MAT 343 Laboratory 5
Least Squares

In this laboratory session we will learn how to

1. Factor a symmetric matrix using the Cholesky decomposition.
2. Solve Least Squares problems using MATLAB
3. Plot the data points together with the least squares approximation.

Introduction

Let S be a symmetric matrix. Since S is square (why?), it has an LU decomposition. In fact, the symmetry makes it possible to write S in the form $S = LL^T$, where L is a lower-triangular matrix. This is called the *Cholesky decomposition* of S, and it is defined only for symmetric matrices. Given a symmetric matrix, the Cholesky decomposition is always preferable in computations because it can be computed in half the time of the LU decomposition.
There is nothing special about writing the Cholesky decomposition in terms of a lower-triangular matrix. We can just as well write $S = U^TU$, where $U = L^T$. In MATLAB, the command

```
U = chol(S)
```

returns the Cholesky decomposition of the symmetric matrix S in the upper-triangular matrix U.
Once the Cholesky decomposition $S = U^TU$ is found, solving a system of the form

$$S\mathbf{c} = \mathbf{z}$$

proceeds in the same 2-step manner as solving $A\mathbf{x} = \mathbf{b}$ with the LU decomposition:

1. Solve $U^T\mathbf{w} = \mathbf{z}$.
2. Solve $U\mathbf{c} = \mathbf{w}$.

The normal equations for the least squares

Suppose we are given a collection of data points of the form $\{(x_i, y_i)\}_{i=1}^n$. The simplest potential statistical relationship between x and y is that of a straight line; more precisely,

$$y_i = c_1 + c_2x_i \tag{L5.1}$$

where c_1 and c_2 give the y-intercept and slope, respectively. Of course, the data points do not, in general, lie on this line. The difference

$$y_i - (c_1 + c_2x_i) = \epsilon_i \tag{L5.2}$$

is the "residual" or "noise", that is, the amount by which $c_1+c_2x_i$ fails to coincide with y_i. Our first goal is to estimate the unknowns c_1 and c_2 from the data. Although MATLAB provides an entire toolbox for statistical analysis, we will use only a few simple commands to get started; the pedagogical purpose is to make sure that you understand how the problem is set up mathematically. Equation (L5.2) can be extended to any polynomial model. For example, if we suppose that

$$y_i = c_1 + c_2x_i + c_3x_i^2 + \ldots + c_px_i^{p-1} + \epsilon_i \tag{L5.3}$$

then we have to estimate the p unknowns $c_1, c_2, \ldots, c_p$ so that ϵ_i is small for all i. Statisticians write the estimation problem in the following form:

$$\mathbf{y} = X\mathbf{c} + \epsilon \tag{L5.4}$$

where $\mathbf{y} = (y_1, y_2, \ldots, y_n)^T$ is an n-vector of observations (i.e., the dependent variable), $\mathbf{c} = (c_1, c_2, \ldots, c_p)^T$ is the p-vector of unknown parameters, X is the $n \times p$ matrix

$$X = \begin{bmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^{p-1} \\ 1 & x_2 & x_2^2 & \ldots & x_2^{p-1} \\ \vdots & \ddots & \vdots & & \\ 1 & x_n & x_n^2 & \ldots & x_n^{p-1} \end{bmatrix}$$

and $\epsilon = (\epsilon_1, \epsilon_2, \ldots, \epsilon_n)^T$ is the vector of residuals corresponding to the n observations.
It is straightforward to verify that the ith row of the matrix-vector product, plus the ith component of ϵ, yields the right-hand side of (L5.3).
The least squares solution of (L5.4) is the solution $\widehat{\mathbf{c}}$ of the *Normal Equations:*

$$X^T X\mathbf{c} = X^T\mathbf{y} \tag{L5.5}$$

The vector $\widehat{\mathbf{c}}$ minimizes the quantity $\| \epsilon \| = \| \mathbf{y} - X\mathbf{c} \|$, that is, minimizes the difference between the sum of squares of the differences between the "predicted" value of $\mathbf{y}$ from the "observed" value of $\mathbf{y}$, where the sum is taken over every point in the dataset.
Notice that $X^T X$ is a symmetric matrix.
The system (L5.5) can be solved as follows:

1. Define $\mathbf{z} = X^T\mathbf{y}$.
2. Define $S = X^T X$.
3. Solve $S\mathbf{c} = \mathbf{z}$.

The last step must be performed using the Cholesky decomposition of S and solving the two triangular systems.

