MATLAB Workshop 3 - Vectors in MATLAB

Objectives: Learn about vector properties in MATLAB, methods to create row and column vectors, mathematical functions with vectors, and element-by-element vector operations.

MATLAB Features:

vector/matrix variable properties

Property	Comment
`var_name`	user chooses names • name should represent *meaning* of variable ex: use **`radius`** rather than **`x`** or **`s`** for radius • letters, digits (0-9), underscore (_) allowed in name • must start with a letter • case sensitive (**`Radius`** is different than **`radius`**) • can be any length, but first 31 characters must be unique
`value`	all numerical values are • double precision ~ 16 significant figures (not decimal places) • and imaginary has both real and imaginary parts **`a ± bi`** (where **`i = (`**$\sqrt{-1}$**`)`**) both **`a`** and **`b`** are double precision if **`b = 0`**, MATLAB only displays real part, **`a`**
`memory location`	MATLAB assigns - you do not need to worry about this

order of precedence rules for vector/matrix arithmetic operators

Order of Precedence	Symbol	Meaning
1	**`( )`**	group together
2	**`'`**	transpose (exchange rows and columns)
	`.^`	raise each element to indicated power
	`^`	multiply matrix by itself indicated number of times
3	**`.*`**	element-by-element multiplication
	`./`	element-by-element right division
	`.\`	element-by-element left division
	`*`	multiply two matrices
	`/`	matrix right division
	`\`	matrix left division
4	**`+`**	addition
	`-`	subtraction

colon (`:`) operator - a special vector/matrix operator

Notation	Action	Result
`a:c`	creates vector with elements **`1`** apart	**`[a a+1 a+2 ... c]`**
`a:b:c`	creates vector with elements **`b`** apart	**`[a a+b a+2b ... (≤c)]`**

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!!!

- **Arithmetic operations with vectors in MATLAB**

MATLAB has arithmetic operations that process vectors either element-by-element (term-by-term) or by matrix algebra rules. For example, consider the function

$$x = \frac{at^2}{2} + v_0 t + x_0$$

which describes the displacement of an object under constant acceleration, a, starting at position x_0 with initial velocity v_0. We might like to display a graph of x vs t. In order to do this, we would need a vector of t-points (the independent variable) and a corresponding vector of x-points (the dependent variable). Generally, we will create an evenly spaced set of independent variable points and then calculate the corresponding dependent variable points.

(8) Creating a set of (*x,y*) data points in MATLAB.

We will generically say (*x,y*) data points for (*independent, dependent*) data points. For the displacement problem, the (*x,y*) data set is really a (*time,displacement*) data set. **Why?**

Enter the following at the Command Line prompt

```
» diary  wkshp3_ac7
»
» % Solution to Workshop 3, Activity 7
» % Your Name(s)
» % Class (e.g. Engr 0012, TH 10:00, Instructor Name)
» % Today's Date
» % Your e-mail address
»
» % set problem parameters
»    accel = 1.0;    % acceleration, m/s/s
»    initvel = 0.0;  % initial velocity, m/s
»    initpos = 0.0;  % initial position, m
»    time = 0:5      % time range, s
time =
     0     1     2     3     4     5
»
» % compute corresponding displacement, m
»    displacement = accel*time.^2/2.0 + ...
initvel*time + initpos
displacement =
         0    0.5000    2.0000    4.5000    8.0000   12.5000
```

Note that the element-by-element exponentiation operator, **.^**, was used to square the **time**. This told MATLAB to process the vector element-by-element so that a **displacement** element was created for each **time** element. Also note that, if a line is extending off the screen, it can be broken and continued on the next line by using **...** Using this feature makes your command window easier to read and avoids needing to use the scroll bars to see everything that you have written.

See what happens if you forget to use the element-by-element exponentiation operator, **.^**, and use the matrix exponentiation operator, **^**, instead. (The difference is a period or dot before the hat). Use the up arrow to move up to the **displacement =** command,

change the operator by deleting the period, and press enter. Then use the up arrow to recall the rest of the equation and press enter.

```
»
»   displacement = accel*time^2/2.0 + ...
initvel*time + initpos
??? Error using ==> ^
Matrix must be square.
»
» diary off
```

MATLAB tried to use the rules of matrix algebra to multiply time by itself. However, a vector cannot be multiplied by itself under those rules, so an error message was issued.

(9) Creating another set of (*x*,*y*) data points in MATLAB.

Let's repeat the previous activity for the equation

$$d(\alpha) = 0.5 + 0.4e^{(-\alpha/2\pi)}\sin(\alpha)$$

which describes the displacement of an irregular cam in cm about its center as a function of angular rotation, α, in radians. Cam shafts are used in engines to control the stroke of the piston. We will look at the displacement through one revolution.

