Lesson 2

Written by Brenda So and Cory Nezin

Objective

After this class, you should be able to:

- Understand how to perform vector operations in MATLAB
- Understand arithmetic and basic functions in MATLAB
- Know how to use for statements

Vector Operations

In lesson 1, we see how to make a vector with the colon operator and linspace, however, if MATLAB only allows us to use these two operations on vectors, it would be pretty useless. MATLAB is often used to perform numerical analysis, hence it would be very useful if we can input vectors into functions and use that to generate another vector. Luckily, that's what MATLAB is for! Note that when you perform an operation, they are element wise operation for vectors.

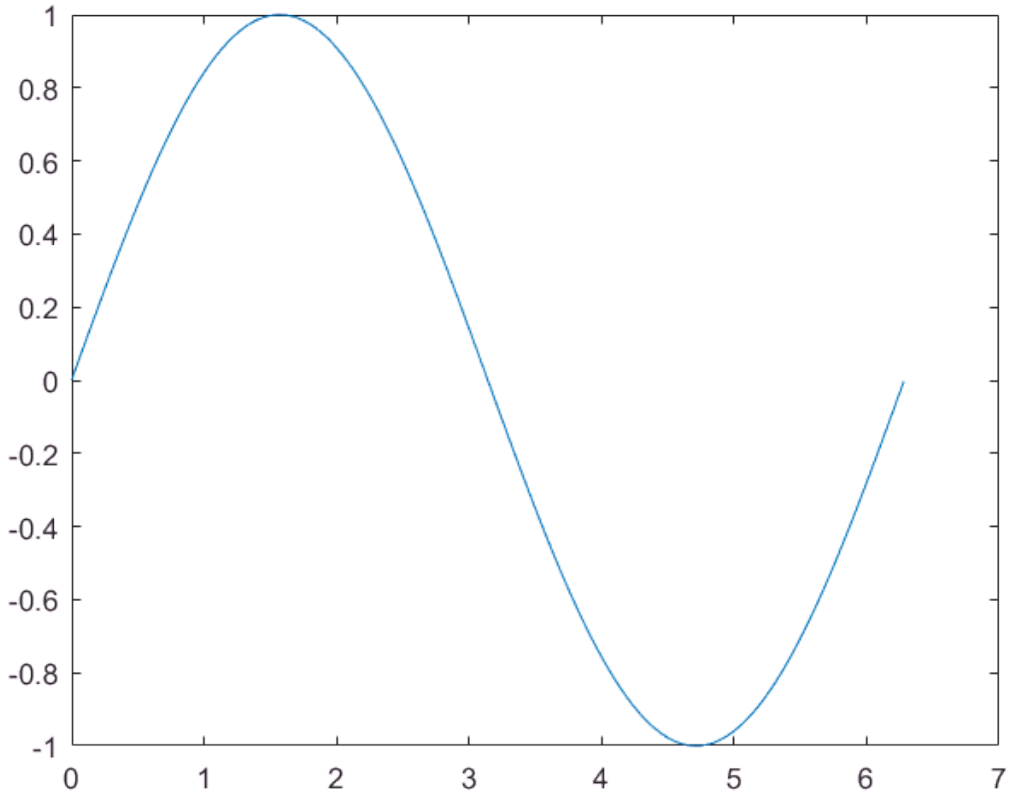
```
x = 0:0.01:2*pi;    % You create a vector
y = sin(x);         % Here, you perform a transformation from one vector to another
xlen = length(x)
```

```
xlen = 629
```

```
ylen = length(y)    % xlen and ylen is equal!
```

```
ylen = 629
```

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



Exercise 1 : Vector Operations

Create a vector t, where t ranges from 0 to 2 millisecond with increments T = 1e-6, and create another vector y1, where $f_0 = 50\,\mathrm{Hz}$ and $\beta = 10\,M\,\mathrm{Hz}\,s^{-1}$

$$y_1(t) = 10\cos\left(2\pi f_0 t + \pi\beta t^2\right)$$

```
%% WRITEME: create t and y1, and run this box. It would output something very interesting!
figure;
plot(t,y1)
```

```
Undefined function or variable 't'.
```

Basic Indexing in MATLAB

To inspect certain values in vectors x and y, we can use indexing. Now we are going to delve deeper into indexing next week, but here we are going to give you enough information to do the homework assignment

Exercise 2 : Basic Indexing

```
% WRITEME: Index the whole x vector
% WRITEME: Index the first element in x
```

```
% WRITEME: Index the fifth element in x
% WRITEME: Index the second to fifth element in x (inclusively)
% WRITEME: Index the last element in x
```

BE CAREFUL!

In other languages, indices start at 0, however, in MATLAB, indices start at 1. Hence in matlab, the last index would be the length of the vector itself.

