vector functions

create vector	**vec = a:b:c**
create vector with **N** elements from **a** to **b**	**vec = linspace(a:b:N)**
number of elements	**length(x)**
maximum value	**max(x)**
maximum value & index	**[greatest, index] = max(x)**
minimum value	**min(x)**
minimum value & index	**[least, index] = min(x)**
average of all values	**mean(x)**
median of all values	**median(x)**
standard deviation of all values	**std(x)**
sum of all values	**sum(x)**

- **Working with vectors in MATLAB**

MATLAB has two types of vectors (row and column) and offers several methods for creating vectors for use. We will use a naming convention that vector variable names contain only lower case letters.

(1) Creating row vectors in MATLAB.

(a) Enter the following at the Command Line prompt

```
» row_vec1 = [3, 9, -4]
row_vec1 =
     3     9     -4
```

A row vector has been created.

Enter the following at the Command Line prompt

```
» row_vec1(1)
ans =
     3
```

The value of the first element is displayed

Enter the following at the Command Line prompt

```
» row_vec1(3)
ans =
    -4
```

The value of the third element is displayed

(b) Enter the following at the Command Line prompt

```
» row_vec2 = [7  -2  12]
row_vec2 =
     7     -2     12
```

A second row vector has been created. Note: elements separated by spaces rather than commas.

Notes on vectors and vector elements
- The vector name, **`vec_name`**, refers collectively to all elements in the vector.
- Each element has an associated index that uniquely identifies the element.
  - Index counting starts at 1
  - **`vec_name(1)`** refers to the first element, **`vec_name(2)`** the second, etc.
- When creating vectors use square braces **`[ ]`**; when accessing elements use parentheses **`( )`**.
- Elements in a row vector can be separated by either commas or blank spaces.

(c) Enter the following at the Command Line prompt

```
» row_vec3 = 1:4
row_vec3 =
     1     2     3     4
```

A row vector has been created using the colon operator, **`:`**.

(d) Enter the following at the Command Line prompt

```
» row_vec4 = [1:4]
row_vec4 =
     1     2     3     4
```

Another row vector has been created using the colon operator. Square braces, **[ ]**, are optional.

(e) Enter the following at the Command Line prompt

```
» row_vec5 = 1:2:5
row_vec5 =
     1     3     5
```

The colon operator has been used to create a row vector with elements spaced 2 apart.

(f) Enter the following at the Command Line prompt

```
» row_vec6 = [1:-3:-6]
row_vec6 =
     1    -2    -5
```

The colon operator has been used to create a row vector with elements spaced -3 apart.

Notes on colon notation **start_value:increment:end_value**

- Colon notation can be used to create row vectors starting at **start_value** and ending at **end_value**.
- Use of square braces is optional when using colon notation.
- If **increment** is not specified, **increment** equals one
- The **increment** cannot add a value to the vector beyond **end_value**
  ⇒ the last element in the vector is less than or equal to **end_value**
- The **increment** can be negative if **end_value < start_value**

(g) Enter the following at the Command Line prompt

```
» start = 12;
» stop = 24;
» numel = 5;
» row_vec7 = linspace(start,stop,numel)
row_vec7 =
    12    15    18    21    24
```

linspace has been used to create a row vector with ascending values.

(h) Enter the following at the Command Line prompt

```
» start = 12;
» stop = 0;
» numel = 5;
» row_vec8 = linspace(start,stop,numel)
row_vec8 =
    12     9     6     3     0
```

linspace has been used to create a row vector with descending values.

Notes on colon notation and **linspace**

- Variable names can be used rather than numbers to specify **start**, **stop**, and **increment**.
  - Remember, a variable name is a command to go get the value assigned to the variable.
- Colon notation is used to create vectors with evenly spaced elements (**increment**) between **start** and **stop**.
- **linspace** is used to create a vector with a specified number of elements (**numel**) starting with **start** and ending with **end**

(2) Creating column vectors in MATLAB.

(a) Enter the following at the Command Line prompt

```
» col_vec1 = [3; 9; -4]
col_vec1 =
     3
     9
    -4
```

A column vector has been created. Rows are separated by a semi-colon.

