LAB 1: Matrix and Vector Computations

(Math 250: Sections C3)

Name: Aryan Malhotra

Last 4 digits of RUID: 8256

Date: 10/26/2024

Learning Outcomes

- How to create matrices and vectors.
- How to manipulate matrices and create matrices of special types.
- How to add matrices and multiply a vector by a matrix.
- How to find the reduced row echelon form of a matrix.

Table of Contents

Random Seed

Initialize the random number generator by typing the following code, where abcd are the last four digits of your RUID.

```
rng('default');
rng(abcd, 'twister');
```

```
% Enter your code here
rng('default');
rng(8256, 'twister');
```

This will ensure that you generate your own particular random vectors and matrices.

The lab report that you hand in must be your own work. The following problems use randomly generated matrices and vectors, so the matrices and vectors in your lab report will not be the same as those of other students doing the lab. Sharing of lab report files is not allowed in this course.

Question 1. Creating Matrices and Vectors (Total Points - 8)

(a) ***Exercise (Points - 1):*** The Matlab command ***rand(m, n)*** creates an m × n matrix whose entries are random real numbers between 0 and 1. Type ***R = rand(2, 3)***. Then use the up-arrow key to generate two more samples of the random matrix R. Notice how the entries in R change each time the command is executed.

```
% Enter your code here
R = rand(2, 3)
```

```
R = 2×3
    0.9025    0.4394    0.9764
    0.1067    0.0050    0.7913
```

(b) ***Exercise (Points - 3)***: You can create a matrix in Matlab by typing the entries in the matrix between square brackets, one row at a time. To separate the entries in the same row, type a comma or press the space bar. To indicate the beginning of a new row, type a semicolon or press the ***Enter*** key. Try this by typing

```
A = [1 2; 3 4; 5 6]
```

```
% Enter your code here
A = [1 2; 3 4; 5 6]
```

```
A = 3×2
     1     2
     3     4
     5     6
```

You could also generate this matrix by pressing the ***Enter*** key at the end of each row, instead of typing a semicolon.

```
% Enter your code here
A = [
    1 2
    3 4
    5 6
    ]
```

```
A = 3×2
     1     2
     3     4
     5     6
```

Now use Matlab to create the following matrix, row vector and column vector:

$$B = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, \quad x = [4 \quad 3 \quad 2], \quad X = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}.$$

```
% Enter your code here
B = [
    1 2 3
    4 5 6
    7 8 9
    ]
```

```
B = 3×3
     1     2     3
     4     5     6
     7     8     9
```

```
x = [4 3 2]
```

```
x = 1×3
     4     3     2
```

```
X = [1;2;3]
```

```
X = 3×1
     1
     2
     3
```

Next, type the names of each of these matrices and vectors that you have created at the Matlab prompt. Note that x and X are different objects; Matlab is *case sensitive*. Finally, type whos at the prompt to get a list of all the matrices and vectors that are in your current Matlab workspace.

```
% Enter your code here
whos
```

```
  Name      Size            Bytes  Class     Attributes

  A         3x2                48  double
  B         3x3                72  double
  C         3x2                48  double
  D         2x3                48  double
  R         2x3                48  double
  S         4x2                64  double
  X         3x1                24  double
  a32       1x1                 8  double
  ans       2x1                16  double
  u         3x1                24  double
  v         3x1                24  double
  x         1x3                24  double
  y         1x2                16  double
  z         1x11               88  double
```

(c) ***Exercise (Points - 1)*****:** The Matlab ***size*** command determines the number of rows and columns in a matrix. Every Matlab command is documented in a ***help*** file, which you can access during a Matlab session. Type ***help size*** now to get information about this command.

Then use the ***size*** command to create a 4 × 2 matrix whose rows are the sizes of A, B, X, x (in that order), by typing

```
[size(A); size(B); size(X); size(x)]
```

```
% Enter your code here
[size(A); size(B); size(X); size(x)]
```

```
ans = 4×2
     3     2
     3     3
     3     1
     1     3
```

Give this matrix the name **S** by typing ***S = ans***. Note that all matrices which occur in Matlab must have names; if a matrix is unnamed, then it gets assigned the temporary name of 'ans' at the moment that it is created.

```
% Enter your code here
S = ans
```

```
S = 4×2
     3     2
     3     3
     3     1
     1     3
```

At the end of Lab 1 and each future lab, you will be instructed to remove from your write-up certain commands, including ***help***, along with their outputs. In particular, be sure to remove from your write-up the command ***help size***, along with its output.

