Matlab Project 4: Least-Squares Problems

Due: Friday 1 May

Goal: Explore methods for computing least-squares solutions and fitting data

Background:

Least-squares problems have the following form: given an $m \times n$ matrix and a vector $\mathbf{b}$ in $\mathbb{R}^m$, find a vector $\hat{\mathbf{x}}$ in $\mathbb{R}^n$ which most nearly solves the linear system $A\mathbf{x} = \mathbf{b}$, in the sense of minimizing the residual $\mathbf{b} - A\mathbf{x}$:

$$\|\mathbf{b} - A\hat{\mathbf{x}}\| \leq \|\mathbf{b} - A\mathbf{x}\| \quad \text{for all } \mathbf{x} \text{ in } \mathbb{R}^n \tag{1}$$

(see section 6.5). Such problems come up in fitting functions to data (see section 6.6). According to Theorem 13 (page 363), the solutions of the least-squares problem (1) are precisely the solutions of the *normal equations*

$$A^T A\mathbf{x} = A^T\mathbf{b} \tag{2}$$

where A^T is the transpose of A. In this project we will look at different ways to use MATLAB to solve least-squares problems, and then use those methods to fit temperature data in search of evidence of global warming.

Step 1: Computing Least-Squares Lines

> *Note: The purpose of this step is to learn to compute least-squares lines using MATLAB. Do not write these answers on the answer sheet, and do not include the calculations in the* **M-file** *which you will write later.*

Suppose we want to find the linear function $y = f(t) = \beta_0 + \beta_1 t$ which best fits this data:

t	2	5	7	8
y	1	2	3	3

This data is from Example 1 on page 372 (with different notation); you can check your answers to this step by referring to that example. The corresponding linear system is $A\mathbf{x} = \mathbf{y}$, where the 4×2 matrix A has ones in its first column and the t values in its second column (see pages 370–372) and $\mathbf{x} = \begin{bmatrix} \beta_0 & \beta_1 \end{bmatrix}^T$. Here we consider three different ways solve this problem using MATLAB.

First, we can set up and solve the normal equations. For this, it is convenient to enter the data into MATLAB in vectors as follows, using the transpose (single quote) to get column vectors:

```
t = [ 2 5 7 8 ]'
y = [ 1 2 3 3 ]'
```

Then we can set up the matrix A by the command

```
A = [ ones(size(t))  t ]
```

and solve the corresponding normal equations $A^T A\mathbf{x} = A^T\mathbf{y}$ by using the backslash operator:

```
x = (A'*A)\(A'*y)
```

Verify that you get the same answer as on page 372.

Second, MATLAB has a simpler approach: for a non-square system, the backslash operator computes the least-squares solution. Here, all you need is simply

```
x = A\y
```

This should give the same result as solving the normal equations (try it!) but uses a better algorithm based on the QR factorization (see the paragraph at the bottom of page 366 and Theorem 15 on page 367); also see **help slash** in MATLAB).

Third, for the specific problem of least-squares fitting with a *polynomial*, MATLAB supplies a command **polyfit** which does the work. To read about how to use this, type **help polyfit**. Here, all you need is

```
x = polyfit(t,y,1)
```

where the last argument **1** specifies the degree of the polynomial you want. Try this and verify that it gives the same answer. *Note that the order of the polynomial coefficients is reversed from that assumed above.*

Step 2: A Specific Least-Squares Line

Now that you know how to compute least-squares lines, start writing a MATLAB script file (i.e., an **M-file**) to do the remaining calculations in this project.[1] The first thing to include in your script is code to compute the least-squares line which best fits the following data:

t	0	2	3	5	7	8	10
y	s_1	s_2	s_3	s_4	s_5	s_6	s_7

Here $s_1, s_2, \ldots, s_7$ are the *seven digits of your student number*.[2] Solve the corresponding least-squares problem using any of the three methods detailed above. To see whether your solution makes sense, plot both the original data and the computed line on the same graph. If you've stored the t and y values in vectors **t** and **y** as above, you can draw this graph as follows:

```
beta0 = x(1); beta1=x(2); % NOTE: if using polyfit, reverse the indices
LSline = beta0 + beta1*t;
plot(t,y,'-b',t,LSline,'-r') % You could replace '-b' with '*b' for points
xlabel('t'); ylabel('y')
title('least-squares line for student number data');
```

Write your student number and solution for β_0 and β_1 (rounded to four decimal places, which is the default MATLAB output) on the answer sheet. Save the plot as a PDF file (in the plot window, choose File | Save As... | PDF) and submit it (along with the other files listed below) on Moodle.

[1] If you need some reminders on how to write such a script, see Project #2.

[2] If you work with a partner, use just *one* student number—either yours or that of your partner.

Step 3: Finding Temperature Trends

Over the past fifty years, average levels of carbon dioxide in the atmosphere have been rising steadily, likely due to increased use of fossil fuels. Many scientists expect that the increasing levels of carbon dioxide will lead to global warming through the so-called "greenhouse effect". To test this hypothesis we can examine temperature data to see if a long-term trend exists. The problem is complicated by the normal variability of temperature, including variations due to location and the normal daily and seasonal cycles. Here we will explore how to use least-squares solutions to fit this data and remove the seasonal cycle.

Suppose we have data for temperatures over a period of time at a single location. This data can be expressed as a set of m points (t_i, T_i), where T_i is the observed temperature at time t_i for $i = 1, 2, \ldots, m$. We will measure time t in years and temperature T in degrees Fahrenheit. Specifically, we will use temperature data from Rocky Ford, Colorado (nearly complete monthly data from 1893–2016) obtained from the Colorado Climate Center at climatetrends.colostate.edu This data is provided in the MATLAB function **set_temps.m** which you will need to download from Moodle). In your **M-file** include the code

[time,temp] = set_temps;

to generate the vectors **time** of times $t_1, \ldots, t_m$ and **temp** of temperatures $T_1, \ldots, T_m$.

