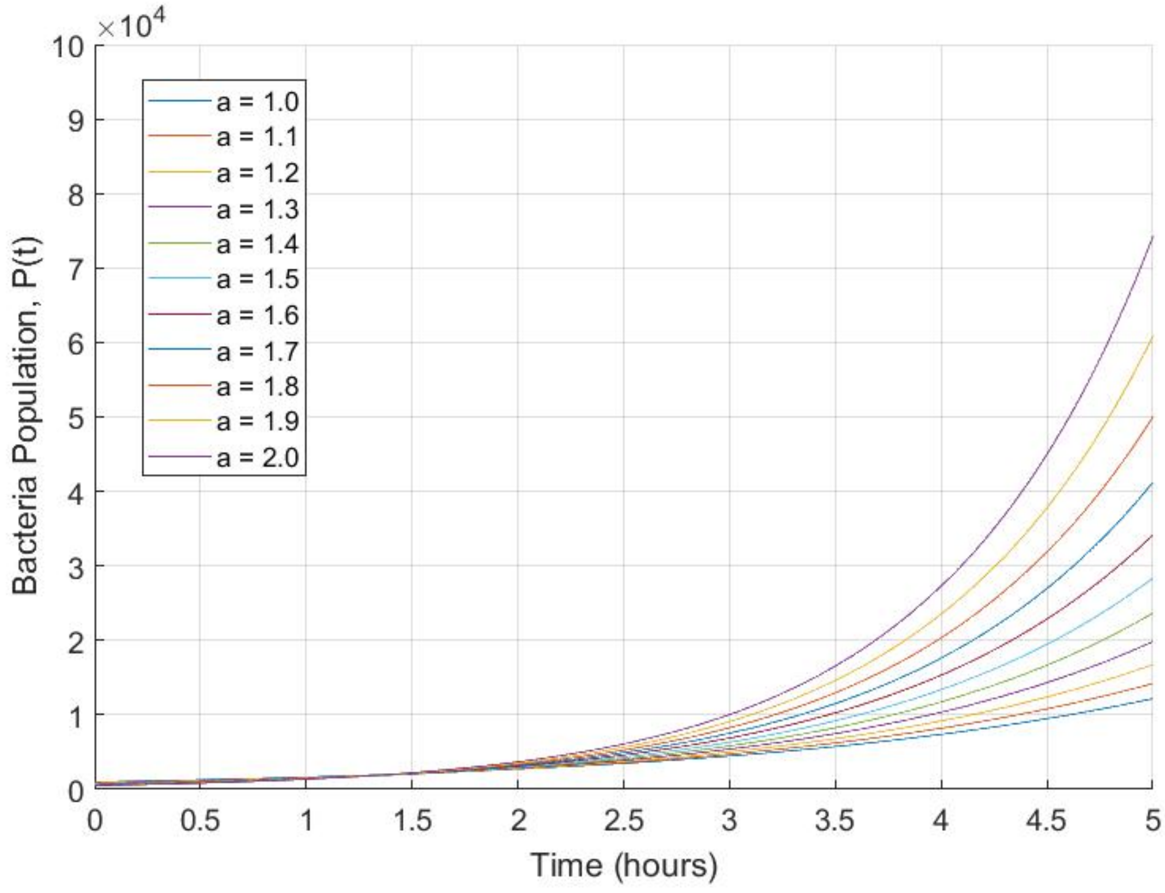
UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE & ENGINEERING
First Year Program – Core 8 and TrackOne

FIRST YEAR PROGRAM
ENGINEERING PROBLEM SOLVING LABS

MAT188: Laboratory #6
Matrices and For Loops



MATRICES AND FOR LOOPS

This laboratory will introduce you to matrices and how you can work with them using matrix operations in MATLAB. It will also introduce you to *for loops*, which allow you to repeat instructions a specific number of times.

