AMath 584 / Math 584
Homework #3
Due 11:00pm PDT
Tuesday, November 8, 2016

Name: Your Name Here

Netid: Your NetID Here

Problem 1. Let

$$A = \begin{bmatrix} 0 & 3 \\ 2 & 0 \\ 0 & 4 \end{bmatrix}.$$

Do this problem "by hand", and write out the matrices that you get at each stage.

(a) Use the classical Gram-Schmidt algorithm to determine a reduced QR factorization of $A = \hat{Q}\hat{R}$.

(b) From this determine the full QR factorization with Q a unitary 3×3 matrix.

(c) Determine a full QR factorization using the Householder triangularization algorithm. In particular:

- Determine the Householder reflector $F_1 = Q_1$ that reduces A to the form

$$Q_1 A = \begin{bmatrix} X & X \\ 0 & X \\ 0 & X \end{bmatrix}$$

  where the X's show elements that are possibly nonzero.

- Determine the 2×2 Householder reflector F_2 so that the 3×3 matrix

$$Q_2 = \begin{bmatrix} 1 & 0^T \\ 0 & F_2 \end{bmatrix} \qquad (\text{where } 0 \in \mathbb{R}^2)$$

  reduces $Q_1 A$ to the desired upper triangular form $Q_2 Q_1 A = R \in \mathbb{R}^{3\times 2}$.

- Determine Q so that $A = QR$ from Q_1 and Q_2.

This is following the notation used in lecture on Friday October 21.

Note: Recall that the choice of v in (10.5) in the book is for numerical stability and mathematically the first term could be given a plus or minus sign and the algorithm would work. In the case when $x_1 = 0$, define $\text{sign}(0) = 1$.

Comment on whether these two approaches gave the same matrices Q and R and if not what the difference is.

Problem 2.

(a) Using the results of Problem 1(a) above, determine the (left) pseudoinverse A^+ for the matrix from Problem 1.

(b) Compute A^+b and A^+c for the two vectors

$$b = \begin{bmatrix} 3 \\ 2 \\ 4 \end{bmatrix}, \qquad c = \begin{bmatrix} -16 \\ 0 \\ 12 \end{bmatrix}.$$

Explain why A^+c gives the zero vector.

Problem 3. Let $q \in \mathbb{C}^m$ have $\|q\|_2 = 1$. Then $P = qq^*$ is a projection matrix.

(a) The matrix P has a singular value decomposition with $U = [q|Q_\perp]$ for some appropriate matrix $Q_\perp$. What are the singular values of P?

(b) Find an SVD of the projection matrix $I - P = I - qq^*$. In particular, what are the singluar values? **Hint:** Write $I = UU^*$ where U is as above and use the SVD of qq^*.

(c) Find an SVD of the reflector $I - 2P$. What are the singular values?

Modified: *Recall that the singular values should be non-negative.*

(d) **Added:** *Suppose you write a general vector x as a linear combination of the columns of U, i.e., as $x = c_1u_1 + c_2u_2 + \cdots + c_mu_m$ where $c_j = u_j^*x$. In terms of a similar representation using the columns of U as a basis, what are the vectors Px, $(I - P)x$, and $(I - 2P)x$. Think about the geometric meaning of these.*

Problem 4. Consider the polynomial of Exercise 13.3 in the book, for which

```
c = [-512, 2304, -4608, 5376, -4032, 2016, -672, 144, -18, 1]
```

(a) Write a function in Matlab or Python `psum(x,c)` that evaluates a polynomial defined by coefficients `c` by the natural summation algorithm, e.g. in pseudo-code:

```
psum = c[0]
for j = 1 to n-1
    psum = psum + c[j]*t**j
```

for a polynomial of degree $n - 1$. Test your function to make sure it works on some simple examples.

(b) Put in a print statement to print out the partial sum `psum` each time through the loop. Now test it on the polynomial $p(x)$ from Exer. 13.3 at $x = 2.1$. How large is the error in the computed value? Comment on how this relates to the size of the partial sums. **Hint:** The true value is easy to compute from the fact that $p(x) = (x - 2)^9$.