MATLAB Notes

- Transposition is denoted by prime (i.e. a single quotation mark): X^T is denoted by `X'`.
- The triangular systems obtained from the Cholesky decomposition must be solved using the MATLAB "backslash" command. For example, to solve $U^T\mathbf{w} = \mathbf{z}$, type `w =U'\z`.
- Given the vector $\mathbf{a} = [a_1, a_2, \ldots, a_n]^T$, you can form the vector $[a_1^2, a_2^2, \ldots, a_n^2]^T$ with the expression `a.^2` (component-wise exponentiation).
- When building the matrix X you should use the special matrix `ones`.

Example 1: Least Squares fit to a Data Set by a Linear Function.

Compute the coefficients of the best linear least-squares fit to the following data.

x	2.4	3.6	3.6	4.1	4.7	5.3
y	33.8	34.7	35.5	36.0	37.5	38.1

Plot both the linear function and the data points on the same axis system.

Solution
We can solve the problem with the following MATLAB commands

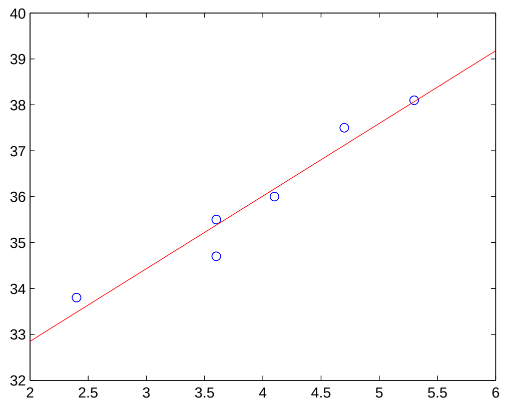
```
x = [2.4;3.6;3.6;4.1;4.7;5.3];          % build vector of x -values
y = [33.8;34.7;35.5;36.0;37.5;38.1];    % build vector of y -values
X = [ones(size(x)),x];    % build the matrix X for linear model
z = X'*y;                 % right hand side of the Normal Equations
```

```
S = X'*X;                 % Left hand side of the Normal Equations
U = chol(S);              % Cholesky decomposition
w = U'\z;        % solve the normal equations using the Cholesky decomposition
c = U\w
plot(x,y,'o')             % plot the data points
q = 2:0.1:6;              % define a vector for plotting the linear fit
fit = c(1)+c(2)*q;        % define the linear fit
hold on
plot(q,fit,'r');      % plot the linear fit together with the data points
```

Running the script file produces the output

```
c =
   29.6821
    1.5826
```

and the following figure.



EXERCISES

Instructions: For each of the following exercise, create an M-file to store the MATLAB commands. Copy and paste the M-file into a text document. Include in the text document the pictures produced by MATLAB. Resize and crop the pictures so that they do not take up too much space. If the question requires written answers, include them in the text file in the appropriate location. Make sure you clearly label and separate all the Exercises. Preview the document before printing and remove unnecessary page breaks and blank spaces.

1. Consider a set of data points $(1, y_1), (2, y_2), ...(100, y_{100})$ where the y's are random numbers with a normal distribution with mean 0 and variance 1 (note that this implies that the probability of the y-values being less than zero is the same as the probability of being greater than zero).

   (a) Based on the distribution of the points, what do you expect is the best straight-line model that can be fit to the data?

   (b) To confirm numerically your answer to part (a), generate the data as follows:

   ```
   x = [1:1:100]';          % note the prime
   y = randn(size(x));      % a column vector of 100 standard normal values.
   ```

   Plot the values as dots with the command `plot(x,y,'.')`. Follow Example 1 to find the best linear fit using MATLAB. Answer the following questions:

(i) What values do you obtain for the slope and y-intercept? Give the values in scientific notation with five digits of accuracy (use `format short e`).
Plot the linear fit together with the points (use `q = x;`)

(ii) Do the values of the slope and y-intercept confirm your answer to part (a)? Explain.

