MATLAB
Variables & Data Types

Week 2

Loosely follows Chapter 2

Variables

- Used to store information
- Can hold numerical values or text values and you don’t need to declare them
- Can be recalled later
- Always use informative names
- Standard conventions
  - speedOfLightKPH
  - speed_of_light_kph
  - SPEED_OF_LIGHT_KPH

Variable Naming Rules

- Must begin with a letter
- Must only contain letters, numbers, and underscores
- Cannot contain spaces or punctuation marks
- Names exceeding 63 characters are truncated
  - Please don’t create variables 63 characters long (30 is a good limit)
- Case sensitive but try **NOT** to use two variables of the same name
  - X and x is acceptable
  - BALANCE and balance is not acceptable

The Workspace

- Contains current variables
- I refer to this as the “environment”
- who displays a list of current variables
- whos displays current variables, data types, and sizes
- Workspace in the UI also displays current variables

Data Types

- MATLAB has 15 different data types
  - str = 'Hello World!'
  - n = 2345
  - d = double(n)
  - un = uint32(789.50)
  - rn = 5678.92347
  - c = int32(rn)

```
str = Hello World!
n =  2345
d =  2345
un = 790
rn = 5678.9
c =  5679
```

- All operations are performed with double precision by default

Variables as arrays

All variables in MATLAB are arrays

Scalar	Vector	Matrix
1x1	nx1 or 1xn	nxm

Vectors and Matrices are mathematical objects, Arrays are lists or tables.

Scalars

- Scalars are simply 1x1 arrays
- They contain a single value, such as the following:

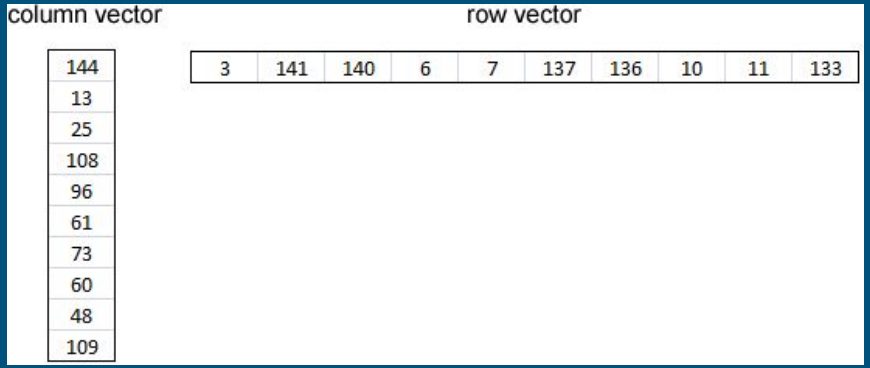
radius = 6

height = 12

width = 2

Vectors

- A vector is a collection of values represented in a 1-dimensional array
- A vector can be n_x1 or 1_xn
- Rows are separated by semicolons (Column Vector)
  - heights = [144; 13; 25; 108; 96; 61; 73; 60; 48; 109]
- Columns are separated by commas or spaces (Row Vector)
  - heights = [3, 141, 140, 6, 7, 137, 136, 10, 11, 133]



Matrices

- A matrix is a collection of values represented as a 2-dimensional array
- Defining a matrix is done so with both
  commas (for columns) and
  semicolons (for rows)

```
matrix = [ 3.0, 1.8, 3.6;
           4.6, -2.0, 21.3;
           0.0, -6.1, 12.8;
           2.3, 0.3, -6.1 ]
```

ROWS \ COLUMNS	1	2	3
1	3.0	1.8	3.6
2	4.6	-2.0	21.3
3	0.0	-6.1	12.8
4	2.3	0.3	-6.1

Scalar & Array Operations

Operation	Algebraic Syntax	MATLAB Syntax
Addition	a + b	`a + b`
Subtraction	a - b	`a - b`
Multiplication	a × b	`a .* b`
Division	a ÷ b	`a ./ b`
Exponentiation	a^b	`a .^ b`

Hierarchy of Operations

- Remember good old PEMDAS?
- **IMPORTANT:** calculations are performed left to right after PEMDAS

Example breakdown

c = 2+3^2+1/(1+2) → c = 2*3^2+1/**3**
c = 2+3^2+1/(1+2) → c = 2***9**+1/3
c = 2*3^2+1/(1+2) → c = **18**+1/3 (left to right)
c = 2*3^2+1/(1+2) → c = 18+**0.33333**
c = 2*3^2+1/(1+2) → c = **18.33333**

Array Operations

- Matrix operations can be performed on an array
- To perform matrix calculations, use the standard operators (+, -, *, /, ^)
- To perform operations to each discrete element, use the dot/element-wise operator. (+, -, .*, ./, .^)
- Use the dot-operator anytime unless you intend to perform matrix math.