Learning Outcomes

By the end of this lab students will…

1) Synthesize knowledge of multi-dimensional arrays (matrices) from class and apply it to perform basic matrix operations, including definition and indexing, and
2) Demonstrate basic understanding of repetitive tools, specifically 'for' loops, in matrix operations.

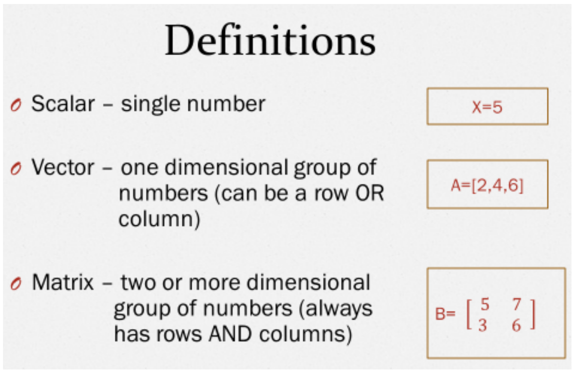
Preparation *(Required doing* ***before*** *you come to the laboratory session)*

1. Read through this lab document.

Related Reference Materials *(Not required, but may be a helpful resource)*

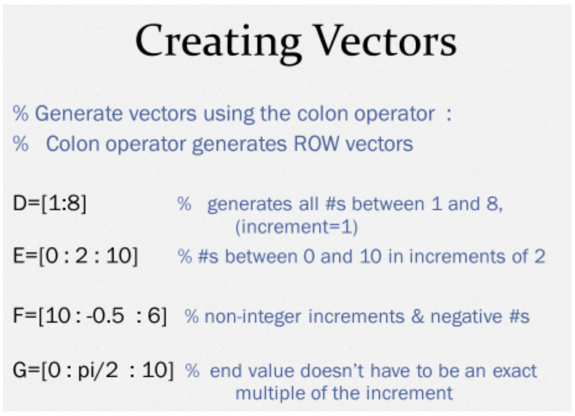
Video demonstration: *Working with Arrays in MATLAB (8:17)*
http://www.mathworks.com/videos/working-with-arrays-in-matlab-69022.html
Pay particular attention to the indexing of arrays after the 3:00 minute mark.

Review



Definitions

- Scalar – single number — X=5
- Vector – one dimensional group of numbers (can be a row OR column) — A=[2,4,6]
- Matrix – two or more dimensional group of numbers (always has rows AND columns) — $B=\begin{bmatrix} 5 & 7 \\ 3 & 6 \end{bmatrix}$



Creating Vectors

```
% Generate vectors using the colon operator :
%   Colon operator generates ROW vectors

D=[1:8]            %  generates all #s between 1 and 8,
                       (increment=1)
E=[0 : 2 : 10]     % #s between 0 and 10 in increments of 2

F=[10 : -0.5  : 6] % non-integer increments & negative #s

G=[0 : pi/2  : 10] %  end value doesn't have to be an exact
                       multiple of the increment
```

MATLAB Skills and Knowledge: Basic Matrix Definition, Operations and Indexing

Matrix Definition and Basic Operations

As you read through this lab document, work along with the module and enter each example command in the MATLAB command window to see the result.

A ***matrix*** of size $m \times n$ is a rectangular array of numbers with m rows and n columns. A ***row vector*** is a $1 \times n$ matrix, while a ***column vector*** is an $m \times 1$ matrix. A square matrix has the same number of rows as columns, for example:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

In MATLAB, commas or spaces are used to separate elements within a row and semicolons are used to separate rows. For example, define the above matrix:

```
>> A = [1 2 3; 4 5 6; 7 8 9]  % row by row input
```

You can use the colon operator to produce numbers in ranges:

```
>> A2 = [1:4; -1:2:5]
```

Notice how the semicolon separates the rows, and the colon (`:`) is used to specify a range of numbers as elements. `1:4` means from 1 to 4 in increments of 1, while `-1:2:5` means from -1 to 5 in increments of 2.

