Day 8

- Vectors
- Defining vectors
  - Component by component
  - Using `for` loops
- Other vector commands
- (Exercises)
- (Quiz, 15 minutes)

Vectors

Vectors are lists of numbers that share some common characteristics. For example, a list of rocket positions at consecutive time intervals.

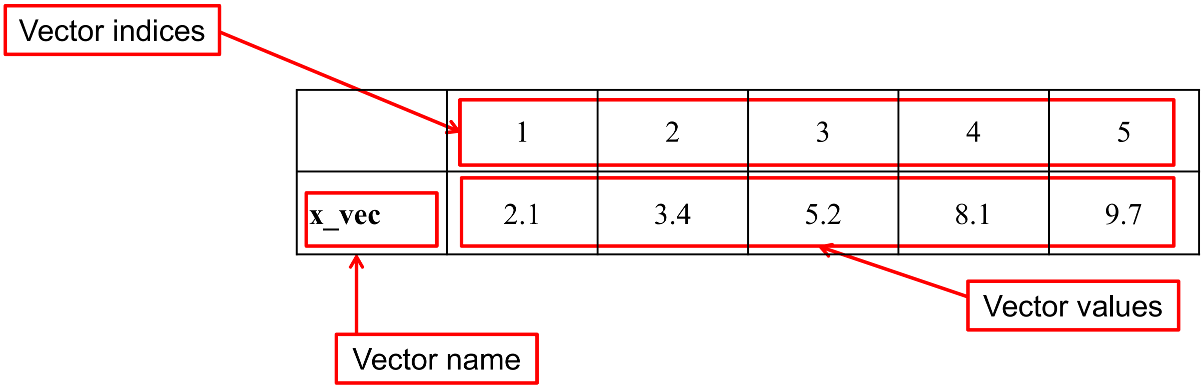
Mathematically, we write vectors as row vectors or column vectors like this:

$$\vec{x} = [2.1 \quad 3.4 \quad 5.2 \quad 8.1 \quad 9.7]$$

$$\vec{x} = \begin{bmatrix} 2.1 \\ 3.4 \\ 5.2 \\ 8.1 \\ 9.7 \end{bmatrix}$$

Vectors

In MATLAB vectors are stored with variable names just like other variables—except that they have a list of values instead of a single value. Each value in the vector is associated with a vector index as illustrated below:



	1	2	3	4	5
x_vec	2.1	3.4	5.2	8.1	9.7

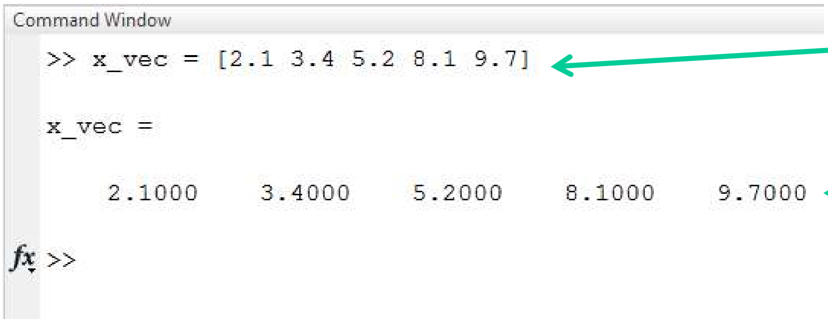
Vector indices

Vector name

Vector values

Defining vectors: Component by component

There are several ways of entering vectors into MATLAB. One way is to enter them component-by-component as follows:



```
>> x_vec = [2.1 3.4 5.2 8.1 9.7]

x_vec =

    2.1000    3.4000    5.2000    8.1000    9.7000

>>
```

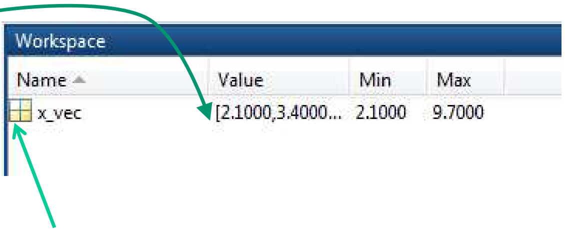
If you type this into the command window

You will get this row vector as a variable called “x_vec”