(d) ***Exercise (Points - 1)*****:** To access a given entry in a matrix, put the row and column number in parentheses following the matrix name. Type ***a32 = A(3,2)*** and check that ***a32*** is the (3,2) entry in A defined above.

```
% Enter your code here
a32 = A(3,2)
```

```
a32 = 
6
```

```
A
```

```
A = 3×2
     1     2
     3     4
     5     6
```

Observe that the equal sign **=** in Matlab (as in other programming languages) executes a *substitution*: the current value of the variable on the right side of the equal sign is placed into the location whose name is on the left side. Type ***A(3,2) = 7*** and check that the (3,2) entry of A is now 7.

```
% Enter your code here
A(3,2) = 7
```

```
A = 3×2
     1     2
     3     4
     5     7
```

Now change the (3,2) entry of A back to 6. One way to do this is to type ***A(3,2) = 6***. Another way that you should try for future use is the ***up-arrow*** key ↑. This lets you cycle through the commands that you have already typed. When you get to the command that generated A, press Enter. If you go too far with the ***up-arrow***, you can use the ***down-arrow*** key ↓.

```
% Enter your code here
A(3,2)=6
```

```
A = 3×2
     1     2
     3     4
     5     6
```

(e) *Exercise (Points - 2)*: To access a whole row or column of a matrix, use the colon operator. For example, A(:, 2) is the second column of A, while B(1,:) is the first row of B. Type the following code to create a 3 × 2 matrix C whose first column is the first column of B and whose second column is the third column of B.

```
     C(:,1) = B(:,1); C(:,2)= B(:,3)
```

```
% Enter your code here
C(:,1)=B(:,1); C(:,2)=B(:,3)
```

```
C = 3×2
     1     3
     4     6
     7     9
```

Then use the colon operator to create a 2 × 3 matrix D whose first row is the first row of B and whose second row is the third row of B.

```
% Enter your code here
D(1,:)=B(1,:); D(2,:)=B(3,:)
```

```
D = 2×3
     1     2     3
     7     8     9
```

Use Matlab to display the matrices C and D by entering

```
C, D
```

```
% Enter your code here
C, D
```

```
C = 3×2
     1     3
     4     6
     7     9
D = 2×3
     1     2     3
     7     8     9
```

Question 2. Block Matrices and Special Matrices (Total Points - 3)

With Matlab it is easy to form new matrices from those that are already in the workspace and to create matrices of special types.

(a) ***Exercise (Points - 2)*****:** You can create *block matrices* by putting two matrices side by side (if they have the same number of rows), or one on top of the other (if they have the same number of columns). Use the matrices A, B, C, D, X created in Question 1 to make the following block matrices (the semicolon means that the matrices are stacked one on top of the other). *Before typing the Matlab commands, insert a comment line that lists which of the matrix combinations in this list fit together.* Then use Matlab (you will get error messages when the matrix sizes are not compatible).

```
[A X] [B C] [C D] [C;B] [B;D]
```

```
% Insert your comment here
% [A X] (same column size)
% [B C] (same column size)
% [B; D] (same row size)
```

```
% Enter your code here
[A X]
```

```
ans = 3×3
     1     2     1
     3     4     2
     5     6     3
```

```
[B C]
```

```
ans = 3×5
     1     2     3     1     3
     4     5     6     4     6
     7     8     9     7     9
```

```
[B;D]
```

```
ans = 5×3
     1     2     3
     4     5     6
     7     8     9
     1     2     3
     7     8     9
```

```
[C D]
```

```
Error using horzcat
Dimensions of arrays being concatenated are not consistent.
```

```
[C;B]
```

(b) ***Exercise (Points - 1)***: Type each of the following commands that generate special matrices.

```
      eye(4)   zeros(3)   zeros(3,5)   ones(2,3)   diag([4  5  6  7])
```

```
% Enter your code here
eye(4)
```

```
ans = 4×4
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
```

```
zeros(3)
```

```
ans = 3×3
     0     0     0
     0     0     0
     0     0     0
```

```
zeros(3,5)
```

```
ans = 3×5
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
```

```
ones(2,3)
```

```
ans = 2×3
     1     1     1
     1     1     1
```

```
diag([4,5,6,7])
```

```
ans = 4×4
     4     0     0     0
     0     5     0     0
     0     0     6     0
     0     0     0     7
```

More about Matlab:

The following commands and their output should be removed from your report before you finalize it. Your next answer that goes into your final report will be for Question 3 below.

Suppressing Displays:

When you place a semicolon at the end of a command, the command will be executed but the result will not be displayed on the screen. This is very useful when you are creating big matrices. Try typing ***z = [3:0.2:5];*** to create a long row vector with equally spaced entries(note the semicolon at the end). Then type ***z*** to display the vector z.

Format Commands:

You can control how numbers are displayed on the screen (this does not affect the internal arithmetic). Type

```
       y = [4/3  1.2345e-6];
```

```
% Enter your code here
%
```

Then observe the values you get for y when you type the following:

```
       format short, y
       format long, y
       format short e, y
       format rat, y
```

```

```

Be sure to remove these sample Matlab commands and their output from your final Lab write-up.