The simplest type of function to fit would be linear. However, with lots of data, we might guess that some other type of function might be better, so we might be tempted to try a quadratic. More appropriately, since we expect both a seasonal cycle and a long-term trend, we might try a function consisting of both. Therefore, we consider the following three functions (models):

(a) linear function: $T(t) = \beta_0 + \beta_1 t$

(b) quadratic function: $T(t) = \beta_0 + \beta_1 t + \beta_2 t^2$

(c) linear function plus a seasonal cycle: $T(t) = \beta_0 + \beta_1 t + \beta_2 \sin(2\pi t) + \beta_3 \cos(2\pi t)$

where β_0, β_1, β_2, and β_3 are constants to be determined. With each of these we seek a least-squares solution of a linear system of the form $A\mathbf{x} = \mathbf{b}$, where $\mathbf{b} = \begin{bmatrix} T_1 & T_2 & \ldots & T_m \end{bmatrix}^T$. For each function:

- Write A and $\mathbf{x}$ in terms of the symbols t_i, β_0, β_1, β_2, and β_3 on the answer sheet.
- Use one of the methods from step 1 to find the least-squares solution. Note that **polyfit** will work for (a) and (b) but not for (c) [why not?]. If the method you choose is ill-conditioned, either try another or fix the underlying problem by shifting the times.[3]
- *Write the computed values of β on the answer sheet—round to four decimal places (except where too small—then use scientific notation).*
- Use the function to predict the temperature in the year 2100, i.e., $T(2100)$. In each case use the full values computed by MATLAB—do not use values copied from the output, which are probably rounded and will give wrong results. For function (c), use only the linear part (the β_0 and β_1 terms) so the seasonal cycle will be omitted. *Write the resulting temperature $T(2100)$ on the answer sheet, rounded to two decimal places.*

[3] Use **t0 = time(1); time = time-t0** to start at 0 rather than 1893. If you do this, you'll need to shift back by **t0** to draw the graph, to compute $T(2100)$, and to use only the data from 2000 on (see the next page).

Finally, to get a hint of some of the complexities of fitting data and extrapolating trends, redo the above calculations using only the data from the year 2000 to the present. An easy way to do this is to reset the variables **time** and **temp** as follows:

```
indx = (time>=2000);      % finds indices of all times from 2000 on
time = time(indx);
temp = temp(indx);
```

Write the computed values of β and predicted temperatures $T(2100)$ on the answer sheet as before. Circle the value of $T(2100)$ which you trust most (of the six values you computed), and briefly explain why. Also—for the linear function only—plot the temperature data and the least-squares line on the same graph; save this graph as a PDF file and submit it (along with your other files) on Moodle.

Completing your M-file:

By the time you're done with the above calculations, you should have written a MATLAB script file (aka **M-file**) to do all of the calculations. Complete this for submission by doing the following:

- At the top of the file put in comment lines (lines starting with **%**) identifying the project, *your* name, and the name of your *partner* (if you worked with one) in the following format:

  ```
  % MA339 Project: Least-Squares Problems
  % Anthony Fauci ⟵ replace with your name
  % Deborah Birx ⟵ replace with name of your partner (if any)
  ```

- Remove any commands *other than* those needed to do the above calculations (for example, anything you tried which didn't work).
- Add a few comment lines to identify the main sections of the script.
- Save your script as an **M-file** with an appropriate name (for example, **project4.m**). If you submit it in the right place on Moodle (and include your name–and that of your partner, if any–in the comments) then it will be identified as yours for grading.
- Verify that it actually works by clearing or restarting Matlab (type *clear* or *exit*) and running it again "cold". This will help catch cases where you've defined something outside the script which the script needs. *Your script should run without input and produce all results needed.*

Submit the following four files on Moodle:

1. The completed answer sheet (as a PDF file). [one way to do this: use camscanner]
2. Your plot from Step 2 (as a PDF file).
3. Your plot from Step 3 (as a PDF file).
4. Your MATLAB script (as an **M-file**).

Note: You may work alone or with *one* other person. If you work with someone else, submit only one copy of your work (with both names on top) and upload only one script (with both names in its comments). *No groups bigger than two. No collaboration between groups.*

Matlab Project: Least-Squares Problems **Name:**_____

ANSWER SHEET **Name:**_____

Step 2: Least-square line for student number data

Student number: $\beta_0 =$ $\beta_1 =$

Step 3: Matrices and vectors *in terms of the symbols* t_i, β_0, β_1, β_2, *and* β_3 (the first is done for you)

Linear: $A = \begin{bmatrix} 1 & t_i \end{bmatrix}$ $\mathbf{x} = \begin{bmatrix} \beta_0 & \beta_1 \end{bmatrix}^T$

Quadratic: $A =$ $\mathbf{x} =$

Linear+cycle: $A =$ $\mathbf{x} =$

Solutions (full data set):

Function	β_0	β_1	β_2	β_3	$T(2100)$
Linear			—	—	
Quadratic				—	
Linear+cycle					

Solutions (only using data from 2000 on):

Function	β_0	β_1	β_2	β_3	$T(2100)$
Linear			—	—	
Quadratic				—	
Linear+cycle					

Circle the value of $T(2100)$ which you trust most, and briefly explain why here:

Upload to Moodle: Scan and upload this sheet to Moodle along with your **M-file** and two plots.