Defining vectors: Component by component

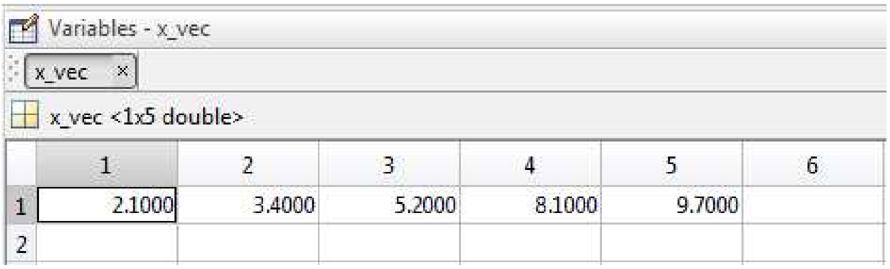
Vectors are variables and show up in the workspace

Note that multiple values are shown under the same variable name



Name	Value	Min	Max
x_vec	[2.1000,3.4000...	2.1000	9.7000

If you double-click on the yellow box, you get a new view of the variable



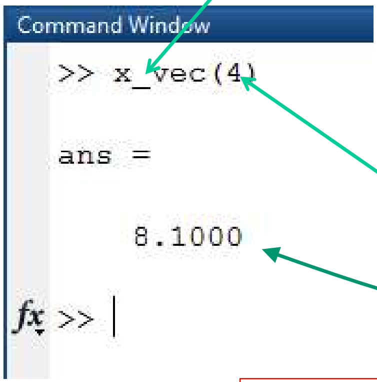
x_vec <1x5 double>

	1	2	3	4	5	6
1	2.1000	3.4000	5.2000	8.1000	9.7000	
2						

Defining vectors: Component by component

If you type the name of the vector with a vector index in parentheses, MATLAB will give you one value out of the vector.

Vector name



```
>> x_vec(4)

ans =

    8.1000

fx >> |
```

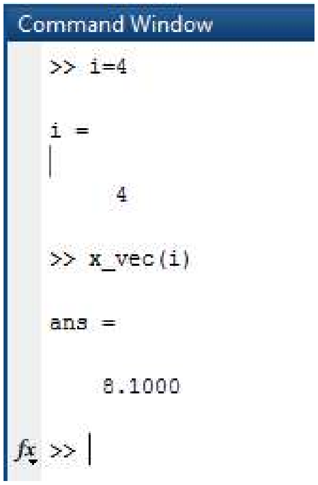
	1	2	3	4	5
x_vec	2.1	3.4	5.2	8.1	9.7

Vector index

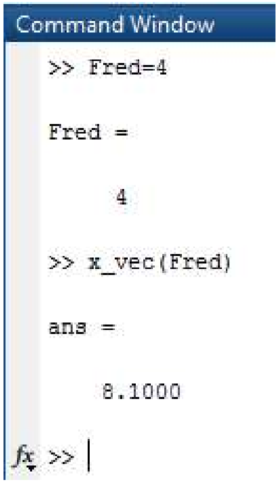
Note that the indices must be positive integers

Defining vectors: Component by component

Vector indices can be variables. Any valid variable name is fine.



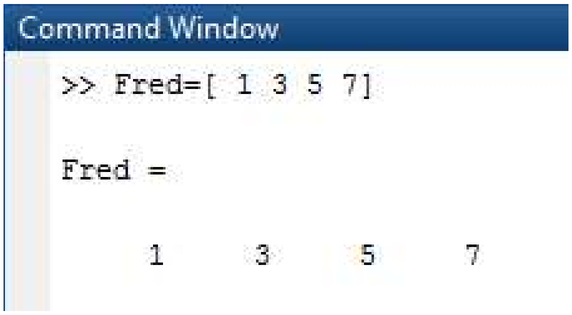
```
>> i=4

i =

     4

>> x_vec(i)

ans =

    8.1000

>> 
```



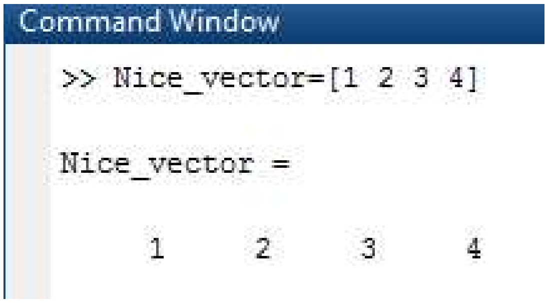
```
>> Fred=4

Fred =

     4

>> x_vec(Fred)

ans =

    8.1000

>> 
```