2. Download the file `co2.dat` from the web page. This file contains two columns of numbers: the year from 1959 to 2012 and the average annual carbon dioxide concentration in the atmosphere (in parts per million), as measured at the Mauna Loa observatory in Hawaii. The data come from the Earth Systems Research Laboratory (Global Monitoring Division) of the National Oceanic and Atmospheric Administration (which is the government agency charged with developing weather and climate forecast models; if you are interested in additional information and data set you can visit http://www.esrl.noaa.gov/gmd/ccgg/trends). The estimated standard error in the observations is 0.12 ppm.
Save the file in the MATLAB working directory. Then load it with

```
dat = load('co2.dat');
```

Note: if you do not save the file in your working directory, you need to include the whole path to load it into MATLAB. For instance, if you saved your file in the `C:\tmp` directory you can load it by entering `load('C:\temp\co2.dat')`.
This creates the 49×2 array `dat` whose first column is the year and whose second column is the CO_2 concentration.
You may find it convenient to create two separate data vectors, as follows:

```
year = dat(:,1);
conc= dat(:,2);
```

Plot the data with

```
plot(year,conc,'o')
```

(a) Follow Example 1 to compute the best-fit line for these data. What are the coefficients c_1 and c_2? Give the values with five digits of accuracy. Plot the line in black (use `q = year;`), together with the original data. Use `'linewidth',2` and `axis tight` in your plot.

(b) Find the best-fit *quadratic* polynomial to the CO_2 data (make sure you change appropriately the matrix X). What are the coefficients c_1, c_2 and c_3? Give the values with five digits of accuracy. Plot the fitted curve in red (use `'linewidth' ,2`) together with the data points and the linear fit from part (a). Add a legend to the graph.

3. Among the important inputs in weather-forecasting models are data sets consisting of temperature values at various parts of the atmosphere. These values are either measured directly with the use of weather balloons or inferred from remote soundings taken by weather satellites. A typical set of RAOB (weather balloon) data is given next. The temperature T in Kelvins may be considered as a function of p, the atmospheric pressure measured in decibars. Pressure in the range from 1 to 3 decibars correspond to the top of the atmosphere, and those in the range from 9 to 10 decibars correspond to the lower part of the atmosphere.

p	1	2	3	4	5	6	7	8	9	10
T	222	227	223	233	244	253	260	266	270	266

Enter the pressure values as a column vector `p` (use the colon `:` operator), and enter the temperature values as a column vector `T`.
The goal of this problem is to find the cubic polynomial $y = c_1 + c_2x + c_3x^2 + c_4x^3$ that gives the best least square fit to the data.

(a) Follow Example 1 to find the coefficients of the best cubic fit (make sure you appropriately modify the matrix X).
Plot the data together with the cubic fit. Make sure you define your vector `q` so that the graph of the cubic is nice and smooth.

(b) Let X be the matrix you used in part (a). Here is an easier way to evaluate and plot the cubic polynomial:

```
c = X\T
c = c([4:-1:1]);
q = 1:0.1:10;
z = polyval(c,q);
figure
plot(q,z,p,T,'o');
```

How do the values of `c` compare to the ones you found in part (a)?
How does the plot compare to the one you found in part (a)?

Note that the system $X\mathbf{c} = \mathbf{T}$ is overdetermined. In this case, the "\" command finds the least-squares solution to the system. In fact, for overdetermined systems, the "\" command is equivalent to solving the Normal Equations using the Cholesky decomposition, just like you did in the previous problems. Thus the vector `c = X\T` is the same one you found in part (a) by solving the normal equations.
The command `polyval(c,q)` evaluates the polynomial with coefficients given by the vector `c` in *decreasing* powers of `q`. Because the powers are decreasing we had to rearrange the entries in `c` using the command `c = c([4:-1:1]);`. Type `help polyval` to learn more.