Other vector operations

Of course, we can perform more complicated arithmetic operations on vectors, just like what we can do with scalars. What's more, there are pre-built functions in matlab that allows us to obtain certain statistics. There's a lot of such functions but we are only showing a few commonly used operations here

Exercise 3 : Other vector operations

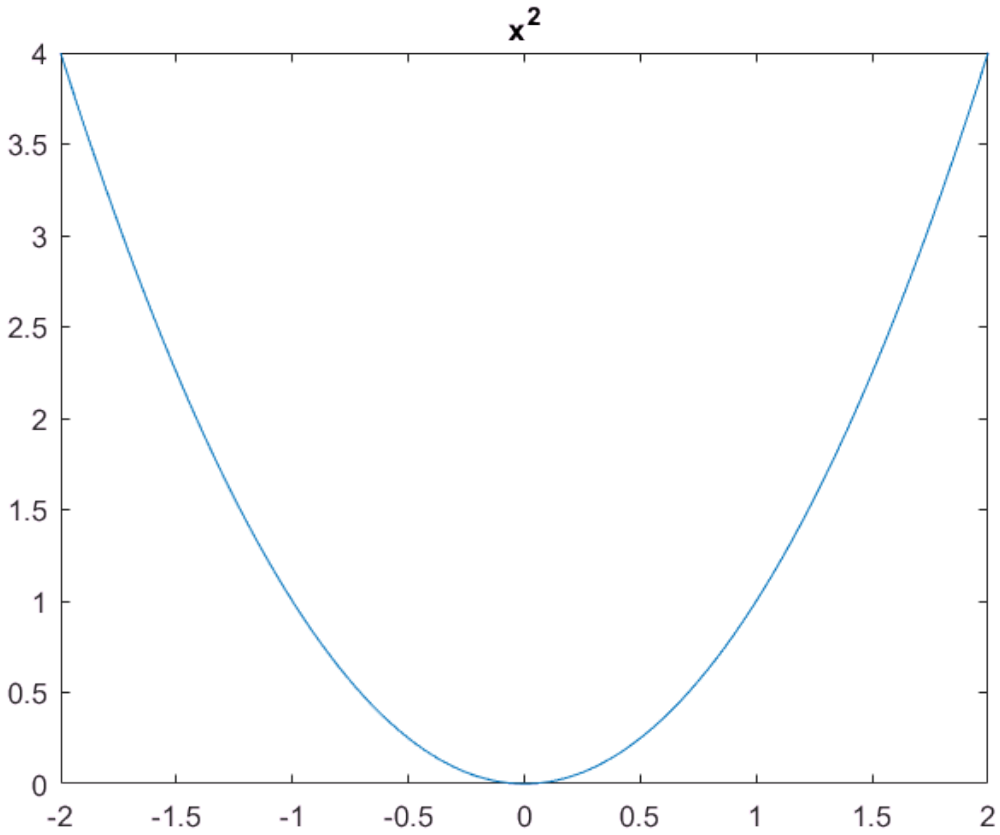
```
% WRITEME: calculate sum of x
% WRITEME: calculate average of x
% WRITEME: calculate minimum in x
% WRITEME: calculate difference between adjacent elements in x
```

Practical example of vector operations : Numerical Estimation of integrals and derivatives

Approximate Derivatives

Here we will look at some approximate numerical methods

```
x = linspace(-2,2,100);
y = x.^2;
plot(x,y)
title('x^2')
```



Now, we can approximate the derivative

$$\frac{\mathrm{dy}}{\mathrm{dx}} = \lim_{\Delta x \to 0} \frac{y(x + \Delta x) - y(x)}{\Delta x}$$

as

$$\frac{\hat{\mathrm{dy}}}{\mathrm{dx}} = \frac{y(x + \Delta x) - y(x)}{\Delta x} = \frac{\Delta y}{\Delta x}$$

where Δx is small

An easy way to do this in MATLAB is the *diff* command which returns the difference between every pair of consecutive numbers in a vector (or each column of an array).

```
dydx = diff(y)./diff(x);
plot(x,dydx)
```

```
Error using plot
Vectors must be the same length.
```

diff returns one less element than the original vector, this simple strategy returns an error. Fix this by plotting against a new x, *xhat.* Note that there are many valid ways to do this. You will need *xhat* later so make sure you get it!