Defining vectors: Component by Component

Vectors can have any valid variable name



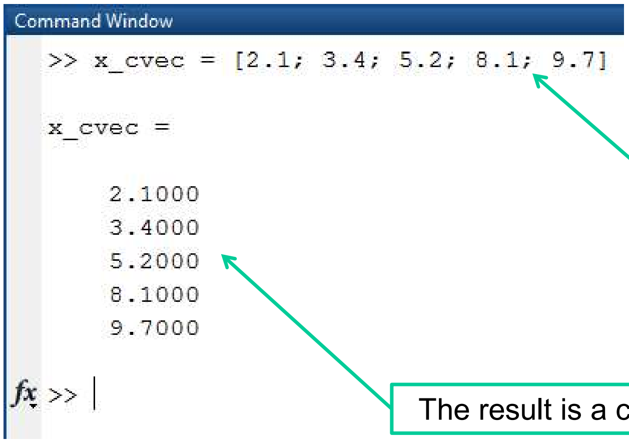
```
>> Fred=[ 1 3 5 7]

Fred =

     1     3     5     7
```



```
>> Nice_vector=[1 2 3 4]

Nice_vector =

     1     2     3     4
```

Defining vectors: Component by Component

If we use semicolons between each component, we can enter vectors as columns rather than rows



```
>> x_cvec = [2.1; 3.4; 5.2; 8.1; 9.7]

x_cvec =

    2.1000
    3.4000
    5.2000
    8.1000
    9.7000

fx >> 
```

Semicolons indicate the end of row. They tell MATLAB to put the next number in the next row

The result is a column vector

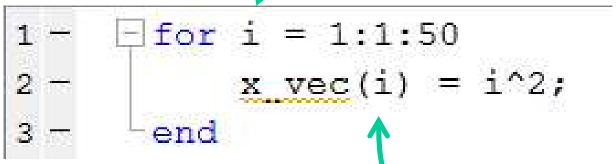
Defining vectors: Using for loops

We can also define a vector using `for` loops.

This method is a good choice when the vector components have a clear mathematical definition.

For example:

$$x_i = i^2 \text{ for } i = 1,2, \dots, 50$$



```
for i = 1:1:50
    x_vec(i) = i^2;
end
```

In this example, the loop variable, `i`, serves as the vector index to place the result in the correct location in the vector, `x_vec`.

Defining vectors: Using for loops

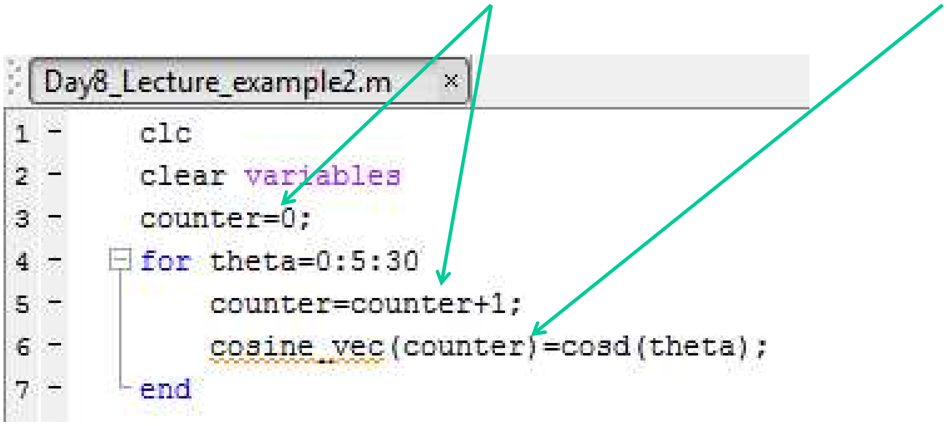
There are at least two ways to define a vector with a `for` loop:

- Method 1: create a *counter* to be the vector *index*
- Method 2: use the *loop variable* as the vector *index*

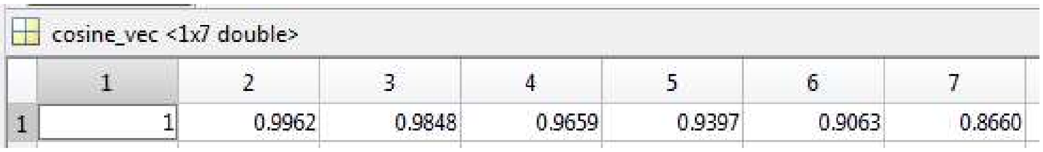
Let us try these two methods to make a vector containing the cosines of angles from 0 to 30 degrees, in 5 degree increments.

