Project B

Due March 5, 2019

I Instruction.

The purpose of this project is to apply

- Matrix methods to fit data on a polynomial curve,
- Series methods to approximate a nonlinear function with prescribed precision,
- Matrix methods to find general formula of a recursive sequence.

Section II and IV are only for practice. Please do not include them in your report. Section III (the three problems) is what you write a report on. You are allowed to use any results/theorems/formulae from the textbook if you need (although the materials already covered in lectures should be sufficient for this project). If you use any result that is not covered in lectures, please cite it from the textbook, for example page number/theorem number/etc.

Due to large amount of computation, you will need to use a mathematical software called Matlab. You can install Matlab on your personal computer with OSU's license by going to the webpage `https://is.oregonstate.edu/service/software/matlab` and follow the instructions therein. Section IV are some basic commands on Matlab. Please practice on those commands to get acquainted with Matlab before proceeding to the project. The recommended order to work on this project is:

Section IV → Practice 1 → Problem 1 → Practice 2 → Problem 2 → Practice 3 → Problem 3.

If you use Matlab for any part of the report, please write the (main) commands that you use. You don't need to write all commands. **The report should be written coherently.** You should explain the methods/procedures that you use. Writing only Matlab commands without any explanation is not sufficient. Answers that are not supported by valid arguments will receive little or no credits. **You are strongly encouraged to work with your group members.**

II Practices.

Practice 1:

Consider 3 points on the plane: (1,2), (2,3), (3,1). Suppose one wants to find a polynomial whose graph passes through these points. The first question is what is the degree of the polynomial. Drawing a picture, you can see that these points are not colinear (i.e. not lying on the same line). They cannot be fit on a line. Thus, the degree must be at least 2. The general form of degree-2 polynomial is:

$$P(x) = ax^2 + bx + c.$$

Fitting the given points on the graph of $P(x)$ is equivalent to solving a system of 3 equations with 3 unknowns:

$$\begin{cases} a(1)^2 + b(1) + c &= 2 \\ a(2)^2 + b(2) + c &= 3 \\ a(3)^2 + b(3) + c &= 1 \end{cases}$$

The matrix form of this system is $AX = u$, where

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 4 & 2 & 1 \\ 9 & 3 & 1 \end{bmatrix}, \qquad X = \begin{bmatrix} a \\ b \\ c \end{bmatrix}, \qquad u = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}.$$

To enter A in Matlab, we type:

```
>> A = [1 1 1;4 2 1;9 3 1]
```

To enter u, we type:

```
>> u = [2;3;1]
```

The augmented matrix is obtained by attaching column b on the right hand side of the coefficient matrix A:

```
>> aug = [A u]
```

We then perform row operations on the augmented matrix to bring it to reduced row echolon form (RREF). Note that RREF is a special type of REF. Note that REF of a matrix is not unique, but RREF is. In practice, RREF is usually more preferred. One can do so by hand or use a built-in command of Matlab:

```
>> rref(aug)
```

which gives

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & -1.5 \\ 0 & 1 & 0 & 5.5 \\ 0 & 0 & 1 & -2 \end{array}\right]$$

(The vertical bar does not appear in Matlab, but let's imagine it is there.) From here we see that the system has a unique solution $a = -1.5$, $b = 5.5$, $c = -2$. At this point, perhaps you already see computational issue when the number of points is large. If one wants to fit 50 points on the graph of a polynomial, the degree of the polynomial should be at least 50 (unless in very special cases, for example when all 50 points lie on the same line). The problem of *polynomial curve fitting* is equivalent to solving a system of 50 equations and 50 unknowns. A pocket calculator is not powerful enough for such computation. This is why one needs a software like Matlab to perform this task.

Practice 2:

Suppose you are to program the exponential function on a simple calculator. Specifically, you want to allow users to enter any number between -1 and 1 (called x); then have the calculator return e^x within an error of $10^{-4} = 0.0001$. Suppose your calculator can do only basic operations: addition, subtraction, multiplication, and division. Combining these operations, it can compute the values of polynomials (and even rational functions, which are the quotient of two polynomials). Taylor polynomials are a means to approximate the function e^x by polynomials. The higher degree of Taylor polynomials, the better the approximation. Too high degree is not efficient because it will slow down the computation. On the other hand, too low degree may not result in good approximation.