Enter the following at the Command Line prompt

```
» col_vec1(1)
ans =
     3
```

The value of the first element is displayed.

Enter the following at the Command Line prompt

```
» col_vec1(3)
ans =
    -4
```

The value of the third element is displayed.

Notes on column vectors

- The vector name, **vec_name**, refers collectively to all elements in the vector.
- Each element has an associated index that uniquely identifies the element.
  - Index counting starts at 1
  - **vec_name(1)** refers to the first element, **vec_name(2)** the second, etc.
- When creating vectors use square braces **[ ]**; when accessing elements use parentheses **( )**.
- Elements in a column vector are separated by semi-colons.

(3) Transposing vectors in MATLAB.

Enter the following at the Command Line prompt

```
» vec1 = [3, 9, -4]
vec1 =
     3     9     -4
```

A row vector is created.

Enter the following at the Command Line prompt (use single quote key)

```
» vec1tr = vec1'
vec1tr =
     3
     9
    -4
```

The row vector is transposed to a column vector.

Enter the following at the Command Line prompt (use single quote key)

```
» vec1trtr = vec1tr'
vec1trtr =
     3     9     -4
```

The column vector is transposed to a row vector.

- We can change from row to column or column to row vectors at will by using the transpose operator, (the single quote key), **'**

(4) Extracting elements from vectors in MATLAB.

Enter the following at the Command Line prompt

```
» xvec = 1:6
xvec =
     1     2     3     4     5     6
»
» xvec_subset = xvec(2:4)
xvec_subset =
     2     3     4
```

- The colon operator can be used to extract a specified range of elements from a vector.

(5) Determining the number of elements in a vector.

Enter the following at the Command Line prompt

```
» length(xvec)
ans =
     6
»
» numel_rv5 = length(row_vec5)
numel_rv5 =
     3
» numel_cv1 = length(col_vec1)
numel_cv1 =
     3
```

- The **length** function returns the number of elements (length) in a row or column vector.

(6) Mathematical functions using vectors in MATLAB.

Enter the following at the Command Line prompt

```
» xvec = linspace(0,pi/2,5)
xvec =
         0    0.3927    0.7854    1.1781    1.5708
»
» sin_xvec = sin(xvec)
sin_xvec =
         0    0.3827    0.7071    0.9239    1.0000
»
» exp_xvec = exp(xvec)
exp_xvec =
    1.0000    1.4810    2.1933    3.2482    4.8105
```

- MATLAB processes a vector in mathematical functions such as **sin** and **cos** element-by-element to produce a vector of the same length where each element in the resulting vector results from performing the specified function on the corresponding element of the argument vector.

(7) Some basic vector operations in MATLAB.

MATLAB is a convenient engineering problem solving tool because it has many "canned" rountines or *functions* that find frequent use in solving problems. For example, think of a vector consisting of grades on an exam. Questions that students frequently ask about the exam are:

(1) what was the average?
(2) what is the median?
(3) what was the maximum score (students rarely ask about the minimum score)
(4) what was the standard deviation?
(5) will the grades be "curved?" (MATLAB is not much use to answer this.)

A variety of vector functions that can answer these and other questions are provided at the start of this workshop.

Enter the following at the Command Line prompt

```
» grades = [97 67 78 88 92 94 84 79 62 95 81 73 91 85 84];
» average = mean(grades)
average =
   83.3333
» mid = median(grades)
mid =
    84
» [high, stnum] = max(grades)
high =
    97
stnum =
     1
» stdev = std(grades)
stdev =
   10.2725
```

Notes on MATLAB functions

- Many MATLAB functions are available to perform different jobs.
- Functions may return more than one value (parameter)
  - **max** returns two values
    the first is the maximum value in the vector
    the second is the location (index) of the maximum value in the vector
- **DO NOT give variables the same name as a MATLAB function!!!!!**

ex: **» sum = 1+2+3**

```
sum =
     6
» total = sum(grades)
??? Index exceeds matrix dimensions.
```

This cryptic error message results because **sum** has been redefined and now refers to the variable rather than the function!!!