Question 3. Matrix Addition and Matrix-Vector Multiplication (Total Points - 4)

Before continuing with the lab, enter ***format short*** and ***format compact*** at the Matlab prompt. Keep these two commands in your report when you finalize your diary file.

```
% Enter your code here
format short
format compact
```

Generate vectors u and v and matrices A and B by typing

```
       u = fix(10*rand(3,1)), v = fix(10*rand(3,1)),
       A = fix(10*rand(2,3)),  B = fix(10*rand(2,3))
```

```
% Enter your code here
u = fix(10*rand(3,1)), v = fix(10*rand(3,1)),
u = 3×1
     3
     2
     0
v = 3×1
     1
     8
     7
```

```
A = fix(10*rand(2,3)),  B = fix(10*rand(2,3))
```

```
A = 2×3
     1     5     1
     1     0     3
B = 2×3
     9     3     4
     2     1     4
```

These matrices have entries randomly selected from the numbers 0, 1, . . . , 9.

(a) ***Exercise (Points - 2):*** To obtain a linear combination sA+tB of the matrices A and B (where s and t are scalars) using Matlab, you must type ***s*A + t*B.*** This is only defined when A and B are the same size. The transpose A^T of A is obtained by typing ***A'***.

Calculate the following using Matlab.

$$A + B, \quad B + A, \quad 6B, \quad 2(3B), \quad 6A + 15B, \quad 3(2A + 5B), \quad 3A, \quad (3A^T)^T$$

```
% Enter your code here
A+B
```

```
ans = 2×3
    10     8     5
     3     1     7
```

```
B+A
```

```
ans = 2×3
    10     8     5
     3     1     7
```

```
6*B
```

```
ans = 2×3
    54    18    24
    12     6    24
```

```
2*(3*B)
```

```
ans = 2×3
    54    18    24
    12     6    24
```

```
6*A + 15*B
```

```
ans = 2×3
```

```
   141    75    66
    36    15    78
```

```
3*(2*A + 5*B)
```

```
ans = 2×3
   141    75    66
    36    15    78
```

```
3*A
```

```
ans = 2×3
     3    15     3
     3     0     9
```

```
(3*A')'
```

```
ans = 2×3
     3    15     3
     3     0     9
```

Insert comments below that explain the properties of matrix addition and scalar multiplication that these calculations illustrate.

```
% Insert your comment here
% We see that A+B = B+A for any 2 matrices where matrix addition holds
(associative)
% 6B = 2(3B) means that order of scalar multiplication is irrelevent
(commutative)
% 3(2A + 5B) = 6A + 15B meaning scalar multiplication of vectors is
distributive
% Lastly, 3A = (3A')' depicting that (A')' = A
```

(b) ***Exercise (Points - 2):*** To obtain the matrix-vector product Au using Matlab, you must type A*u (remember that a vector is just a matrix with one column).

Calculate the following using Matlab.

$\mathrm{Au} + \mathrm{Av}, \quad A(u+v), \quad (A+B)u, \quad \mathrm{Au} + \mathrm{Bu}, \quad A(3u), \ (3A)u$

```
% Enter your code here
A*u + A*v
```

```
ans = 2×1
    61
    25
```

```
A*(u+v)
```

```
ans = 2×1
    61
    25
```

```
(A+B)*u
```

```
ans = 2×1
    46
    11
```

```
A*u + B*u
```

```
ans = 2×1
    46
    11
```

```
A*(3*u)
```

```
ans = 2×1
    39
     9
```

```
(3*A)*u
```

```
ans = 2×1
    39
     9
```

Insert comments below that explain the properties of the matrix-vector product that these calculations illustrate.

```
% Insert your comment here
% A(u+v) = Au + Av, suggesting that matrix vector multiplication is
% distributive (matrix can be distributed)
% (A+B)u = Au + Bu suggests that matrix vector multiplication is
% distributive (the vector can also be distributed... obviously since
vectors ARE matrices)
% The order in which a scalar is multiplied doesn't matter (scalar
% multiplication of matrices is commutative)
```

Question 4. Gaussian Elimination and Reduced Row-Echelon Form (Total Points - 5)

Now that you know how to do matrix calculations with Matlab, you can easily carry out the steps in Gaussian elimination. Before starting work on this question, type ***rrefmovie*** at the Matlab prompt.

```
% Enter your code here
```

If you get an error message *??? Undefined function or variable rrefmovie*, click on ***Set Path*** on the toolbar. Insert the name of the directory where you have copied the T-code rrefmovie.m, and save the result.

[IMPORTANT: You will need to do this in future labs also, when necessary.]

