have $m=3$ but a column rank of only 2. This is because if we tried to use a_0, a_1, and a_2 to best fit a line

$$l(x)=a_0+a_1x+a_2(x+1)$$

to a set of data, column 3 of the matrix would be a linear combination of the first two columns. Hence, for LSQ problems, the assumption of full column rank is reasonable since if the columns are not linearly independent, then typically this can be fixed by eliminating redundant columns and unknowns.

We now derive a solution method for the LSQ problem $\mathbf{Ax} \cong \mathbf{b}$. Since this problem is defined to be finding $\mathbf{x}$ that minimizes $\|\mathbf{Ax}-\mathbf{b}\|_2$ and with that also $\|\mathbf{Ax}-\mathbf{b}\|_2^2$, we define the function

$$h(\mathbf{x}) = \|\mathbf{Ax}-\mathbf{b}\|_2^2.$$

Expanding this norm using its definition, we obtain

$$h(\mathbf{x}) = (\mathbf{Ax}-\mathbf{b})^T(\mathbf{Ax}-\mathbf{b}) = \mathbf{x}^T\mathbf{A}^T\mathbf{Ax} - \mathbf{x}^T\mathbf{A}^T\mathbf{b} - \mathbf{b}^T\mathbf{Ax} + \mathbf{b}^T\mathbf{b}.$$

To minimize $h(\mathbf{x})$, a key point is to notice that $h(\mathbf{x})$ is a quadratic function in $x_1,\ x_2, ...,\ x_m$, and moreover, the coefficients of $x_1^2,\ x_2^2, ...,\ x_m^2$ in $h(\mathbf{x})$ must be positive (see Exercise 1). A quadratic function with positive coefficients of these terms is an upward facing paraboloid (in $\mathbb{R}^m$), and thus must have a unique minimum at its vertex. Hence we can solve the LSQ problem by finding the vertex of the paraboloid defined by $h(\mathbf{x})$, and this can be done by setting $\nabla h(\mathbf{x}) = \mathbf{0}$ and solving for $\mathbf{x}$ (i.e. set each first partials $\frac{\partial h}{\partial x_i}$ to zero and solve).

A calculation (see Exercise 2) shows that

$$\nabla h(\mathbf{x}) = 2\mathbf{A}^T\mathbf{Ax} - 2\mathbf{A}^T\mathbf{b},$$

and so the LSQ solution $\mathbf{x}$ must satisfy

$$(\mathbf{A}^T\mathbf{A})\mathbf{x} = \mathbf{A}^T\mathbf{b}$$

Since $\mathbf{A}$ has full column rank, $\mathbf{A}^T\mathbf{A}$ is square, SPD, and therefore nonsingular (see Exercise 3). Thus, we can solve this $m \times m$ linear system to get the LSQ solution! That is, we have transformed the minimization problem $\mathbf{Ax} \cong \mathbf{b}$ into the square linear system $(\mathbf{A}^T\mathbf{A})\mathbf{x} = \mathbf{A}^T\mathbf{b}$.

We now perform an example to demonstrate the theory above.

Example 15. *Find the line of best fit for the data points*

```
xy = [
     1.0000    1.0000
     1.9000    1.0000
     2.5000    1.1000
     3.0000    1.6000
     4.0000    2.0000
     7.0000    3.4500
]
```

We proceed to find the best fit coefficients a_0 and a_1. Following the derivation above, this defines a LSQ problem $\mathbf{Ax} \cong \mathbf{b}$. We can input $\mathbf{A}$ and $\mathbf{b}$ easily into MATLAB using the data points:

