Project A

Due March 5, 2019

I Instruction.

The purpose of this project is to apply

- Matrix methods to balance chemical equations,
- Series methods to evaluate definite integrals,
- Matrix methods to model a population problem in ecology.

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 a system of 3 equations and 3 unknowns:

$$\begin{cases} x + 2y = 5 \\ y - 3z = 5 \\ 3x - z = 4 \end{cases}$$

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

$$A = \begin{bmatrix} 1 & 2 & 0 \\ 0 & 1 & -3 \\ 3 & 0 & -1 \end{bmatrix}, \qquad X = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \qquad b = \begin{bmatrix} 5 \\ 5 \\ 4 \end{bmatrix}.$$

To enter A in Matlab, we type:

```
>> A = [1 2 0;0 1 -3;3 0 -1]
```

To enter b, we type:

```
>> b = [5;5;4]
```

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

```
>> aug = [A b]
```

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

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

(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 $x = 1$, $y = 2$, $z = -1$.

Practice 2:

If function $f : [a, b] \to \mathbb{R}$ has an antiderivative F, one can compute definite integral of f over the interval $[a, b]$ using the Fundamental Theorem of Calculus:

$$\int_a^b f(x)dx = F(b) - F(a).$$

In 1830s, Joseph Liouville showed that there are elementary functions (i.e. functions that are constructed from the polynomials, trigonometric functions, inverse trigonometric functions, exponential functions, logarithm functions, and combinations of those using addition, subtraction, multiplication and division) whose antiderivatives are non-elementary functions. To name a few:

$$\frac{\sin x}{x},\ \cos(x^2),\ e^{x^2},\ e^{e^x},\ \sqrt{1-x^3}, \ldots$$

In fact, in 1960s Robert Risch developed an algorithm to check if a function has an antiderivative that is an elementary function. With such a function, one needs a numerical method to approximate its definite integral. Perhaps you have once used Riemann sum (left-point, right-point, midpoint, trapezoid rule,...) to approximate a definite integral. Taylor polynomial/series provides an alternate numerical method: replace the function f by a Taylor polynomial, the antiderivative of which is known. For example, consider

$$\int_0^1 \cos(x^2)dx$$

Here $f(x) = \cos(x^2)$. The 4th order Taylor polynomial of $f(x)$ is

$$T_4(x) = 1 - \frac{x^4}{2}.$$

Thus,

$$\int_0^1 \cos(x^2)dx \approx \int_0^1 \left(1 - \frac{x^4}{2}\right) dx = \left(x - \frac{x^5}{10}\right)\Big|_0^1 = 0.9$$

Practice 3:

To compute the eigenvectors of a square matrix (or a linear map), we usually start with computing

the eigenvalues by solving the equation

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

This is the problem of finding roots of a polynomial. The degree of the polynomial is equal to the size of matrix A. If A has size greater than 4, we run into trouble. In fact, it was proven by Ruffini and Abel around 1799–1825 that a general polynomial of degree 5 or higher cannot be solved by radicals. The strategy is to avoid solving for *all* eigenvalues before finding the eigenvectors. There exist many numerical method to do so. Let us consider a simple method called *power iteration*. This method only produces the largest (in sense of absolute value) eigenvalue and a corresponding eigenvector. Such an eigenvalue is called *dominant eigenvalue*. The direction of the corresponding eigenvector is called a *principal direction*. This direction is important in many applications. The power iteration method only works under certain conditions:

- The matrix has n distinct eigenvalues.
- The absolute values of the eigenvalues are also distinct.

If you pick randomly n^2 real numbers to form an $n \times n$ matrix, almost surely the resulting matrix will satisfy the above conditions. Therefore, the conditions are almost always satisfied. Let us consider an example:

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

To find a principal direction of A, we pick some direction to start with, say $v_0 = (2, 3)$. Then update the direction by the following procedure:

- The updated direction is Av_0. The normalized direction is vector $v_1 = Av_0/\|Av_0\|$ where $\|Av_0\|$ is the magnitude of vector Av_0.
- The updated direction is Av_1. The normalized direction is vector $v_2 = Av_1/\|Av_1\|$.
- (continue this procedure)

Drawing a picture for each step will help you visualize this procedure. The more steps we make, the closer the resulting direction (v_m after m steps) is to the principal direction (an eigenvector corresponding to the largest eigenvalue). The reason why we normalize the direction is to prevent the magnitude of the resulting vector from growing too large. Theory guarantees that after many steps the resulting direction seems to settle on some direction v. This vector v is a fixed point of the loop, i.e.

