EECE 540 **Spring 2018**

Project 2: Matrix Applications

****** Lab Intro ********

1. Data Reading

a) As part of this assignment two files are posted on the webpage. The first "Data01.csv" is a text file and the second "Data02.mat" is a binary MATLAB file. Download these two files and read them into MATLAB, being sure to document the time needed to read in each file.

b) The two files were written from MATLAB, using the same data and thus should contain the same numbers. Compute the Maximum Absolute Relative Error (MaxARE) between the data from the two files, assuming that the binary (Data02) data is more accurate.

Based on the time required to read in the text file and its size as compared to the binary file, why would anyone use a text file?

2. Curve Fitting:

a) There is a file provided as part of this assignment called "Data02.mat". It is a matlab file that contains the data to be analyzed in this section. The third row will be assumed to be the dependent variable and first two rows are to be treated as the independent variables. In this case we want to extend the matching to include the independent variables and all their products. The form of the A matrix used in this variation is shown at the end of this assignment. As part of this analysis you will need to compute the parameters for the equation below and compute the CD for its fit.

$$z_i = a*x_i + b*x_i^2 + c*x_i*y_i + d*y_i + e*y_i^2 + g$$

b) At this point a common approach would be to compute the fitting equation and CD for all the possible combinations of independent variables and cross products, however this would require 5! or 120 different cases. As a simplification, a set of 5 additional fits will be done, where one of the terms is replaced with a set of random numbers, and the fit and CD are computed for these five cases. The form of the matrices is shown in some detail in Appendix PII-1.

c) Based on the weights, and CD computed from the 6 cases of regression analysis performed, what can be said about the data?

What data or terms appear to be related to the dependent variable and which are not related?

d) Pull out what you think are the only important terms and repeat the match using only those terms. Compute the CD for this reduced match.

How does the reduced match compare with the match obtained using all the terms?

EECE 540 **Spring 2018**

Project 2: Matrix Applications

3. Eigenvalues and EigenVectors: (15)

The data used in this section, hereafter called X, is created by multiplying the matrix read in times its transpose (Data02 * Data02'). This should result in a 3 x 3 matrix, and should have some important properties.

a) First show that it is positive semi-definite or in other words, $\underline{x}' * X * \underline{x} > 0$ for all $\underline{x} \neq \underline{0}$. This should be done by hand.

b) Create a matlab function that employs the Power Method (PM) to find the largest eigenvalue (evalue) and eigenvector (evector) of a matrix X. The prototype of the function should look like the following.

```
function [evalue, evector] = PM_eigen( X );
```

Be sure to check that the matrix is square and to limit the number of iterations to some large count. Use some method to indicate to the calling function that no solution was found (evalue = NaN)? **A comment block describing the function is important**, be sure to have one in your function. It should be complete with inputs and outputs described.

c) Once we have an eigenvalue and vector, we will want to remove that eigenvalue - vector set from the matrix. This removal (or deflation) can be done by the following equation

Xreduced = X - l * v * v' where l is the eigenvalue and v

is the eigenvector.

Create a function that will remove an eigenvalue – vector set from a matrix returning a matrix.

```
function [Xreduced] = Eigen_Deflate( X, eigenValue, eigenVector );
```

A comment block describing the function is important, be sure to have on in your function.

d) Employ these two functions to search for all the eigenvalue – vectors in the matrix X. `function`

```
[vectors,values] = EigenVV( X );
```

Note the eigenvectors will be the columns of the return-value vectors, and the eigenvalues should be placed in the diagonal elements of return-value values, an N x N matrix, all other elements are zero.

A **comment block describing the function is important**, be sure to have one in your function. It should be complete with inputs and outputs described.

B

Project 2: Matrix Applications

e) Does it find all the eigenvalues? Test the results from EigenVV against all the values found by the matlab function eig(X).

f) Compute the condition of the matrix X, using the ratio of the largest and smallest eigenvalues and compare that to the condition found using cond(X).

