Table of Contents

House Keeping

```
close all; clear all; clc;

%{
    CSCI 3656 HW9 main script
    Author: Connor O'Reilly
    email: coor1752 @colorado.edu
    Last Edited: 11/7/2021
%}
```

Part 1:

```
%define function
f = @(x) sin(2 * pi * x) + cos(3 *pi * x);

%define interval
n = 33; %number of data points
d = 7; %degree of least squares polynomial
x_train = linspace(-1 , 1 , n);
y_train = f(x_train);

%use poly val for degree d polynomial approximation
%used poly val examples from
%https://www.mathworks.com/help/matlab/ref/polyval.html#d123e1107356
 as
%reference

[p_part1, ~] = polyfit(x_train, y_train, d); %Used polyfit to fit a
 seventh degree polynomial to the data.
                                                % p: function
 values,
                                                % S: standard
 error for prediction

p_eval_part1 = polyval(p_part1 , x_train);      %evaluates model at
 all training points for plotting

%plotting provided in plotting section of code
```

Part 2 (Create testing data):

```
%generatee 100 random points between on the interval [-1,..,1]

%using matlab documentation on rng and rand, in addition to the
 homework 9
%.m files as reference

rng('shuffle','twister');
x_test = -1 + (1 + 1)*rand(100,1);
x_test = sort(x_test); %order array for plotting

y_test = f(x_test);

% for degree = 1 -> 31 compute least squares coefficients of
 polynomial of
% degree d with training data as last problem. imported functions
 below

%obtain size of x data for vandermonde
size_train = size(x_train);
%initialize arrays
err_poly = zeros(31,1);
err_chol = err_poly;
err_QR = err_poly;
cond_ata = err_poly;
con_vander = err_poly;
for d = 1:31
    [p_part2, ~] = polyfit(x_train, y_train, d); %Used polyfit to fit
 a seventh degree polynomial to the data.
                                                  % p: function
 values,
                                                  % S: standard
 error for prediction

    p_eval_part2 = polyval(p_part2 , x_test);      %evaluates model at
 all training points for plotting

   %obtain vandermonde matrix
   %initialize
   A_vander = zeros(size_train(2) , d+1);
  %i think this is inefficient but im tired lol
   for i = 1 : size_train(2)
       A_vander(i,:) = x_train(i);
   end
   %raise elements to power (d+1 columns)
   for i = 0 : d
       A_vander( : , i + 1) = A_vander(: , i + 1).^i;
   end

   %now find coefficients using fucntions from hw 8
```

```
    [c_chol, cond_ata(d)] =
 method_NormEq_Cholesky(A_vander,y_train'); %adjust to vertical for
 function
    c_QR = method_ThinQR(A_vander, y_train', 0); %adjust to vertical
 for function

    %only evaluate cholesky if matrix was sym pos def matrix is not
 nan
    if (sum(isnan(c_chol(:))) < 1)
        %flip for poly val function
        c_chol = flip(c_chol);

        %evalutate polynomials for c_chol and c_QR
        p_chol = polyval(c_chol, x_test);

        %compute error
        err_chol(d) = norm(y_test - p_chol) / norm(y_test);

    else
        %if error doesnt make sense
        %for plotting and discussion sake change value to 10^0;
        err_chol(d) = 1;

    end
    %obtain condition number for vandermonde
    con_vander(d) = cond(A_vander);
    %flip for poly val function
    c_QR = flip(c_QR);

    %evaluate polynomial for c_qr
    p_QR = polyval(c_QR, x_test);

    %compute error
    err_poly(d) = norm(y_test - p_eval_part2) / norm(y_test);
    err_QR(d) = norm(y_test - p_QR) / norm(y_test);
end

%error plotting in plot section
```