Most of the basic operators defined for scalars are defined in MATLAB for matrices. For example, to add two matrices:

```
>> B = [1 1 1; 2 2 2; 3 3 3]
>> C = [1 1 1; 2 2 2; 3 3 3]
>> B + C
```

Like with vectors, standard multiplication and division do not operate on the individual components. For example,

```
>> B*C
```

does not multiply the individual components of `B` and `C`. It performs a different operation called *matrix multiplication*, which you will learn about later this term in linear algebra (MAT188).

Like with vectors, to do *element by element* multiplication (multiply each element in one matrix by the corresponding element in the other matrix), we must prefix the operator with the dot operator (`.`), for example:

```
>> B.*C
```

You must also use the dot operator for element by element division (`./`) and exponentiation (`.^`).

Exercises:

Evaluate the following expressions using MATLAB and fill in the table.

First, define the matrices $A = \begin{bmatrix} 3 & 1 & 3 \\ 1 & 2 & 1 \\ 3 & 4 & 1 \end{bmatrix}$ and $B = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{bmatrix}$.

Table 1 – Matrix Expressions

Expression	Result
`A./B`	
`A.*B`	
`A.^2`	
`rank(A)`	
`rank(B)`	

Matrix Indexing

Specific elements of a matrix can be accessed using *indexing*, in which **index 1 represents the first element**. For example, consider the matrix A:

$$A = \begin{bmatrix} -1 & 0 & 5 \\ 6 & 2 & -3 \\ 10 & -4 & 8 \end{bmatrix}$$

```
>> A=[-1 0 5;6 2 -3;10 -4 8];
```

To get the element in the 1st row, 2nd column:

```
>> A(1,2)
```

Entire rows or columns can be selected using the colon operator:

```
>> A(1,:)
>> A(:,2)
```

Partial rows or columns, can be selected by limiting the range:

```
>> A(1:2,2)
```

We can assign individual elements to a particular location within the matrix:

```
>> A(3,2)=cos(pi)
```

Or make use of the colon operator to assign new values to a part of a matrix:

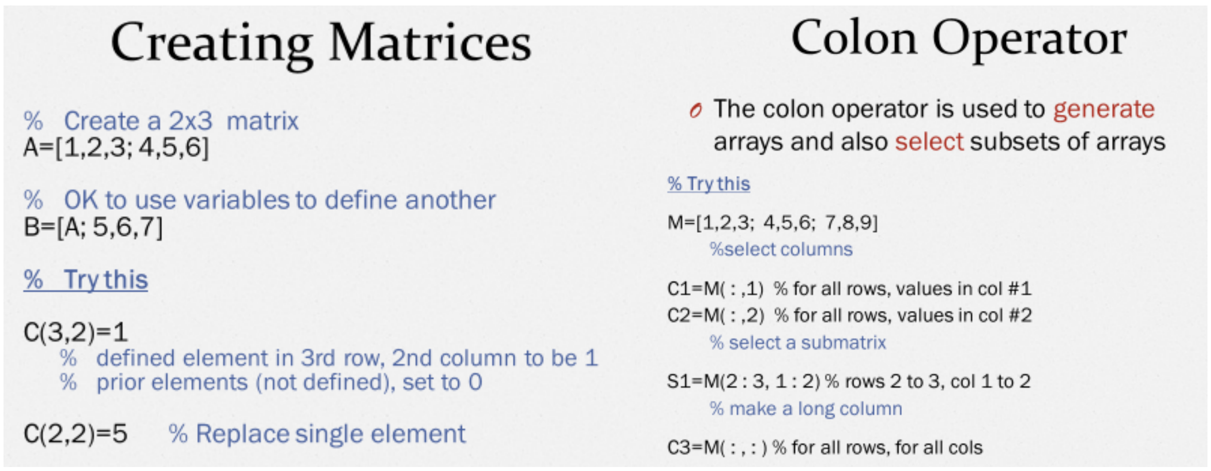
```
>> A(1:2,1) = [1;1]
```

Or entire rows:

```
>> x=[-2:2:2];
>> A(2,:) = x.*exp(-2*x)
```

Now, assign the value 0 to all the elements in the second column of A.