EECE 540 **Spring 2018**

Project 2: Matrix Applications

**********MY LAB REPORT**********

Section 1. Data Reading

The first section calls for a test of the reading of data in both text and binary formats. A script that times the reading of both types of files is included in Appendix A. The output of the script is included here.

```
>> ReadTest

It took 1.75634 seconds to read the CSV file.
and it took 0.0339314 seconds to read the MAT file. The
maximum absolute relative eror was 5.68243e-16
```

In the print out we can see that text files require a much longer time to read than a binary file. However if we were to try and look at the data with an editor, it is much easier to interpret the data in the text file and almost impossible to interpret the binary data.

So, why would anyone use a text file? You would use a text file if the data is to be read by a person, or if you want to import that data into another program or system. It is important to note that text data will not match the data in the computer perfectly but will be epsilon close.

Section 2. Curve Fitting

In this section, an experiment is set up to demonstrate the use of CD and least-square curve fitting. A data set in the Data02.mat file is used to test a generalized curve fit.

A script that sets up the analysis requested in Parts a and b can be found in Appendix B. It should be noted that the script writes the CD's and parameters into a csv file. That file was imported into excel, where it was annotated. An image of the spreadsheet in included here.

	Drop x	Drop x^2	Drop x*y	Drop y	Drop y^2	Full Model	Reduced	Only y^2
CD = >	0.94758	0.97565	0.97565	0.97565	0.94474	0.97565	0.97565	0.41775
a =>	0.00976	5.9762	6.00401	5.9934	5.96758	5.98971	5.99046	0
b =>	-0.1893	-0.01458	0.00068	0.00067	0.00018	0.00067	0	0
c =>	-0.19053	-0.00143	0.01051	-0.00106	-0.00274	-0.00143	0	0
d =>	-1.91489	-0.01165	0.00269	-0.00565	-5.0269	-0.01164	0	0
e =>	0.24952	0.25035	0.25036	0.25072	-0.03451	0.25035	0.25023	0.24977
g =>	-37.0665	0.75346	0.95312	0.88372	-20.3302	0.81018	0.88373	-58.9933

Figure 2-1. Table of CD's and Parameters

Part c calls for an analysis of the CD's and parameters computed in parts a and b. From the table above we can see that the CD's are consistent except if x or y * y is dropped. These cases are marked in red.

Based on this we can say that these term appear to be the only parameters that are important. Note it is possible that the data set accidentally has this trend, so what can definitively say is that x^2, x*y and y are not related to the dependent variable.

The final step was to produce a model using only these terms. The results of which are shown in the next to last column of the spreadsheet. In this column we can see that the CD is almost the same as that for the full set of terms. Hence these terms are basically all that is needed to model the dependent variable.

As an extra test, a match was done only using y^2, which is the term that had the largest effect on the CD. In the last column the results of this test can be seen. It shows that the CD of this case is not that close to full match, showing that x*y contributes to the fitted curve.

3. Eigenvalues and EigenVector

The first thing called for in this section was to show that the matrix we were using is semi-positive definite. This proof is started by writing the definition of semi-positive definite, substituting in the definition of X.

Parts b, c and d called for functions to be written. These can be found in Appendix C.

Also in part d was to test the functions on the matrix. The results of this test can be found in the print out included below. In this output we can see that the eigenvalues and vectors are for all practical purposes the same, even though they are in opposite order and some of the vectors are of an opposite sign.