Now that you are given the maximum allowed error, you want to compute a degree (ideally minimum) of the Taylor polynomial so that the error stays within 10^{-4}. Read carefully Example 4, page 124-126, in the textbook to see how to find such a degree.

Practice 3:

Consider matrix

$$A = \begin{bmatrix} 6 & -1 \\ 2 & 3 \end{bmatrix}$$

To compute the eigenvectors of a square matrix (or a linear map), we start with computing the eigenvalues by solving the equation

$$\det(A - \lambda I_2) = 0.$$

This is the problem of finding roots of a polynomial. In this case, there are two roots $\lambda_1 = 4$ and $\lambda_2 = 5$. Then you solve two systems of linear equations to find the corresponding eigenvectors for

λ_1 and λ_2 respectively: $v_1 = (1, 2)$ and $v_2 = (1, 1)$. Note that v_1 and v_2 are not unique. Let P be the matrix consisting of vector v_1 as its first column, and vector v_2 as its second column. Matrix D is a diagonal matrix with λ_1 as the first entry on the diagonal, λ_2 the second.

$$P = \begin{bmatrix} | & | \\ v_1 & v_2 \\ | & | \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix}, \qquad D = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} = \begin{bmatrix} 4 & 0 \\ 0 & 5 \end{bmatrix}$$

If P is invertible, theory guarantees that one has the identity: $A = PDP^{-1}$. In this case, A is said to be *diagonalizable* (referring to its relation with the diagonal matrix D). Equivalently, $D = P^{-1}AP$ (by multiplying both sides of the previous equation by P^{-1} to the left, and by P to the right). Raising A to power is quite simple thanks to many cancellations, for example:

$$A^3 = AAA = (PDP^{-1})(PDP^{-1})(PDP^{-1}) = PD\underbrace{P^{-1}P}_{I_2}D\underbrace{P^{-1}P}_{I_2}DP^{-1} = PD^3P^{-1}$$

In general,

$$A^n = PD^nP^{-1} = P\begin{bmatrix} \lambda_1^n & 0 \\ 0 & \lambda_2^n \end{bmatrix}P^{-1} = P\begin{bmatrix} 4^n & 0 \\ 0 & 5^n \end{bmatrix}P^{-1} \tag{1}$$

One can compute the inverse of P by hand because it is a 2-by-2 matrix. Alternatively, one can use the built-in function *inv* in Matlab:

```
>> inv(P)
```

Then one can perform matrix multiplication at (1) to find explicitly a formula of A^n. Please proceed to finish this task.

III Problems.

1. A surveyor attempts to measure the depth across a river. He samples of the depth as follows. First, he marks the width of the river by numbers -3, -2, -1, 0, 1, 2, 3. These marks are equally spaced. 0 is at the middle of the river; -3 and 3 are at the banks. In other words, the width extends along the x-axis from -3 to 3. The depth at position x is a function $y = f(x)$ (unknown). Then he obtains a chart of data:

Position (x)	-3	-2	-1	0	1	2	3
Depth (y)	0	-0.5	-0.8	-1	-1.2	-1.5	0

   Insisting that the depth $f(x)$ should be a nice and smooth function with respect to x, he approximates $f(x)$ by a polynomial $P(x)$.

   (a) What would you choose for the degree of P? (Answer is not unique.)

   (b) Determine this polynomial P.

   (c) Graph P.

   (d) Does function $P(x)$ look realistic to you as the depth of a river?

2. Suppose you want to program the cosine function $f(x) = \cos x$ on a "minimalist" calculator. A few trigonometric identities can help you narrow the range of inputs: first, the cosine function is periodic with period 2π. For this reason, you can assume that users will only enter numbers in the interval $[0, 2\pi]$. Secondly, $\cos x = \cos(2\pi - x)$. This implies that if the calculator can compute cosine of numbers in the interval $[0, \pi]$, users can infer the cosine of

numbers that are in $[\pi, 2\pi]$. Finally, $\cos x = -\cos(\pi - x)$. If the calculator can compute the cosine of numbers in $[0, \pi/2]$, users can deduce the cosine of numbers in $[\pi/2, \pi]$. Therefore, you only need to program the cosine function with inputs $x \in [0, \frac{\pi}{2}]$.