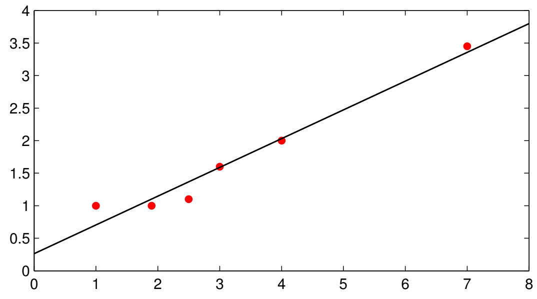
```
>> A = [xy(:,1).^0, xy(:,1).^1]
A =
        1.0000      1.0000
        1.0000      1.9000
        1.0000      2.5000
        1.0000      3.0000
        1.0000      4.0000
        1.0000      7.0000

>> b = xy(:,2)
b =
        1.0000
        1.0000
        1.1000
        1.6000
        2.0000
        3.4500
```

Next, solve the system using the normal equations $(\mathbf{A}^T\mathbf{A})\mathbf{x} = \mathbf{A}^T\mathbf{b}$*:*

```
>> x = (A' * A) \ (A' * b)
x =
        0.2627
        0.4419
```

We now have the coefficients, $a_0 = 0.2627$ *and* $a_1 = 0.4419$*. As a sanity check, we plot the line* $a_0 + a_1x$ *along with the data points and see what we expected: a line of best fit!*



3.2 QR factorization and the Gram Schmidt process

Solving a LSQ problem, as done above, is called solving with the normal equations, since the matrix $\mathbf{A}^T\mathbf{A}$ is a 'normal' matrix. For smaller problems, this is generally effective. For problems where m is large (greater than a few thousand or so), the linear solve can be difficult because $\mathbf{A}^T\mathbf{A}$ is often fairly dense, even if $\mathbf{A}$ is sparse. Further, this linear solve is highly prone to conditioning problems. To see this, consider for simplicity a square, symmetric, nonsingular matrix $\mathbf{M}$. Here, one can calculate under properties of normal matrices that

$$\text{cond}_2(\mathbf{M}^T\mathbf{M}) = \|\mathbf{M}^T\mathbf{M}\|_2\|(\mathbf{M}^T\mathbf{M})^{-1}\|_2 = \|\mathbf{M}\|_2^2\|\mathbf{M}^{-1}\|_2^2 = (\text{cond}_2(\mathbf{M}))^2.$$

Although this is the simplest case, it is a similar situation for rectangular matrices (we would have to build up notation and theory for conditioning of rectangular matrices), but the takeaway is this: if a matrix $\mathbf{M}$ is even mildly ill-conditioned, then $\mathbf{M}^T\mathbf{M}$ is very ill-conditioned. We next show a way to solve the LSQ problems without performing a linear solve with a potentially ill-conditioned matrix.

The Gram Schmidt process is a method to transform a set of m linearly independent vectors into an orthonormal set of m vectors that have the same span. Recalling from Calculus III the notion of a vector projection of $\mathbf{v}$ onto $\mathbf{u}$ (the part of $\mathbf{v}$ that is in the direction of $\mathbf{u}$),

$$\text{proj}_{\mathbf{u}}(\mathbf{v}) = \frac{\mathbf{u}\cdot\mathbf{v}}{\|\mathbf{u}\|^2}\mathbf{u},$$

a linearly independent set of vectors $\{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, ...\}$ can be transformed into an orthogonal set by the process

$$\begin{aligned}
\mathbf{a}_1 &= \mathbf{a}_1,\\
\mathbf{a}_2 &= \mathbf{a}_2 - \text{proj}_{\mathbf{a}_1}(\mathbf{a}_2),\\
\mathbf{a}_3 &= \mathbf{a}_3 - \text{proj}_{\mathbf{a}_1}(\mathbf{a}_3) - \text{proj}_{\mathbf{a}_2}(\mathbf{a}_3),
\end{aligned}$$

and so on. After each step, each $\mathbf{q}_i$ is normalized via

$$\mathbf{q}_i = \frac{\mathbf{a}_i}{\|\mathbf{a}_i\|}.$$

For numerical stability, we alter the process to the following mathematically equivalent process:

$$\begin{aligned}
\mathbf{q}_1 &= \mathbf{a}_1\\
\mathbf{q}_1 &= \mathbf{q}_1/\|\mathbf{q}_1\|\\
\mathbf{q}_2 &= \mathbf{a}_2\\
\mathbf{q}_2 &= \mathbf{q}_2 - (\mathbf{q}_1\cdot\mathbf{q}_2)\mathbf{q}_1\\
\mathbf{q}_2 &= \mathbf{q}_2/\|\mathbf{q}_2\|\\
\mathbf{q}_3 &= \mathbf{a}_3\\
\mathbf{q}_3 &= \mathbf{q}_3 - (\mathbf{q}_1\cdot\mathbf{q}_3)\mathbf{q}_1\\
\mathbf{q}_3 &= \mathbf{q}_3 - (\mathbf{q}_2\cdot\mathbf{q}_3)\mathbf{q}_2\\
\mathbf{q}_3 &= \mathbf{q}_3/\|\mathbf{q}_3\|
\end{aligned}$$

and so on.

One can easily check that the set $\{\mathbf{q}_i\}_{i=1,\ldots,n}$ is orthonormal. Considering the set of vectors as columns of a matrix $\mathbf{A}$, the Gram-Schmidt process transforms it into a matrix $\mathbf{Q}$, where the columns of $\mathbf{Q}$ are orthonormal and have the same span as the columns of $\mathbf{A}$.

Note that if we consider $\mathbf{A}$ to be the matrix with columns $\mathbf{a}_i$, then each step of the Gram-Schmidt algorithm is an operation using just the columns of (modified) $\mathbf{A}$. This is a fairly straightforward MATLAB algorithm, which we give below. Note that here we normalize the $\mathbf{q}_i$'s after the orthogonalization process, before moving on to the next column. We also keep track of the multipliers (scalar projections) $r_{ij} = \mathbf{q}_i \cdot \mathbf{a}_j$, as keeping track of these will allow us to reverse the process, which results in a matrix factorization.

```
function [Q,R] = gramschmidt_QR(A)
    Q = A;
    n = size(Q,2);
    R = zeros(n,n);
    for j = 1 : n
        q = A(:,j);
        for k = 1 : j-1
            qk = Q(:,k);
            R(k,j) = (qk'*q);
            q = q - qk*R(k,j);
        end
        R(j,j) = norm(q,2);
        Q(:,j) = q / R(j,j);
    end
```

We test the algorithm below on three random vectors of length 5, which are stored in a 5×3 matrix A. One can check that the vectors are orthonormal by checking that $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$, since the i,j component of $\mathbf{Q}^T\mathbf{Q}$ is $\mathbf{q}_i \cdot \mathbf{q}_j$.

```
>> A = rand(5,3)
A =

    0.8407    0.3500    0.3517
    0.2543    0.1966    0.8308
    0.8143    0.2511    0.5853
    0.2435    0.6160    0.5497
    0.9293    0.4733    0.9172

>> Q = gramschmidt_QR(A)
Q =

    0.5476   -0.1063   -0.4617
    0.1656    0.1400    0.8493
    0.5304   -0.2697    0.0545
    0.1586    0.9456   -0.1711
    0.6052    0.0465    0.1824

>> Q' * Q
ans =

    1.0000    0.0000   -0.0000
    0.0000    1.0000    0.0000
   -0.0000    0.0000    1.0000
```

Interestingly, the transformation of the matrix $\mathbf{A}$ to $\mathbf{Q}$ involves only rescaling of columns, and replacing columns with their sum with another column. This means that each operation is a linear transformation, and moreover, these linear transformations can be combined together into a matrix. In particular, if we know $\mathbf{Q}$ (which we do from the algorithm), then we can easily undo the Gram-Schmidt operations using the upper triangular matrix $\mathbf{R}$, which holds the scalar projections that were created from the orthogonalization process. This creates a factorization of the form

$$\mathbf{A}_{n\times m} = \mathbf{Q}_{n\times m}\mathbf{R}_{m\times m}.$$

This $\mathbf{A} = \mathbf{QR}$ factorization turns out to be quite useful for solving the LSQ problem $\mathbf{Ax} \cong \mathbf{b}$. Since the unique LSQ solution satisfies