Finally the condition number computed by using the largest and smallest eigenvalues is the same, for eleven digits, as the MATLAB built in function.

```
>> EigenSection

X =

   1.0e+08 *

   0.232218965188203   0.200073202325504   0.792656793085658
   0.200073202325504   0.231882574965245   0.440710210067659
   0.792656793085658   0.440710210067659   3.865385886143497


Eigen Vectors from Power Method
| 0.2058876205406, -0.4895866991729,  0.847298738168 |
| 0.1216340575205, -0.8463309565946, -0.5185837135515 |
| 0.9709868401576, 0.2098303502896, -0.1146986501147 |

Eigen Values from Power Method
|  408866758.5862,               0,               0 |
|               0,  23835616.65574,               0 |
|               0,               0,  246367.3877212 |
```

Project 2: Matrix Applications

```
Eigen Vectors from MATLAB
|  0.847298738168, 0.4895866991729, 0.2058876205406 |
| -0.5185837135514, 0.8463309565946, 0.1216340575205 |
| -0.1146986501146, -0.2098303502896, 0.9709868401576 |

Eigen Values from MATLAB
|  246367.3877212,               0,               0 |
|               0,  23835616.65574,               0 |
|               0,               0,  408866758.5862 |

Condition from eigenvalues and MATLAB =  1659.581498867,  1659.581498867
```

Appendix A: Script to Test the Reading of Data.

```
%% Reading Data

% Read in the csv file first.
tic % Start Timer
load( 'Data01.csv' ); % Read in csv file
T_CSV = toc; % Stop and record time.

% Read in Binary data
tic % Start timer
load Data02.mat % read in Binary Data
T_MAT = toc; % Stop and record time.

% Compute the maximum absolute relative error
MaxARE = max(max(abs(Data02-Data01)./abs(Data02)));

% Document results
fprintf( '\nIt took %g seconds to read the CSV file. ', T_CSV );
fprintf( '\nand it took %g seconds to read the MAT file. ', T_MAT );
fprintf( '\nThe maximum absolute relative eror was %g \n', MaxARE );

%% Curve Fitting

% Renaming Data02 for simplier equations
X = Data02;
Z = X';

% Compute total variation in dependent variable
St = (X(3,:)-mean(X(3,:)))*(X(3,:)-mean(X(3,:)))';

% Create large full coverage Vandermonde Vander =
[Z(:,1) Z(:,1).*Z(:,1) Z(:,1).*Z(:,2) ...
          Z(:,2) Z(:,2).*Z(:,2) ones(length(Z(:,1)),1) ];
Params = [];

% Solve full solution
p5 = pinv(Vander)*Z(:,3); % compute parameters
f5 = Vander * p5;         % Fitted curve
Sr = (Z(:,3)-f5)'*(Z(:,3)-f5); % Residual Error Spread

CD(6) = (St-Sr)/St;  %CD for full model

% Save off parameters for later comparison
Params(6,:) = p5;
 % Solve for case of dropping first column
(x).
Vander2 = Vander;
```