Extra credit

```
x_extra = [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 17];
y_extra = [89 105 125 159 227 331 444 564 728 1000 1267 1645 2204 2826
 3485 7038];
y_extra_log = log10(y_extra);

%for non log

%need more testing points ig
rng('shuffle','twister');
x_test_extra = 1 + (17 - 1)*rand(30,1);
x_test_extra = sort(x_test_extra); %order array for plotting
```

```
%obtain vandermonde matrix, no need to flip
vand_extra = vander(x_extra);

%get least squares coefficients, flip y for dimensions
c_extra = vand_extra\y_extra';

%evaluate
p_extra = polyval(c_extra, x_test_extra);
ex = find(p_extra<0);
p_extra(p_extra < 0) = [];
x_test_extra(ex) = [];
%for log
c_extra = vand_extra\y_extra_log';
p_extra_log = polyval(c_extra, x_test_extra);

growthrate = (p_extra(14) - p_extra(11) )/ p_extra(11) ;
doubletime = (x_test_extra(14)-x_test_extra(11)) * (log(2) /
 log(1+growthrate));
```

```
Warning: Matrix is close to singular or badly scaled. Results may be
inaccurate. RCOND =  1.548594e-24.
Warning: Matrix is close to singular or badly scaled. Results may be
inaccurate. RCOND =  1.548594e-24.
```

Interpretation

```
fprintf('\n-----------------------------------------------------------------------
\n')
fprintf('Analysis:')
fprintf('\n-----------------------------------------------------------------------
\n\n')
fprintf(['Initailly looking at the graph, when d = 19 the error for
 the method using normal equations with cholesky factorization is not
 included and set to 1. \n' , ...
'this is due to the following operation A_vander''*A resulting in
 an ill-conidtioned, non-symettric positive definite matrix which
 introduces round-off error causing cholesky to fail.\n ' , ...
'therefore it does not make sense to include this error. The condition
 number for the vandermonde matrix was recorded for d = 1, ... , 31
 and as expected around d = 20, cholesky failed. \n ', ...
'what was not expected was it seems the error of both methods QR
 and NE decreased as the condition number of the vandermonde system
 increased. \n',...
'I dont really understand why this is  but might be due to the
 fact the degree of the polynomial remains well below the number of
 training data points being used\n\n'])

fprintf(['After adjusting the number of training data points to be
 less than the maximum degree d used to compute the least-squares
 coefficients ( n << d ) , the error significantly increased whenever
 a polynomial of degree d was fit to a smaller amount of training
 data.\n ', ...
```

```
'i re-ran the code with 10 evenly spaced points on the interval [-1,1]
 for the training data. \nas the polynomial degree increased past 10
 the normalized testing error increased as the condition number of the
 vandermonde system increased. \n ',...
'so to sum up this long probably wrong analysis, my interpretation is
 the condition number of the matrix in the least-squares problem plays
 a minor role in the overal accuracy of each method. \n',...
'As long as the degree of polynomial remains less than or equal to the
 number of training points, the accuracy of the QR and Normal equation
 method will increase as the polynomial degree increases.']);
fprintf('\n\nsorry about this long analysis')

fprintf('\n------------------------------------------------------------------
\n')
fprintf('Extra Credit:')
fprintf('\n------------------------------------------------------------------
\n\n')
fprintf('My log linear model does seems to model the growth accuratly
 except for the initial first three and last three data points,
 doubling time from least squares fit around %0.4f days. \ni got tired
 and the extra credit is sketchy',doubletime)


----------------------------------------------------------------
Analysis:
----------------------------------------------------------------

Initailly looking at the graph, when d = 19 the error for the method
 using normal equations with cholesky factorization is not included
 and set to 1.
this is due to the following operation A_vander'*A resulting in
 an ill-conidtioned, non-symettric positive definite matrix which
 introduces round-off error causing cholesky to fail.
 therefore it does not make sense to include this error. The condition
 number for the vandermonde matrix was recorded for d = 1, ... , 31
 and as expected around d = 20, cholesky failed.
 what was not expected was it seems the error of both methods QR
 and NE decreased as the condition number of the vandermonde system
 increased.
I dont really understand why this is  but might be due to the fact the
 degree of the polynomial remains well below the number of training
 data points being used

After adjusting the number of training data points to be less than the
 maximum degree d used to compute the least-squares coefficients ( n
 << d ) , the error significantly increased whenever a polynomial of
 degree d was fit to a smaller amount of training data.
 i re-ran the code with 10 evenly spaced points on the interval [-1,1]
 for the training data.
as the polynomial degree increased past 10 the normalized testing
 error increased as the condition number of the vandermonde system
 increased.
```

so to sum up this long probably wrong analysis, my interpretation is the condition number of the matrix in the least-squares problem plays a minor role in the overal accuracy of each method.
As long as the degree of polynomial remains less than or equal to the number of training points, the accuracy of the QR and Normal equation method will increase as the polynomial degree increases.