$$\mathbf{A}^T\mathbf{Ax} = \mathbf{A}^T\mathbf{b},$$

inserting the factorization yields

$$\mathbf{R}^T\mathbf{Q}^T\mathbf{QRx} = \mathbf{R}^T\mathbf{Q}^T\mathbf{b}.$$

Since $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}_{m\times m}$ and $\mathbf{R}$ is nonsingular, since $\mathbf{A}$ has full column rank, the equation reduces to

$$\mathbf{Rx} = \mathbf{Q}^T\mathbf{b}.$$

This solve with $\mathbf{R}$ is fast and easy since $\mathbf{R}$ is upper triangular, and moreover turns out to be much better for conditioning, compared to using the normal equations. To see this, again take for simplicity the case of a square symmetric matrix $\mathbf{A}$, we have that $\text{cond}_2(\mathbf{R}) = \text{cond}_2(\mathbf{A})$, since $\mathbf{Q}^{-1} = \mathbf{Q}^T$ and $\text{cond}_2(\mathbf{Q}) = 1$, and thus, solving the LSQ problem this way is not ill-conditioned unless $\mathbf{A}$ is ill-conditioned (as opposed to solving using the normal equations).

There are several algorithms to compute QR factorizations, such as the Householder reflection or Givens rotations, with each having advantages over the others in different situations. As a general rule, it is safe to use MATLAB's QR factorization process based on Householder reflections, which can be called via `[Q,R] = qr(A,0)`. The second argument `0` is to have it produce $\mathbf{Q}_{n\times m}$ instead of a full square $\mathbf{Q}_{n\times n}$, where the columns $m+1,\ m+2, ...,\ n$ are created so that $\mathbf{Q}_{n\times n}$ is a square orthogonal matrix - this is unnecessary and wasteful, if one is only interested in using the QR factorization to solve the LSQ problem.

3.3 Curve of best fit

Since many data relations are not linear, it is important to also consider 'curves of best fit'. The ideas in the previous sections can be extended to this setting without much difficulty. There are three types of two-parameter curves to consider:

1. exponential: $y = be^{mx}$
2. inverse: $y = \frac{1}{mx+b}$
3. power: $y = bx^m$

Each of these functions depend on two parameters, which we call m and b. It is our goal to find the m and b that provide a curve of best fit for given data points (x_i, y_i), $i = 1, 2, ..., n$. The ideas below can easily be extended to other types of two-parameter curves. The procedure is to find a transformation that creates a linear relationship in transformed data, fit a line to the transformed data, and finally un-transform the line to get a curve of best fit.

Step 1 of this procedure is the hardest, as it requires us to look at data and determine what kind of function that data might represent. If the data is exponential, then we would expect the points $(x_i, \log(y_i))$ to look linear. Similarly, if the data comes from an inverse function, then we would expect $(x, \frac{1}{y})$ to appear linear. Hence, Step 1 requires some educated guess, and checking that the transformed data is indeed linear.

Step 2 of the procedure is clear. Once we have linearized data, we already know how to find the line of best fit for it.

Step 3 is to take the line of best fit for the transformed data and turn it into a curve for the original data. How to do this depends on the transformation, but for example, for the exponential case, we fit a line $\log(y) = a_0 + a_1 x$, then to un-transform, we raise e to both sides and get $y = e^{a_0 + a_1 x} = e^{a_0} e^{a_1 x}$. If we set $b = e^{a_0}$ and $m = a_1$, we now have our parameters for a curve of best fit.

The transformation process for each of these data types is as follows:

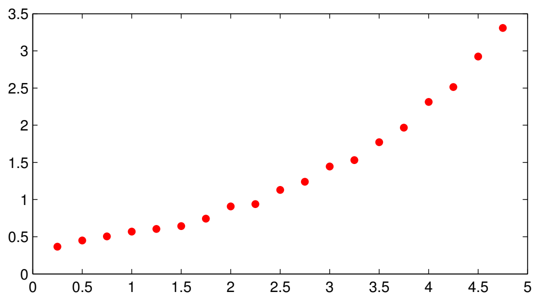
1. If the data comes from $y = be^{mx}$, then taking the log of both sides gives $\log(y) = \log(b) + mx$. Thus, a plot of x_i vs. $\log(y_i)$ will look linear.