Excercise 4: Approximate Derivative

```
%xhat = ?
plot(xhat,dydx)
```

Approximate Integrals

Now suppose we want to approximate the cumulative integral of a function, i.e.

$$Y(u) = \int_c^u y(x)\mathrm{dx} = \lim_{\Delta x \to 0} \sum_{c+n\Delta x \in [c,u]} y(c + n\Delta x)\Delta x$$

Again, we can approximate this by simply dropping the limit

$$\hat{Y} = \sum_{c+n\Delta x \in [c,u]} y(c + n\Delta x)\Delta x$$

The *cumsum* function in MATLAB returns the running sum of a vector, i.e.

$$\mathrm{cumsum}(v)_{1,n} = \sum v(1:n)$$

Use *cumsum* to approximate the running integral of y. Again, there are a few valid ways of doing this.

Excercise 5. Cumulative Integral

```
%Y = ?
plot(xhat,Y)
```

Now, use the approximate derivative to get the original function, y back as *yhat* and plot it. You may need to use/create another variable for the x axis when plotting.

Excercise 6. Anti-derivative.

```
%yhat = ?
%plot(?)
```

Matrix

Matrix is closely related to vectors, and we have also explored some matrix operations last class. This class, we are going to explore 3 functions that are very useful but are hard to grasp for beginners, namely reshape, meshgrid, row-wise and column-wise operations.

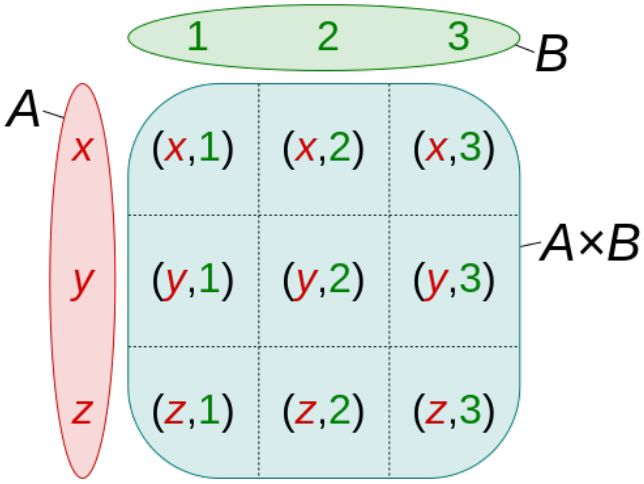
Reshape

When you are reshaping an array / matrix, the first dimension is filled first, and then the second dimension, so on and so forth

```
M = 1:100;
N1 = reshape(M,2,2,[]);    % It would create a 2*2*25 matrix
N2 = reshape(M,[2,2,25]);  % Same as N1
N2(:,:,25)                 % Gives you 97,98,99,100
N2(:,1,25)                 % Gives you 97 and 98
```

Meshgrid

Meshgrid is quite hard to understand. Think of it as a way to replicate arrays, like the following example:




```
a = 1:3;
b = 1:5
[A,B] = meshgrid(a,b)
```

You have created two arrays A and B, note that in A, the vector a is copied row-wise, while the vector b is transposed and copied column-wise. This is useful, because when you lay one above the other, you essentially create a CARTESIAN PRODUCT, and it is useful when we need to plot a 3D graph: https://en.wikipedia.org/wiki/Cartesian_product

Here is a more complicated example to show you when meshgrid is useful

```
a1 = -2:0.25:2;
b1 = a1;
[A1,B1] = meshgrid(a1);
% Here we plot the surface of f(x) = x*exp^(x.^2+y.^2)
F = A1.*exp(-A1.^2-B1.^2);
surf(A1,B1,F)
```

Row-wise / Column-wise operations

```
H = magic(4)     % create the magical matrix H
sum(H,1)         % column wise sum, note that this is a row vector(default)
fliplr(H)        % flip H from left to right
flipud(H)        % flip H upside down
H(1,:) = fliplr(H(1,:)) % flip only ONE row upside down
H(1,:) = []      % delete the first column
```

Exercise 7 : Matrix Operations

```
% WRITEME: Create a random 4-by-5 matrix H2 with randi that draws random numbers from the set
% WRITEME: Take the sum of second column of H2
% WRITEME: Take the average of the third row of H2
% WRITEME: reshape the 4-by-5 matrix into a 2-by-2-by-5 3D array, and name the array as C
% WRITEME: Delete the second row of each layer in C
```

Control Sequence - For loops

Similar to other languages, MATLAB have if, while and for control sequences. For loops is one of the commonly used control sequences in MATLAB. We will continue the discussion of if and while in the next lesson

```
n = 1000;
D = zeros(1000);    % This is called pre-allocation
D(1) = 1;
D(2) = 2;
tic
for i = 3:n
    D(i) = D(i-1)+D(i-2);
end
toc
```

BE CAREFUL!

For loops are considered the more inefficient type of operation in MATLAB Do not use it unless it is completely necessary. The action of using vectors to perform operations is called VECTORIZATION. You are going to explore more on computational efficiency in the next lesson.