(a) Find a general formula for $f^{(n)}(x)$ (the n'th derivative of cosine).

(b) Find the n'th polynomial for $f(x)$.

(c) What degree of Taylor polynomial for $\cos x$ (about 0) is needed to approximate $\cos x$ for $0 \leq x \leq \pi/2$ to within the error of $10^{-4} = 0.0001$?

(d) What would be the suitable degree if you allowed users to enter any numbers in $[-30, 30]$?

3. Let x_0, x_1, x_2, x_3,... be a sequence defined recursively as follows:

$$\begin{aligned} x_0 &= 0, \\ x_1 &= 1, \\ x_{n+1} &= 2x_n + 3x_{n-1}. \end{aligned}$$

(a) Write the first 6 terms of this sequence.

(b) Put $y_n = \begin{bmatrix} x_{n+1} \\ x_n \end{bmatrix}$. What is y_0?

(c) Find a matrix A such that $y_n = Ay_{n-1}$ (matrix multiplication).

(d) Apply this formula for y_{n-1}, then for y_{n-2}, and so on until you get an expression for y_n that involves only n, A and y_0.

(e) By using eigenvalues and eigenvectors as illustrated in Practice 3, find the general formula of A^n .

(f) Find the general formula of x_n in terms of n.

IV Supplementary Material.

MATLAB is a common and powerful tool in science and engineering. The following link contains instruction on how to download and install MATLAB on your computer `https://is.oregonstate.edu/service/software/matlab`. Let us start with some simple commands.

1. Start Matlab

2. Enter the following commands one by one

```
1+2
x=1+2
x
x;
y=x^2+2*x+1/x;
y
```

Notice the use of semicolon.

3. To enter a row vector, simply type e.g.

```
A = [1 2 -1 3]
```

or

```
A = [1,2,-1,3]
```

To enter a column vector, use semicolon instead of space or comma

```
B = [1;2;-1;3]
```

Now try

```
A(1)
B(3)
```

You can see that the starting index of entries is 1 (not 0).

4. Try the following commands

```
C = [1 2 3;4 5 6;7 8 9;10 11 12]
size(C)
transpose(C)
C(1,:)
C(:,2)
C(2,:) = C(2,:) - 4*C(1,:)
```

Notice what each command does.

5. Try the following commands

```
x = 3:15
y = 3:2:15
z = 15:-3:3
length(y)
size(x)
```

Notice what each command does.

6. The following commands create special matrices.

```
ones(3)
zeros(2,5)
eye(4)
eye(2,4)
eye(4,2)
```

7. Let us discuss how to plot the graph of function $y = x^2 + 2x - 3$ on the interval $[-4, 2]$. First, we discretize the interval $[-4, 2]$ by sample points, say

```
h = 1
x = -4:h:2
```

After this command, we see that x is a row vector of sample points with spacing $h = 1$. We want to compute y at each point. This is done by the command

```
y = x.^2 + 2*x - 3
```

Note that the dot in x.^2 is to tell Matlab that we want to square each entry of x. In other words, we want to square component by component. Without the dot, Matlab will interpret x^2 as x*x (multiplying row x by itself), which does not make sense. Now y is a row vector of the same length as x. Each point $(x(1), y(1))$, $(x(2), y(2))$,... belongs to the graph. A rough rendition of the graph is obtained by connecting these points by straight line segments. This is done by the command

```
plot(x,y)
```

To get a more exact graph, one should decrease the spacing size, for example choose $h = 0.1$. You can use arrow keys (Up or Down) to recall and edit previous commands.

```
h = 0.1
x = -4:h:2
y = x.^2 + 2*x - 3
plot(x,y)
```

8. Now try

```
x
y
clear x y
x
y
```

Notice what the command "clear" does. To erase the command window, type

```
clc
```