Defining vectors: Using for loops

Method 1: create a *counter* to be the vector *index*



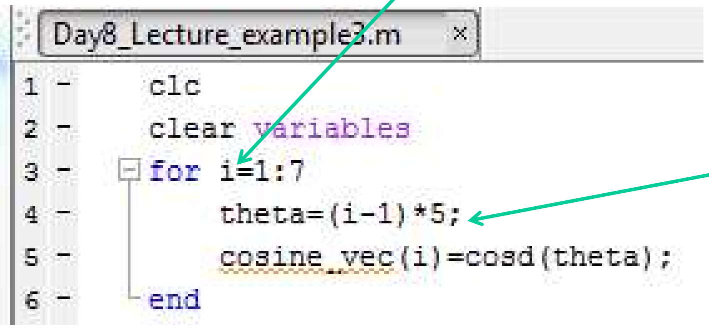
```
clc
clear variables
counter=0;
for theta=0:5:30
    counter=counter+1;
    cosine_vec(counter)=cosd(theta);
end
```


cosine_vec <1x7 double>

	1	2	3	4	5	6	7
1	1	0.9962	0.9848	0.9659	0.9397	0.9063	0.8660

Defining vectors: Using for loops

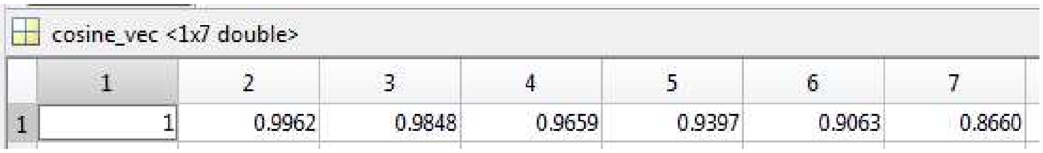
Method 2: use the *loop variable* as the vector *index*



Day8_Lecture_example3.m

```
clc
clear variables
for i=1:7
    theta=(i-1)*5;
    cosine_vec(i)=cosd(theta);
end
```

Notice that the angle (theta) must be calculated from the loop variable (i)



cosine_vec <1x7 double>

	1	2	3	4	5	6	7
1	1	0.9962	0.9848	0.9659	0.9397	0.9063	0.8660

Defining vectors: Using for loops

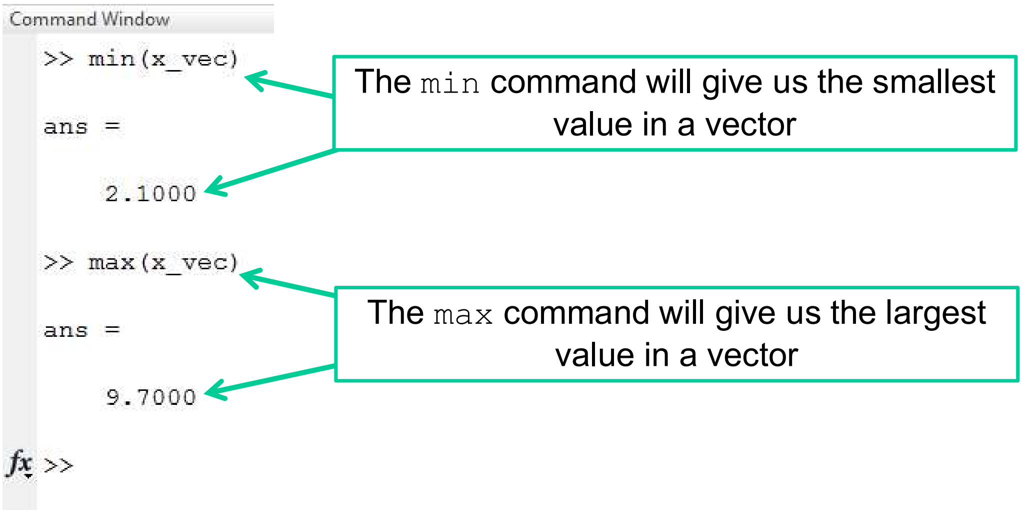
As these two methods illustrate, a given problem can usually be solved any number of different ways.

There is plenty of room to be creative when programming!

Other vector commands: `min`, `max`

MATLAB has commands for finding the minimum or maximum value in a vector.

We can illustrate this using the vector, x_vec, from earlier:



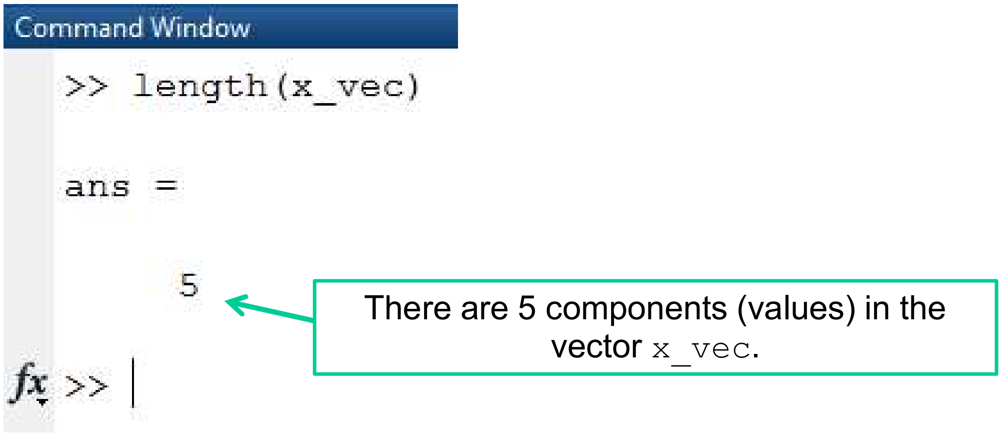
```
>> min(x_vec)

ans =

    2.1000

>> max(x_vec)

ans =

    9.7000

fx >>
```

The `min` command will give us the smallest value in a vector

The `max` command will give us the largest value in a vector

Other vector commands: `length`

We can use the command length to find out how many components a vector has.

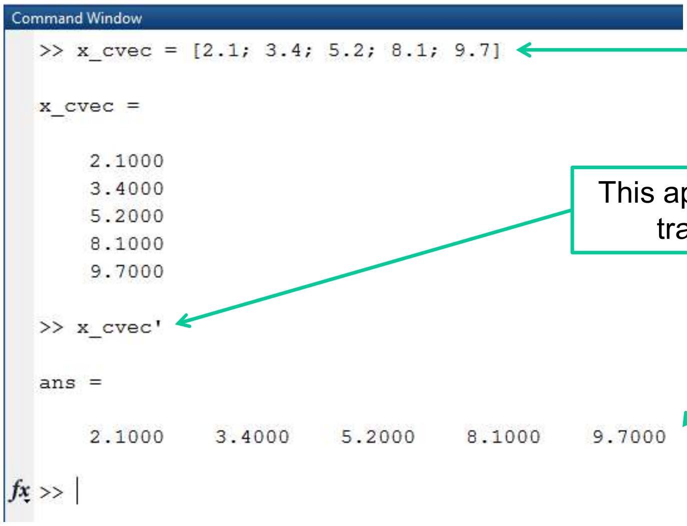


```
>> length(x_vec)

ans =

     5

fx >> |
```

There are 5 components (values) in the vector `x_vec`.

Other vector commands: transpose

The “transpose” of a column vector is a row vector and vice versa. An apostrophe can be used to get the “transpose” of a vector.



```
>> x_cvec = [2.1; 3.4; 5.2; 8.1; 9.7]

x_cvec =

    2.1000
    3.4000
    5.2000
    8.1000
    9.7000

>> x_cvec'

ans =

    2.1000    3.4000    5.2000    8.1000    9.7000

fx >> |
```

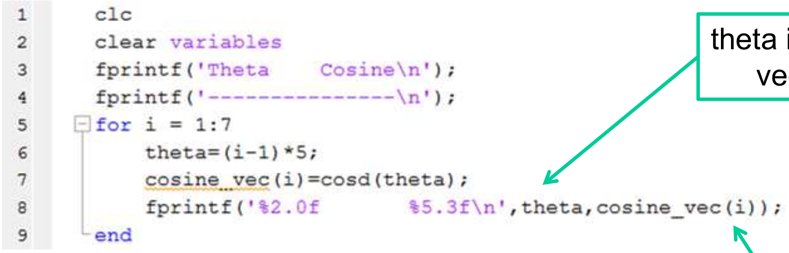
Create a column vector

This apostrophe gives us the transpose of x_cvec

The result is a row vector

Creating tables using vectors

A simple way to print out the values in a vector is to use a for-loop.

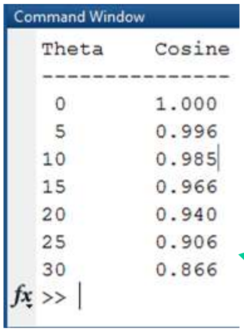


```
clc
clear variables
fprintf('Theta     Cosine\n');
fprintf('---------------\n');
for i = 1:7
    theta=(i-1)*5;
    cosine_vec(i)=cosd(theta);
    fprintf('%2.0f       %5.3f\n',theta,cosine_vec(i));
end
```

theta is just a single value; it is not a vector so no index is needed.

You must specify which entry in the vector you wish to print by using the index *i*.

DO NOT try to print the whole vector at once!



```
Theta     Cosine
---------------
 0        1.000
 5        0.996
10        0.985
15        0.966
20        0.940
25        0.906
30        0.866
fx >> |
```

The resulting printout to the command window