sorry about this long analysis

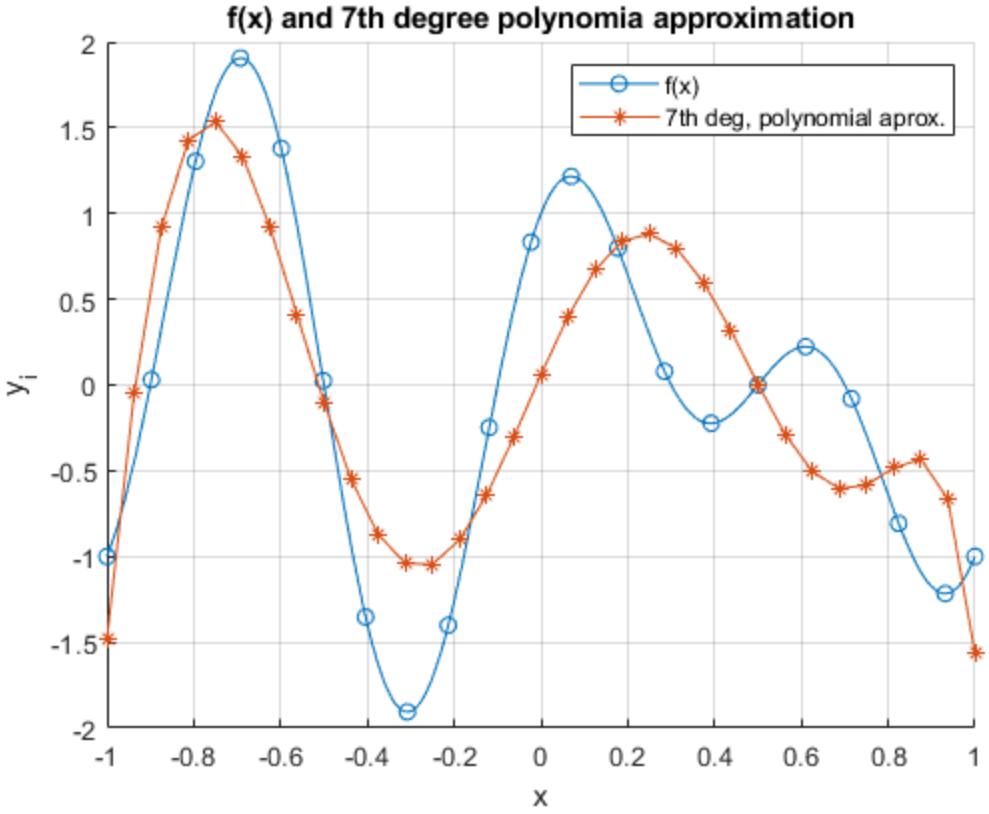
Extra Credit:

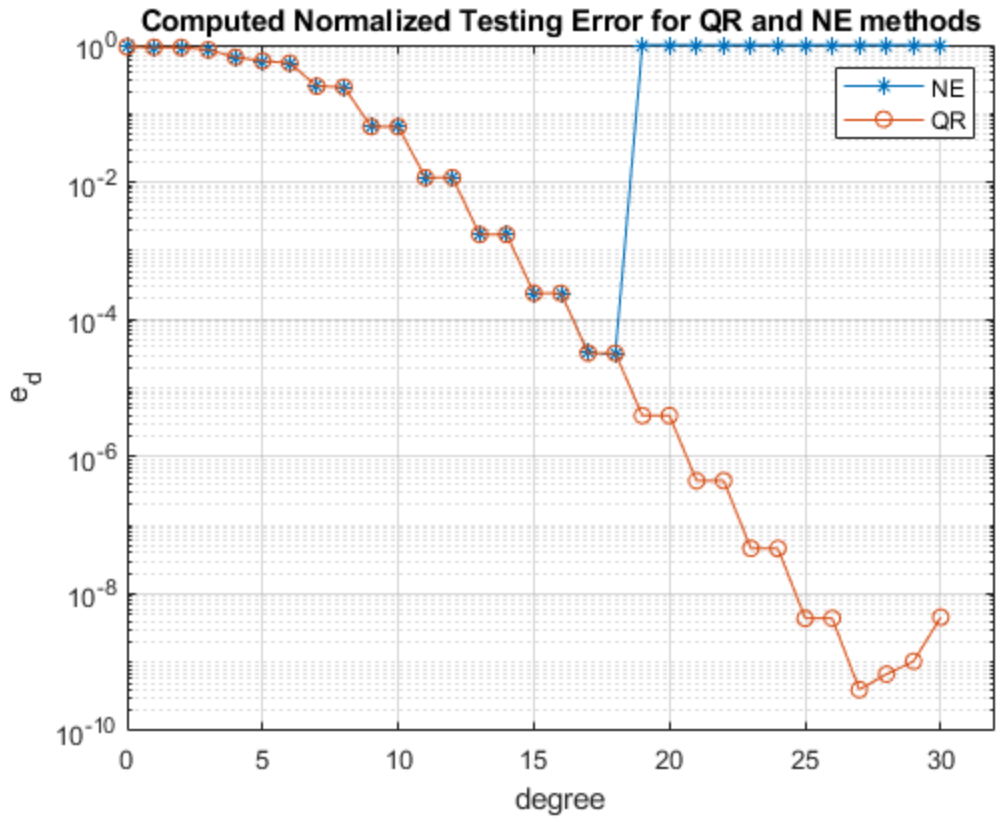
My log linear model does seems to model the growth accuratly except for the initial first three and last three data points, doubling time from least squares fit around 2.6238 days.
i got tired and the extra credit is sketchy