2. If the data comes from $y = \frac{1}{mx+b}$, then taking reciprocals of both sides gives $y^{-1} = mx + b$. So a plot of x_i vs. y_i^{-1} will look linear.
3. If the data comes from $y = bx^m$, then taking the log of both sides gives $\log(y) = \log(b) + m\log(x)$. Thus, a plot of $\log(x_i)$ vs. $\log(y_i)$ will look linear.

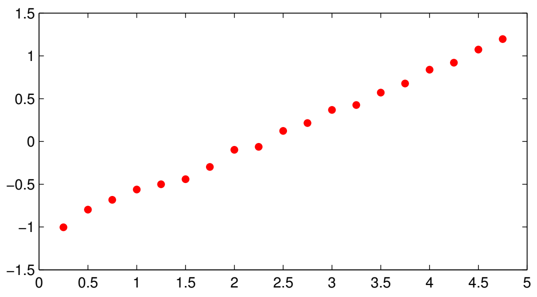
Example 16. *Fit a curve to the data points.*

```
>> xy = [
      0.2500    0.3662
      0.5000    0.4506
      0.7500    0.5054
      1.0000    0.5694
      1.2500    0.6055
      1.5000    0.6435
      1.7500    0.7426
      2.0000    0.9068
      2.2500    0.9393
      2.5000    1.1297
      2.7500    1.2404
      3.0000    1.4441
      3.2500    1.5313
      3.5000    1.7706
      3.7500    1.9669
      4.0000    2.3129
      4.2500    2.5123
      4.5000    2.9238
      4.7500    3.3070
]
```

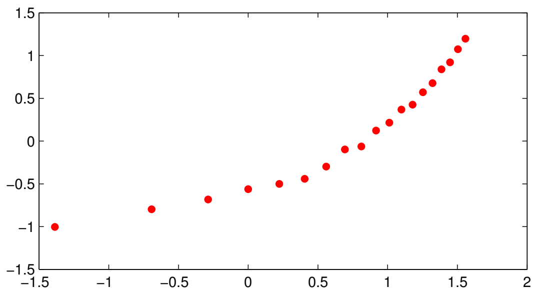
Our first step is to plot the data points (x_i, y_i), as follows



By examining the data, we suspect this could be either an exponential or a power function. Thus we plot the transformed data $(x_i, \log(y_i))$:



and $(\log(x_i), \log(y_i))$*:*



From these plots, we conclude the data most likely comes from an exponential function. The next step is to fit a line of best fit to the transformed (linear) data $(x_i, \log(y_i))$*, so we use the line of best fit procedure with* $\log(y_i)$ *in place of* y_i*.*

```
>> A = [xy(:,1).^0, xy(:,1).^1];    b = log(xy(:,2));
>> [Q,R] = qr(A,0);
>> x = R \ (Q'*b)
x =
   -1.0849
    0.4752
```

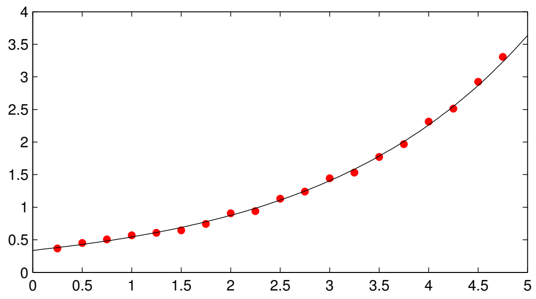
Thus, we have fit a line of best fit to the data $(x_i, \log(y_i))$*, and it is given by*

$$\log(y) = -1.0849 + 0.4752x$$

Now we un-transform the line (into a curve) by raising e to both sides to get

$$y = e^{-1.0849}e^{0.4752x} = 0.3379e^{0.4752x}$$

Since the line was a line of best fit for the transformed data, this will be a curve of best fit for the original data, which we can see in the plot below.