Enter the following at the Command Line prompt

```
» diary  wkshp3_ac8
»
» % Solution to Workshop 3, Activity 8
» % Your Name(s)
» % Class (e.g. Engr 0012, TH 10:00, Instructor Name)
» % Today's Date
» % Your e-mail address
»
» % set problem parameters
»   alpha = linspace(0,2*pi,20)      % alpha range, radians
alpha =
  Columns 1 through 7
       0    0.3307    0.6614    0.9921    1.3228    1.6535    1.9842
  Columns 8 through 14
  2.3149    2.6456    2.9762    3.3069    3.6376    3.9683    4.2990
  Columns 15 through 20
  4.6297    4.9604    5.2911    5.6218    5.9525    6.2832
»
» % compute corresponding displacement, cm
»   displacement = 0.5 + 0.4*exp(-alpha/2/pi).*sin(alpha)
displacement =
  Columns 1 through 7
  0.5000    0.6232    0.7211    0.7860    0.8141    0.8064    0.7671
  Columns 8 through 14
  0.7036    0.6250    0.5410    0.4611    0.3933    0.3435    0.3152
  Columns 15 through 20
  0.3092    0.3239    0.3557    0.3996    0.4496    0.5000
```

Note how MATLAB reports the contents of a (row) vector if it contains more elements than can be displayed on a single line. Also note the use of **`linspace`** rather than the colon

operator to create values for the angle vector. **Why was `linspace` used?** Finally, note that multiplication by scalars does not require the element-by-element operator while multiplication of the two vectors created by the **`exp`** and **`sin`** functions does. A function with a vector as an argument creates a new vector with values obtained from element-by-element processing of the function.

What happens if you forget to use the element-by-element exponentiation operator, **`.^`**, and use the matrix exponentiation operator, **`^`**, instead?

```
»
»   displacement = 0.5 + 0.4*exp(-alpha/2/pi)*sin(alpha)
??? Error using ==> *
Inner matrix dimensions must agree.
»
» diary off
```

MATLAB tried to use the rules of matrix algebra to multiply the two vectors. However, two vectors cannot be multiplied under those rules, so an (different) error message was issued.

Exercises: Perform the following operations using array arithmetic where appropriate.

1. *Equation of a line*. The equation of a line is given by $y = mx + b$ where m is the slope (a scalar) and b is the intercept (also a scalar).

   a. Compute the corresponding y-coordinates for the following x-coordinates if m = 3 and b = -1
   ```
   x = [1, 2, 3, 5, 8, 13, 21]
   ```

   b. Compute the corresponding x-coordinates for the following y-coordinates if m = 5 and b = 2.3
   ```
   y = [0, 1, 1, 2, 3, 5, 8]
   ```

   c. Compute the corresponding y-coordinates for the following x-coordinates if m = π/2 and b = 2/π
   ```
   x = linspace(0,30,8)
   ```

   d. Given the equation $d = 3\sin(t) + t/2$, compute the corresponding d-coordinates for
   ```
   t = 0:10
   ```

2. *Vector multiplication, division, exponentiation*. Create a vector, **g**, with 10 evenly spaced elements starting at 1 and ending at 10. Compute the following with vector operations:

   a. $h = g\cos(g)$

   b. $z = \dfrac{g-1}{h+1}$

   c. $s = \dfrac{\cos(g^2)}{hg}$ (try this by squaring **g** and then by multiplying **g** times **g**)

d. Given the equation $q = 5\sin(\pi t / 2.5)e^{-t/2}$, compute the corresponding q-coordinates for

```
t = 0:100
```

3. *Parametric equation for a circle*. The parametric equation for a circle is $x = r\cos(\theta)$ and $y = r\sin(\theta)$ where r is the radius and θ is the angle of rotation counter-clockwise from the positive x-axis. Defined this way, x and y satisfy the equation $x^2 + y^2 = r^2$. Show this using MATLAB. Use `linspace` to create an angle vector, **theta**, with values (0, π/3, 2π/3, π, 4π/3, 5π/3, 2π). Compute the corresponding **x**- and **y**-vectors for r = 5. Show that the **x**- and **y**-vectors satisfy the equation of a circle.

Recap: You should have learned

- How to declare row and column vectors in MATLAB
- How to declare a vector with evenly spaced elements (two methods)
- How to transpose a vector
- How to extract elements from a vector
- Arithmetic operations between a scalar and a vector
- Arithmetic operations between two vectors
- Simple function operations with a vector
- Arithmetic operations between functions of vectors