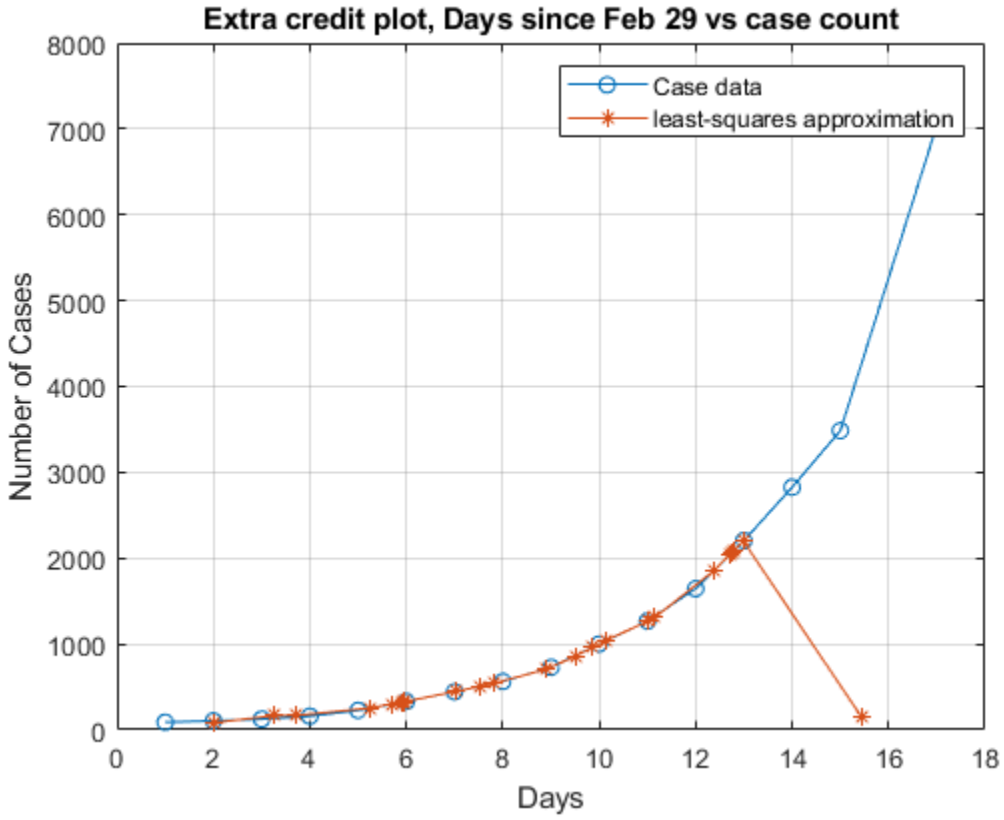
Plotting

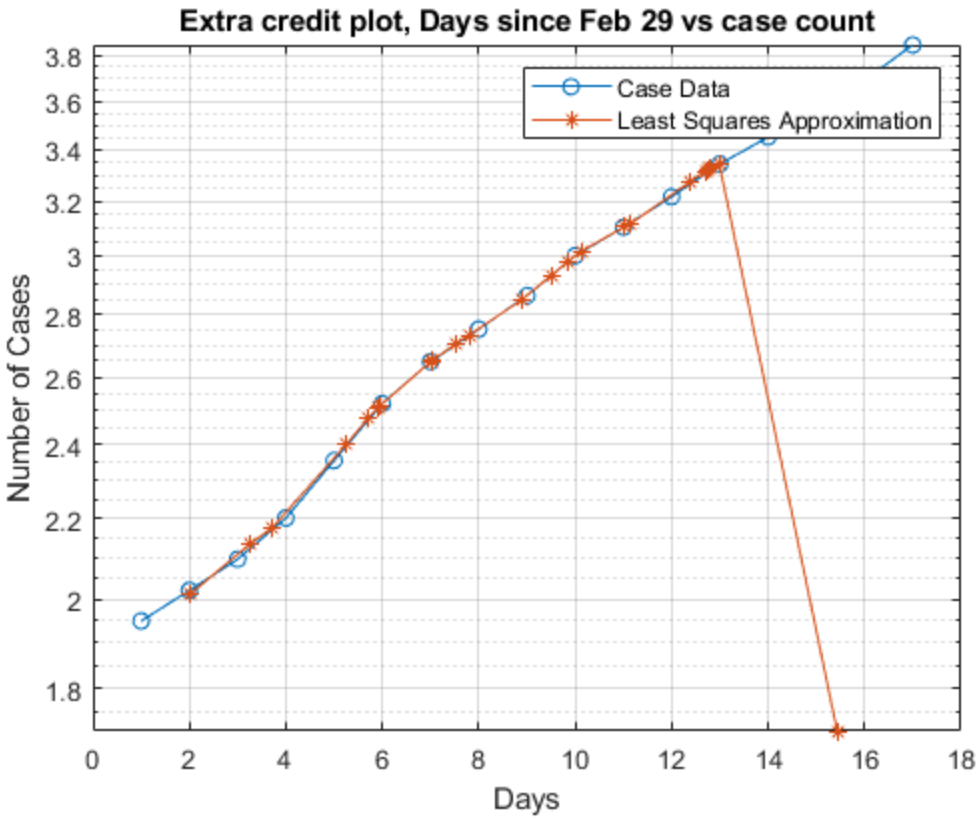
```
% seventh degree polynomial approximation
figure(1)
hold on;
fplot(f,[x_train(1) , x_train(end)], '-o') %plot function
plot(x_train, p_eval_part1,'-*');
grid on;
title('f(x) and 7th degree polynomia approximation')
xlabel('x')
ylabel('y_i')
legend('f(x)' , '7th deg, polynomial aprox.')
hold off;
 snapnow
%error for QR vs NE
figure(2)
semilogy(0:30, err_chol,  '-*' , 0:30, err_QR, '-o')
hold on;
grid on;
ylabel('e_d')
xlabel('degree')
xlim([0 32])
title('Computed Normalized Testing Error for QR and NE methods')
legend('NE','QR')
hold off;
 snapnow
%extra credit plot
%non log
figure(3)
plot(x_extra, y_extra, '-o')
hold on;
plot(x_test_extra, p_extra, '-*')
grid on;
title('Extra credit plot, Days since Feb 29 vs case count')
legend('Case data', 'least-squares approximation')
xlabel('Days')
```

```
ylabel('Number of Cases')
hold off;
 snapnow
%log
figure(4)
semilogy(x_extra, y_extra_log, '-o', x_test_extra, p_extra_log, '-*')
hold on;
grid on;
xlabel('Days')
ylabel('Number of Cases')
title('Extra credit plot, Days since Feb 29 vs case count')
legend('Case Data', 'Least Squares Approximation')
hold off;
 snapnow
%figure
```




Computed Normalized Testing Error for QR and NE methods
NE
QR
e_d
degree


Extra credit plot, Days since Feb 29 vs case count
Case data
least-squares approximation
Number of Cases
Days