Let us now summarize the different curves to fit:

name:	**equation:**	**fit line ($y = a_0 + a_1 x$) to:**	**answer:**
line	$y = b + mx$	(x_i, y_i)	$b = a_0,\ m = a_1$
exponential	$y = b \cdot e^{mx}$	$(x_i, \log y_i)$	$b = e^{a_0},\ m = a_1$
power	$y = b \cdot x^m$	$(\log x_i, \log y_i)$	$b = e^{a_0},\ m = a_1$
inverse	$y = (b + mx)^{-1}$	$(x_i, 1/y_i)$	$b = a_0,\ m = a_1$

3.4 Exercises

1. Let $\mathbf{x} = [x_1\ x_2]^T$. Prove that the coefficients of x_1^2 and x_2^2 that arise in the polynomial $\mathbf{x}^T(\mathbf{A}^T\mathbf{A})\mathbf{x}$ are positive (you may assume that $\mathbf{A}$ is 3×2).
2. For $\mathbf{A}_{3\times 2}$, $\mathbf{x}_{2\times 1}$, let $h(\mathbf{x}) = \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2$. Show that $\nabla h(\mathbf{x}) = 2\mathbf{A}^T(\mathbf{A}\mathbf{x} - \mathbf{b})$.
3. If $\mathbf{A}$ has full column rank, prove that $(\mathbf{A}^T\mathbf{A})$ is symmetric positive definite.
4. Prove that, for a matrix norm induced by a vector norm, $\|\mathbf{AB}\| \leq \|\mathbf{A}\|\|\mathbf{B}\|$ for any matrices $\mathbf{A}$ and $\mathbf{B}$ where $\mathbf{AB}$ is well defined.
5. Calculate $\text{cond}_2(\mathbf{A})$ and $\text{cond}_2(\mathbf{A}^T\mathbf{A})$ for

$$\mathbf{A} = \begin{pmatrix} 3.0170 & 1.8236 \\ -5.8075 & -3.5070 \end{pmatrix}.$$

   Is the condition number of the normal matrix approximately the square of the matrix condition number?
6. Consider fitting a nonlinear curve to the data points

```
x = [-1.0000   0    0.5000   1.0000   2.2500   3.0000   3.4000]
y = [0.3715  1.0849  1.7421  2.7862  9.5635  20.1599  30.0033]
```

(a) For the different transformation in this chapter, plot the corresponding transformed data, and decide which looks closest to linear.
(b) Find a curve of best fit for the data.

7. Solve LSQ problems to find the line of best fit, quadratic of best fit, cubic of best fit, and quartic of best fit for the points

```
    0.1622    0.0043
    0.7943    0.5011
    0.3112    0.0301
    0.5285    0.1476
    0.1656    0.0045
    0.6020    0.2181
    0.2630    0.0182
    0.6541    0.2798
    0.6892    0.3274
    0.7482    0.4188
```

Plot them all together on [0.1622, 0.7943]. Which is best?

8. Consider the matrix $\mathbf{A}$ defined by

```
A = [
    0.7577    0.7060    0.8235    0.4387    2.7260
    0.7431    0.0318    0.6948    0.3816    1.8514
    0.3922    0.2769    0.3171    0.7655    1.7518
    0.6555    0.0462    0.9502    0.7952    2.4471
    0.1712    0.0971    0.0344    0.1869    0.4896
]
```

and vector $\mathbf{b}$ defined to be a 5×1 vector of all 1's. Solve $\mathbf{Ax} = \mathbf{b}$ two ways:

– Using the normal equations, i.e. solve $\mathbf{A}^T\mathbf{Ax} = \mathbf{A}^T\mathbf{b}$
– Using a QR factorization

Which is the most accurate (compare $\|\mathbf{b} - \mathbf{Ax}\|_2$ for each solution)? Explain the differences using the condition numbers of the associated matrices.