$$v = \frac{Av}{\|Av\|}$$

If we denote $\lambda = \|Av\|$ then $Av = \lambda v$. Then λ is an eigenvalue (it is in fact the dominant eigenvalue), and v is a corresponding eigenvector. There is a subtle issue worth noting: for the procedure to converge, the dominant eigenvalue must be positive. For simplicity, we will only consider examples when this is the case. The recursive procedure above can be programed in Matlab as a function:

```
function v = powerIter(A,v0,m)
v = v0
for i = 1:m
    v = A*v/norm(A*v)
end
```

The inputs are: matrix A, the initial direction v_0 (which is a column vector), and the number of steps m. The output is the direction at the m'th step. (The resulting directions at previous steps

are also printed out.) Copy the code segment above to a new script file (.m) and save under the name powerIter.m (Make sure that the file's name is the same as the function's name, which is "powerIter" in this case.) Now apply this function for $m = 30$ for example.

```
>> v = powerIter(A,v0,30)
```

Since what step do you notice that the resulting direction starts to converge? The limiting direction seems to be $v = (0.7071,\ 0.7071)$. This is an eigenvector corresponding to the dominant eigenvalue:

```
>> lambda = norm(A*v)
```

which gives $\lambda = 4$. "Norm" is a built-in command in Matlab to compute the magnitude of a vector. Now try using a different initial direction, say $v_0 = (50,\ 39)$. Will the limiting direction and dominant eigenvalue change?

Another way to look at the above procedure is that v_m, the resulting direction at the m'th step, is given by

$$v_m = \frac{A^m v_0}{\|A^m v_0\|}.$$

You can verify by hand this formula for small values of m without difficulty. This is why the method has the name "power iteration". If v_m converges to v as $m \to \infty$, then $A^m v_0$ tends to be aligned with (i.e. parallel to) v, the principal direction of A, for large m regardless of the choice of v_0.

III Problems.

1. Balance the following chemical equation

   $$\mathrm{KMnO_4 + HCl \to KCl + MnCl_2 + H_2O + Cl_2}$$

   In other words, find intergers $x_1,\ x_2,\ldots,\ x_6$ such that

   $$x_1\mathrm{KMnO_4} + x_2\mathrm{HCl} = x_3\mathrm{KCl} + x_4\mathrm{MnCl_2} + x_5\mathrm{H_2O} + x_6\mathrm{Cl_2}$$

2. Use the 4th order Taylor polynomial to approximate the definite integral:

   $$\int_0^1 e^{-\frac{x^2}{2}}\,dx$$

   The integrand plays an important role in Probability Theory, called the *density function of the normal distribution* (modulo a constant factor).

3. Consider an ecosystem consisting of lions, hyenas and antelopes. Each species is counted at the end of each year. To give an approximate mathematical model for the system, we adopt the convention that the population (i.e. the number of individuals) is not necessarily a whole number. Denote by l_n, h_n, a_n the population of lions, hyenas and antelopes respectively counted at the end of year n. Over many years, ecologists observe a pattern:

   - The population of lions is equal to 101% of their population of the previous year (due to minimal reproduction rate), minus 1% of the hyena population of the previous year (due to food competition and fightings), plus 3% of the antelope population of the previous year (due to food supply).
   - The population of hyenas is equal to 102% of their population of the previous year, minus 1% of the lion population of the previous year, plus 5% of the antelope population of the previous year.

- The population of antelopes is equal to 140% of their population of the previous year, minus 10% of the lion population of the previous year, minus 10% of the hyena population of the previous year.

(a) Express l_{n+1} as a linear combination of l_n, h_n, a_n. Similar question for h_{n+1} and a_{n+1}.

(b) Let v_n be the column vector whose entries are l_n, h_n, a_n. Express the relation between v_{n+1} and v_n in matrix form.

(c) Suppose the initial populations ($n = 0$) are 7 lions, 15 hyenas and 100 antelopes. Determine the population of each species at the end of the sixth year.

(d) The ratio of lion : hyena : antelope at year n is defined as

$$\frac{l_n}{s_n} : \frac{h_n}{s_n} : \frac{a_n}{s_n}$$

where $s_n = l_n + h_n + a_n$. For example, the initial ratio of the three species is

$$7 : 15 : 100 = \frac{7}{122} : \frac{15}{122} : \frac{100}{122} \approx 5.74\% : 12.30\% : 81.96\%$$

Determine the ratio of lion : hyena : antelope at an ecosystem equilibrium (i.e. the limiting ratio over the years). Does it change when you vary the initial populations l_0, h_0, a_0?

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. Now try

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

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

```
clc
```