Functions

```
function [x, con_vand] = method_NormEq_Cholesky(A,b)
%{
    Prupose: Matlab implementation of method to solve least squares
 problem using normal
    equations with cholesky
    Inputs:
        A: LHS matrix for Ax = b
        b: RHS matrix for Ax = b
    output:
        x: variable matrix x in Ax = b  (x) which minimizes
        ||b-Ax||_2^2
%}

%form A^T*A e R^n*n and A^T*b e R^n
    ata = A'*A;
    atb = A'*b;

%Solve normal eqns, (A^T*A)x = A^T*b for x
%first lets double check
%get size to make sure A = n x n
    size_A = size(ata);
%check rank
    r_ata = rank(ata);
```

```
%updated for hw9
    con_vand = cond(ata);
    if( (size_A(1) == size_A(2) ) && ( r_ata == size_A(1) )  )
        %matrix is sym, pos def, compute and solve cholesky
 factorization (
        %determine variable vector using cholesky factorization
        R = chol(ata);
        x = R\(R'\atb);



    else
        %cant use error func for this homework so itll break, if using
        %outside of hw uncomment
        %error('Matrix is not Symmetric Positive Definite')
        %warning does not apply to this hw so it will be adjusted
        %if this error occurs create nan array maybe later could be
 used as a flag in
        %error calc
        siz_atb = size(atb);
        warning('Error not accounted for due to d_%i vandermonde not
 sym pos def', siz_atb(1))
        x = NaN(siz_atb);
    end

end


function [ x ] = method_ThinQR(A, b, bool)
%{
    Purpose: Matlab implementation to solve the linear least-squares
    problem using a method based on Thin QR factorization
    Input:
        A: LHS matrix, Ax = b
        b: RHS matrix, Ax = b
        bool: boolean value, 1 if you want to use my janky code which
 seems
        wrong, or built in qr function that works
    Output:
        x: variable matrix x in Ax = b  (x) which minimizes
        ||b-Ax||_2^2
%}

%alright so because i am trying to do both the homeowkrs this weekend
 im
%assuming the grant-schmidt orthogonalization and least squares is
 similar
%to that of the thin QR decomp in lecture. Sauer's numerical analysis
 and
%https://ocw.mit.edu/courses/mathematics/18-06-linear-algebra-
spring-2010/related-resources/MIT18_06S10_gramschmidtmat.pdf
%were used as a reference
```

```
%compute the thin QR decomposition


    if bool
        %obtain size of A
            size_A = size(A);

        %initialize r matrix. upper right traingular n x n
            r = zeros(size_A(2),size_A(2));
        %initialize q tall matrixx orthonormal columns, m x n
            q = zeros( size_A );

        %Grant-Schmidt orthogonalization
            for j = 1 : size_A(2)

                y = A(:,j);

                for i = 1 : j-1

                    r(i,j) = q(: , i)' * A(: , j);
                    y = y - r(i,j) * q(: , i);

                end
                r(j , j )= norm(y);
                q(:,j) = y/r(j , j);

            end
    else

        % if this doesnt work just use qr matlab
        [q,r] = qr(A, 0);
    end
    % After thin QR is computed, finish computation
    b_til = q'*b;

    % Solve
    x = r\b_til;

end
```

```
Warning: Error not accounted for due to d_21 vandermonde not sym pos
 def
Warning: Error not accounted for due to d_22 vandermonde not sym pos
 def
Warning: Error not accounted for due to d_23 vandermonde not sym pos
 def
Warning: Error not accounted for due to d_24 vandermonde not sym pos
 def
Warning: Error not accounted for due to d_25 vandermonde not sym pos
 def
Warning: Error not accounted for due to d_26 vandermonde not sym pos
 def
Warning: Error not accounted for due to d_27 vandermonde not sym pos
 def
```

```
Warning: Error not accounted for due to d_28 vandermonde not sym pos
 def
Warning: Error not accounted for due to d_29 vandermonde not sym pos
 def
Warning: Error not accounted for due to d_30 vandermonde not sym pos
 def
Warning: Error not accounted for due to d_31 vandermonde not sym pos
 def
Warning: Error not accounted for due to d_32 vandermonde not sym pos
 def
```

Published with MATLAB® R2021a