Matrix Operations

Operation	Algebraic Syntax	MATLAB Syntax
Addition	A + B	`A + B`
Subtraction	A - B	`A - B`
Multiplication	A x B	`A * B`
Division	A ÷ B	`A / B`
Exponentiation	A^b	`A ^ b`

Array vs Matrix Arithmetic Example

- Enter the following:
  - >> x = [2, 1; 3, 4]
  - >> y = [5, 6; 7, 8]
  - >> zMat = x * y
  - >> zDot = x .* y

zMat should contain [17, 20; 43, 50] due to **matrix** multiplication
zDot should contain [10, 6; 21, 32] due to **array** multiplication

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} \times \begin{pmatrix} E & F \\ G & H \end{pmatrix} = \begin{pmatrix} AE+BG & AF+BH \\ CE+DG & CF+DH \end{pmatrix}$$

Explicitly Defining & Assigning Arrays

- An array can be defined by typing in a list of numbers enclosed in square brackets.
  - Commas or spaces separate numbers or columns
    - A = [12, 18, -3] OR A = [12 18 -3]

      A =

      12 18 -3
  - Semicolons separate rows
    - B = [2, 5, 2; 1, 1, 2; 0, -2, 6]

      B =

      2 5 2
      1 1 2
      0 -2 6

Dynamically Defining & Assigning Arrays

- Colon notation can be used to define evenly spaced vectors with first:last
  - H = 1:6

    H =

    1 2 3 4 5 6

- The default increment is 1, you can specify with first:increment:last
  - I = 1:2:11

    I =

    1 3 5 7 9 11

The *linspace()* and *logspace()* Functions

- *linspace(start, end, n)*
- Creates a finite linearly spaced set of n values between start and end
- Useful if you know the start, end, and number of values you want
  - linspace(1,10,4)
    ans =
    1 4 7 10

- *logspace(a,b,n)*
- Creates n points between 10^a and 10^b.
- Useful in signal processing

Array Inception

- You can define an array with other arrays
  - C = [A; B]

    C =

12	18	-3
2	5	2
1	1	2
0	-2	6

  - D = [C, C]

    D =

12	18	-3	12	18	-3
2	5	2	2	5	2
1	1	2	1	1	2
0	-2	6	0	-2	6

Recall:

A =

12	18	-3

B =

2	5	2
1	1	2
0	-2	6

Creating Arrays of Zeros & Ones

- For zeros use the zeros(numRows, numCols) function
  - E = zeros(3,5)

```
E =
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
```

- For ones use the ones(numRows, numCols) function
  - F = ones(2,3)

```
F =
     1     1     1
     1     1     1
```

NOTE: Placing a single number inside either function will return an nxn array

Accessing Values in an Array or Vector

- Referred to as a subscript
- An index is a number used to identify elements in an array
- Add parenthesis after your array name with row and col index (row,col)
  - G = [1, 2, 3; 4, 5, 6; 7, 8, 9]

    G =

1	2	3
4	5	6
7	8	9

  - G(2,1) ← Retrieves the element from row 2, col 1

    ans = 4

Changing/Setting Values in an Array

- Just as with retrieval, you can change values in an array with the same notation.
  - A = ones(2) ← Create a 2x2 matrix of ones
  - A(2, 1) = 8 ← Sets the value of row two, col one to 8

```
A =
     1     1
     8     1
```

- You can extend an array by defining a new element
  - A(2, 3) = 10

```
A =
     1     1     0
     8     1    10
```

NOTE: When defining a new element, notice any undefined values are filled with zeros

Retrieving an Entire Vector

- With retrieval notation, an entire row or column can be represented as a colon in place of an index value.
  - G = [1, 2, 3; 4, 5, 6; 7, 8, 9]

```
G =
     1     2     3
     4     5     6
     7     8     9
```

```
G(:,1)
ans =
     1
     4
     7
```

```
G(:,3)
ans =
     3
     6
     9
```

```
G(2,:)
ans =
     4     5     6
```

More Extraction Methods with Colon

- The colon operator can also be used to extract a range of rows or columns
  - G = [1, 2, 3; 4, 5, 6; 7, 8, 9]

```
G =
     1     2     3
     4     5     6
     7     8     9
```

```
G(2:3,:)
ans =
     4     5     6
     7     8     9
```

```
G(1,2:3)
ans =
     2     3
```

Manipulating Arrays & Vectors

- The transpose operator, a single apostrophe ' changes all of an array's rows to columns and columns to rows
  - J = [1, 3, 7]

```
J =
     1     3     7
```

  - J'

```
J =
     1
     3
     7
```

  - K = [1, 2, 3; 4, 5, 6; 7, 8, 9]

```
K =
     1     2     3
     4     5     6
     7     8     9
```

  - K'

```
J =
     1     4     7
     2     5     8
     3     6     9
```

Manipulating Arrays & Vectors (cont.)

- The functions fliplr() and flipud() flip an array left to right (lr) and upside-down (ud) respectively.
  - G = [1, 2, 3; 4, 5, 6; 7, 8, 9]

```
G =
     1     2     3
     4     5     6
     7     8     9
```

```
fliplr(G)
ans =
     3     2     1
     6     5     4
     9     8     7
```

```
flipud(G)
ans =
     7     8     9
     4     5     6
     1     2     3
```

Matrix Manipulation Exercises

- Create the following matrix using colon notation:

W =

1	2	3	4	5
10	12	14	16	18
6	5	4	3	2

- All three rows are evenly spaced
  - The first row ranges from 1 to 5 in increments of 1 (1:5 OR 1:1:5)
  - The second row ranges from 10 to 18 in increments of 2 (10:2:18)
  - The third row ranges from 6 to 2 in increments of -1 (6:-1:2)
  - Put them all together:
    - W = [1:5; 10:2:18; 6:-1:2]

Matrix Manipulation Exercises

- Create the following matrix using colon notation:

X =

1.2	2.3	3.4	4.5	5.6
1.9	3.8	5.7	7.6	9.5
0	-3	-6	-9	-12

- Transpose this matrix and assign it to variable Y
  - Y = x'

- Extract the 2nd row from Y and assign it to variable Z
  - Z = Y(2, :)