(c) Do the same thing using Horner's method to evaluate the polynomial. Use the Python function given in the notebook `notebooks/LeastSquares2.ipynb` or write a Matlab version.

(d) Plot the polynomial near $x = 2$ by evaluating at the points `linspace(1.95, 2.05, 101)` using both evaluation methods and plot these curves together. (You will want to take out the print statements for this.)

Also plot the polynomial when it is evaluated using the expression $(x-2)^9$. This should be the most accurate and give a smooth curve that approximates the true polynomial.

Problem 5. Exercise 14.1 in the book. Briefly explain your answers.

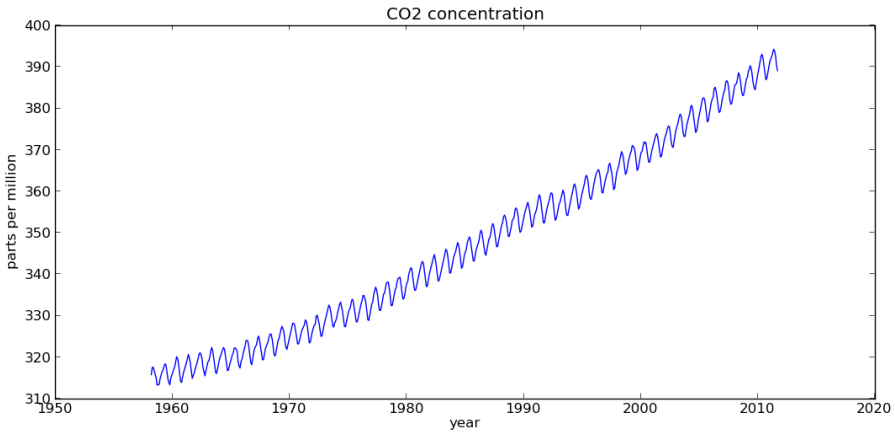
For (d), look up Stirling's approximation for the factorial.

Problem 6.

The file `homeworks/hw3/co2.txt` in the class repository contains a set of data that you will need for this assignment.

This file contains 90 lines of comments followed by 636 lines of data. The third column is time (in years) and the fourth column is CO_2 concentration in the atmosphere (in parts per million) as measured at Mauna Loa. These give a set of $(t_i,\ y_i)$ values that we will apply least squares to. See `http://www.esrl.noaa.gov/gmd/ccgg/trends/mlo.html` for more information on this data. (Note that the file you are using is from 2011, not the most recent version.)

Plotting this data should give:



To produce such a plot (and to do the rest of this assignment) you will have to read the data into Matlab or Python and first pre-process it by stripping out any (t, y) values for which $y = -99.99$ (indicating that data was not available at this time).

The codes at the end of the assignment show how to do this, and print out the figure as a `png` file that can be incorporated in your latex as illustrated in this file.

There is a clear oscillation in this data with a period of one year, due to seasonal variations. There is also an upward trend that might be fit with a linear function as the first attempt. This suggests we perform a least squares fit with the basis functions

$$\phi_0(t) = \cos(2\pi t), \quad \phi_1(t) = \sin(2\pi t), \quad \phi_2(t) = 1, \quad \phi_3(t) = t - 1985.$$

The shift by 1985 in the last basis function keeps the values in the matrix smaller, and becomes particularly important when we add higher powers.

(a) Use Matlab or Python to compute the least squares fit.

Added to assignment: *Do this by computing the QR factorization (using the built-in* `qr` *function) and then solving* $Rc = Q^T y$. *In Matlab you can solve this nonsingular linear system using the backslash operator,* `c = R\Q'*y`.

(Note that in Matlab you can also solve a least square problem directly with the backslash operator, `c = A\b`, *and in Python the function* `numpy.linalg.lstsq(A,b)` *can be used.*

Plot the fit as a red curve on the same plot.