Project 2: Matrix Applications

```
Vander2(:,1) = randn(length(Vander(:,1)),1); p5_1 =
pinv(Vander2)*Z(:,3); % Solve for parameters f5_1 =
Vander2 * p5_1;       % Computed fitted curve
Sr = (Z(:,3)-f5_1)'*(Z(:,3)-f5_1); % Residual Error Spread
CD(1) = (St-Sr)/St; %CD
Params(1,:) = p5_1; % Save off parameters for later comparison
 % Solve for case of dropping first column
(x).
Vander2 = Vander;
Vander2(:,2) = randn(length(Vander(:,1)),1); p5_2 =
pinv(Vander2)*Z(:,3); % Solve for parameters f5_2 =
Vander2 * p5_2;       % Computed fitted curve
Sr = (Z(:,3)-f5_2)'*(Z(:,3)-f5_2); % Residual Error Spread
CD(2) = (St-Sr)/St; %CD
Params(2,:) = p5_2; % Save off parameters for later comparison
 % Solve for case of dropping first column
(x).
Vander2 = Vander;
Vander2(:,3) = randn(length(Vander(:,1)),1); p5_3 =
pinv(Vander2)*Z(:,3); % Solve for parameters f5_3 =
Vander2 * p5_3;       % Computed fitted curve
Sr = (Z(:,3)-f5_3)'*(Z(:,3)-f5_3); % Residual Error Spread
CD(3) = (St-Sr)/St; %CD
Params(3,:) = p5_3; % Save off parameters for later comparison
 % Solve for case of dropping first column
(x).
Vander2 = Vander;
Vander2(:,4) = randn(length(Vander(:,1)),1); p5_4 =
pinv(Vander2)*Z(:,3); % Solve for parameters f5_4 =
Vander2 * p5_4;       % Computed fitted curve
Sr = (Z(:,3)-f5_4)'*(Z(:,3)-f5_4); % Residual Error Spread
CD(4) = (St-Sr)/St; %CD
Params(4,:) = p5_4; % Save off parameters for later comparison
 % Solve for case of dropping first column
(x).
Vander2 = Vander;
Vander2(:,5) = randn(length(Vander(:,1)),1); p5_5 =
pinv(Vander2)*Z(:,3); % Solve for parameters f5_5 =
Vander2 * p5_5;       % Computed fitted curve
Sr = (Z(:,3)-f5_5)'*(Z(:,3)-f5_5); % Residual Error Spread
CD(5) = (St-Sr)/St; %CD
Params(5,:) = p5_5; % Save off parameters for later comparison


% Reduced set with only 2 terms plus a constant
n = [1 5 6]; pR =
pinv(Vander(:,n))*Z(:,3); fR =
Vander(:,n) * pR;
Sr = (Z(:,3)-fR)'*(Z(:,3)-fR);
CD(7) = (St-Sr)/St;


Params(7,n) = pR; % Save off parameters for later comparison


% Reduced set with only 1 term plus a constant
n = [5 6]; pR1 = pinv(Vander(:,n))*Z(:,3); fR1
= Vander(:,n) * pR1;
Sr = (Z(:,3)-fR1)'*(Z(:,3)-fR1);
CD(8) = (St-Sr)/St;


Params(8,n) = pR1; % Save off parameters for later comparison


% Save off CD's and Parameters in csv file
Double2CSV([CD; Params'], 'ResultsFitting.csv' );


%% Eigen vector and Eigen value Section
```

EECE 540 **Spring 2018**

Project 2: Matrix Applications

```
% Create X matrix
X = Data02 * Data02'

% Use power method to compute eigen vectors and values
[Vectors, Values] = EigenVV( X );

% Compute eigen vectors and values using MATLAB function
[v,d] = eig( X );

% Compute condition using eigen values previously found
eigenConditions = Values(1,1)/Values(3,3);

% Compute condition using MATLAB function
matlabConditions = cond(X);

% Print out results
fprintf( '\nEigen Vectors from Power Method\n');
for k = 1:length(Vectors(:,1))     fprintf('|
');
    fprintf('%15.13g, ', Vectors(k,1:end-1) );
fprintf('%15.13g',   Vectors(k,end) );
fprintf(' |\n');
end
fprintf( '\nEigen Values from Power Method\n');
for k = 1:length(Values(:,1))     fprintf('|
');
    fprintf('%15.13g, ', Values(k,1:end-1) );
fprintf('%15.13g',   Values(k,end) );     fprintf(' |\n');
end
fprintf( '\nEigen Vectors from MATLAB\n');
for k = 1:length(v(:,1))     fprintf('|
');
    fprintf('%15.13g, ', v(k,1:end-1) );
fprintf('%15.13g',   v(k,end) );
fprintf(' |\n');
end
fprintf( '\nEigen Values from MATLAB\n');
for k = 1:length(d(:,1))     fprintf('|
');
    fprintf('%15.13g, ', d(k,1:end-1) );
fprintf('%15.13g',   d(k,end) );
fprintf(' |\n');
end  fprintf( '\nCondition from eigenvalues and MATLAB = %15.13g, %15.13g\n',
...          eigenConditions, matlabConditions );
```