You will see a step-by-step example of the row operations that transform a matrix A into its reduced row echelon form R = rref(A). In this demonstration, each pivot is chosen to be the largest in its column (for numerical stability), so extra row interchanges are used. Since rref(A) is uniquely determined by A, this does not affect the final answer. (Do not include this part in your lab write-up. Your answers for Question 4 will begin with the answer to part (a) below.)

(a) ***Exercise (Points - 2):*** Generate a 3 × 4 matrix A with random integer entries by the command

```
A = fix(10*rand(3,4))
```

```
% Enter your code here
A = fix(10*rand(3,4))
```

```
A = 3×4
     2     5     4     7
     7     3     3     3
     0     4     0     8
```

To transform A into R = rref(A), start with R = A. Normalize the first row of R to get R(1, 1) = 1:

```
R = A; R(1,:) = R(1,:)/R(1,1)
```

```
% Enter your code here
R = A; R(1,:) = R(1,:)/R(1,1)
```

```
R = 3×4
    1.0000    2.5000    2.0000    3.5000
    7.0000    3.0000    3.0000    3.0000
         0    4.0000         0    8.0000
```

(note the use of the colon to operate on whole rows of the matrix). If your random matrix happens to have A(1, 1) = 0, then interchange rows to get a nonzero entry in the (1, 1) position before doing the calculation above. Now subtract a multiple of the first row of R from the second row to make R(2, 1) = 0:

```
R(2,:) = R(2,:) - R(2,1)*R(1,:)
```

```
% Enter your code here
R(2,:) = R(2,:) - R(2,1)*R(1,:)
```

```
R = 3×4
    1.0000    2.5000    2.0000    3.5000
         0  -14.5000  -11.0000  -21.5000
         0    4.0000         0    8.0000
```

Repeat this procedure to make R(3, 1) = 0:

```
R(3,:) = R(3,:) - R(3,1)*R(1,:)
```

```
% Enter your code here
R(3,:) = R(3,:) - R(3,1)*R(1,:)
```

```
R = 3×4
    1.0000    2.5000    2.0000    3.5000
         0  -14.5000  -11.0000  -21.5000
         0    4.0000         0    8.0000
```

The first column of your matrix R should now be the same as the first column of rref(A).

(b) ***Exercise (Points - 1):*** Operate on R by the same method as in (a) to obtain the second column of rref(A). First normalize row 2 of R, then subtract multiples of row 2 from rows 1 and 3 to put zeros in the (1, 2) and (3,

2) positions. Be sure to refer to the entries in your matrix symbolically as was done in part (a); don't type explicit decimal numbers in the Matlab commands.

```
% Enter your code here
R(2,:) = R(2,:)/R(2,2)
```

```
R = 3×4
    1.0000         0    0.1034   -0.2069
         0    1.0000    0.7586    1.4828
         0    4.0000         0    8.0000
```

```
R(1,:) = R(1,:) - R(1,2) * R(2,:)
```

```
R = 3×4
    1.0000         0    0.1034   -0.2069
         0    1.0000    0.7586    1.4828
         0    4.0000         0    8.0000
```

```
R(3,:) = R(3,:) - R(3,2) * R(2,:)
```

```
R = 3×4
    1.0000         0    0.1034   -0.2069
         0    1.0000    0.7586    1.4828
         0         0   -3.0345    2.0690
```

(c) ***Exercise (Points - 1):*** Operate on R by the same method as in (b) to obtain the third column of rref(A). First normalize row 3 of R, then subtract multiples of row 3 from rows 1 and 2 to put zeros in the (1, 3) and (2, 3) positions.

```
% Insert your comment here
R(3,:) = R(3,:)/R(3,3)
```

```
R = 3×4
    1.0000         0    0.1034   -0.2069
         0    1.0000    0.7586    1.4828
         0         0    1.0000   -0.6818
```

```
R(1,:) = R(1,:) - R(1,3) * R(3,:)
```

```
R = 3×4
    1.0000         0         0   -0.1364
         0    1.0000    0.7586    1.4828
         0         0    1.0000   -0.6818
```

```
R(2,:) = R(2,:) - R(2,3) * R(3,:)
```

```
R = 3×4
    1.0000         0         0   -0.1364
         0    1.0000         0    2.0000
         0         0    1.0000   -0.6818
```

(d) ***Exercise (Points - 1):*** Your matrix R should now be transformed into rref(A) (since A is a random 3 × 4 matrix, rref(A) is (almost) sure to have rank 3). Check your answer by Matlab with the command ***rref(A)***. If the Matlab answer is not the same as the current value of your R, go back and redo your calculations.

```matlab
% Insert your comment here
rref(A)
```

```
ans = 3×4
    1.0000         0         0   -0.1364
         0    1.0000         0    2.0000
         0         0    1.0000   -0.6818
```