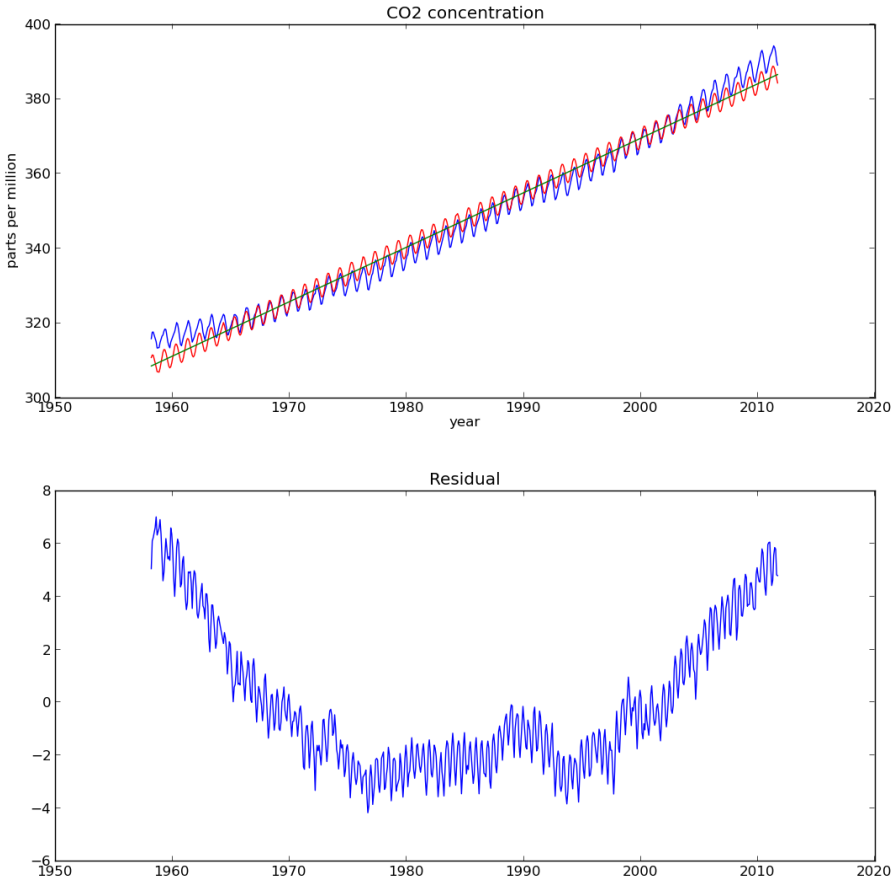
To see the trend without the oscillation, take the coefficients $c \in \mathbb{R}^4$ just found and plot the function

$$p(t) = c_2\phi_2(t) + c_3\phi_3(t)$$

on the same plot, as a green curve.

Also plot the residual as a function of time on a separate plot.

You should get something like these:



(b) This fit doesn't match the trend very well, so we might try improving it by using a quadratic function instead of linear for the "secular terms", especially since the residual looks something like a parabola. This can be done by adding another basis function $\phi_4(t) = (t-1985)^2$. Repeat part (a) using this. Plot the fit, the quadratic trend, and the residual.

(c) This fit looks good enough that we might be tempted to use it to predict the future. Evaluate the function $f(t)$ over the interval $1960 \le t \le 2050$ with enough points to show how it behaves and plot this.

As a sanity check we might also try extrapolating backwards in time. Plot the same function $f(t)$ over the interval $1800 \le t \le 2010$ and comment on what you see.

Matlab:

```
% Read column data with whitespace delimiter, skipping 90 rows:
d = importdata('co2.txt',' ',90)
data = d.data;
t = data(:,3);
y = data(:,4);

% filter out the bad data:
good = y>0;   % indices of good data
t = t(good);  % this portion of array
y = y(good);

plot(t,y)
title('CO2 concentration')
xlabel('year')
ylabel('parts per million')
print -dpng co2figure.png
```

Python:

```
from pylab import *

# Read column data with whitespace delimiter, skipping 90 rows:
data = loadtxt('co2.txt',skiprows=90)
t = data[:,2]   # First column is index 0
y = data[:,3]

# filter out the bad data:
good = y>0   # indices of good data
t = t[good]  # this portion of array
y = y[good]

figure(1)
clf()
plot(t,y)
title('CO2 concentration')
xlabel('year')
ylabel('parts per million')
savefig('co2figure.png')
```