***Optional** – if you need some additional exercise in flexing your matrix muscle, try the following:*



Creating Matrices

```
% Create a 2x3 matrix
A=[1,2,3; 4,5,6]

% OK to use variables to define another
B=[A; 5,6,7]

% Try this

C(3,2)=1
   % defined element in 3rd row, 2nd column to be 1
   % prior elements (not defined), set to 0

C(2,2)=5    % Replace single element
```

Colon Operator

- The colon operator is used to generate arrays and also select subsets of arrays

```
% Try this

M=[1,2,3; 4,5,6; 7,8,9]
    %select columns

C1=M( : ,1)  % for all rows, values in col #1
C2=M( : ,2)  % for all rows, values in col #2
    % select a submatrix

S1=M(2 : 3, 1 : 2) % rows 2 to 3, col 1 to 2
    % make a long column

C3=M( : , : ) % for all rows, for all cols
```

MATLAB Skills and Knowledge:
For Loops

Open a new script. As you read through this section, type each example into the script window and run it.

Sometimes, you want to repeat a section of code multiple times. You can use a ***for loop*** to iterate over a range of values and repeatedly execute a set of commands for each. For example, define a vector of each of the numbers from 1 to 5 multiplied by 2. (This can be done more easily with a range, but just as an example.) Create a script to run the following and observe the output:

```
x=[]    % This vector will grow as we add values to it

for i=1:5
    x(i)=i*2
end
```

This for loop iterates over the range of numbers `1:5` and runs the code in the loop (until `end`) for each value. In this example, the loop defines the variable (also called the ***argument***) `i=1`, runs the code in the loop, returns to the beginning, defines `i=2`, runs the code in the loop, returns to the beginning, and so on, with the last iteration of the loop running with `i=5`.

The general form of the for loop is as follows:

```
for <variable> = <range>
    <code to repeat>
end
```

Exercise: Bacteria Growth and Interpretation (for submission)

You must use a *for loop* in this exercise. Observe that doing this is easier than repeatedly typing the same commands for every a value.

Consider various populations of bacteria whose populations are modelled by the function $P(t) = \frac{1000}{a}e^{0.5at}$, where $P(t)$ is the population, t is the time in hours, and a is some positive constant.

Your job is to create a plot of how different populations behave for different values of a. Use a `for loop` to go through values of a in the interval $[1,2]$ in steps of 0.1 ($a = 1, a = 1.1, a = 1.2 \ldots a = 2$), and plot all of them on the same figure (11 different functions). **Hint**: use `hold on` to keep all plots on the same figure.

Use your figure to answer the following questions:

1. Using your understanding of exponential functions, identify the general order or pattern of which function (a value) corresponds to which line.
2. Which population grows the fastest as $t \to \infty$? At what value of t does it becomes the biggest population? (you may need to use the `axis` function and the zoom tool in the figure window to see this. You may also want to adjust the domain of t values you are using.)
3. Choose three of the populations and read the figure to estimate the doubling time (amount of time it takes for the population to double) for each. Does this match what you can find analytically using the function expression? Can you find a relationship between the doubling time and a value?
4. In a paragraph or two, please discuss similarities/differences in doing matrix math by-hand versus via MATLAB. When is one more appropriate than another? What are some important considerations an engineer should think about when using “for” loops?

Submission is due Wednesday February 21th 11:59pm via MAT188 PRA website: 1-page PDF file is required, including appropriate screenshots (of the plot(s), not your code) and clear answers to the 4 questions above. Please write your name and student number on your submission for credit.