Day 9

- (Concept Question)
- Matrices
- Defining matrices
  - Component by component
  - Single `for` loops
  - Nested `for` loops
- Printing matrices
- Reading spreadsheet data as matrices
- Reminder: precedence of operators
- (Exercises)

Matrices

Recall: Column vectors have just one column.

$$\overrightarrow{theta} = \begin{bmatrix} 0 \\ 10 \\ 20 \\ \vdots \\ 90 \end{bmatrix} \quad \overrightarrow{sine} = \begin{bmatrix} 0.00 \\ 0.17 \\ 0.34 \\ \vdots \\ 1.00 \end{bmatrix} \quad \overrightarrow{cosine} = \begin{bmatrix} 1.00 \\ 0.98 \\ 0.94 \\ \vdots \\ 0.00 \end{bmatrix}$$

Matrices

Matrices have many rows and columns.

$$\vec{M} = \begin{bmatrix} 0 & 0.00 & 1.00 \\ 10 & 0.17 & 0.98 \\ 20 & 0.34 & 0.94 \\ \vdots & \vdots & \vdots \\ 90 & 1.00 & 0.00 \end{bmatrix}$$

This matrix is arranged so that the three columns are angle, sine, and cosine.

Matrices

You choose how your matrix is arranged. This matrix is arranged so that the three rows are angle, sine, and cosine.

$$\vec{M} = \begin{bmatrix} 0 & 10 & 20 & \dots & 90 \\ 0.00 & 0.17 & 0.34 & \dots & 1.00 \\ 1.00 & 0.98 & 0.94 & \dots & 0.00 \end{bmatrix}$$

Matrices

We use a standard way of identifying the entries in a matrix.

$$\vec{M} = \begin{bmatrix} 0 & 10 & 20 & \dots & 90 \\ 0.00 & 0.17 & 0.34 & \dots & 1.00 \\ 1.00 & 0.98 & 0.94 & \dots & 0.00 \end{bmatrix}$$

$M_{23} = 0.34$

Row 2

Column 3

Matrices

Notice that the order of the subscripts matters:
$M_{23} \neq M_{32}$

$$\vec{M} = \begin{bmatrix} 0 & 10 & 20 & \dots & 90 \\ 0.00 & 0.17 & 0.34 & \dots & 1.00 \\ 1.00 & 0.98 & 0.94 & \dots & 0.00 \end{bmatrix}$$

$M_{32} = 0.98$

$M_{23} = 0.34$

Defining matrices: component by component

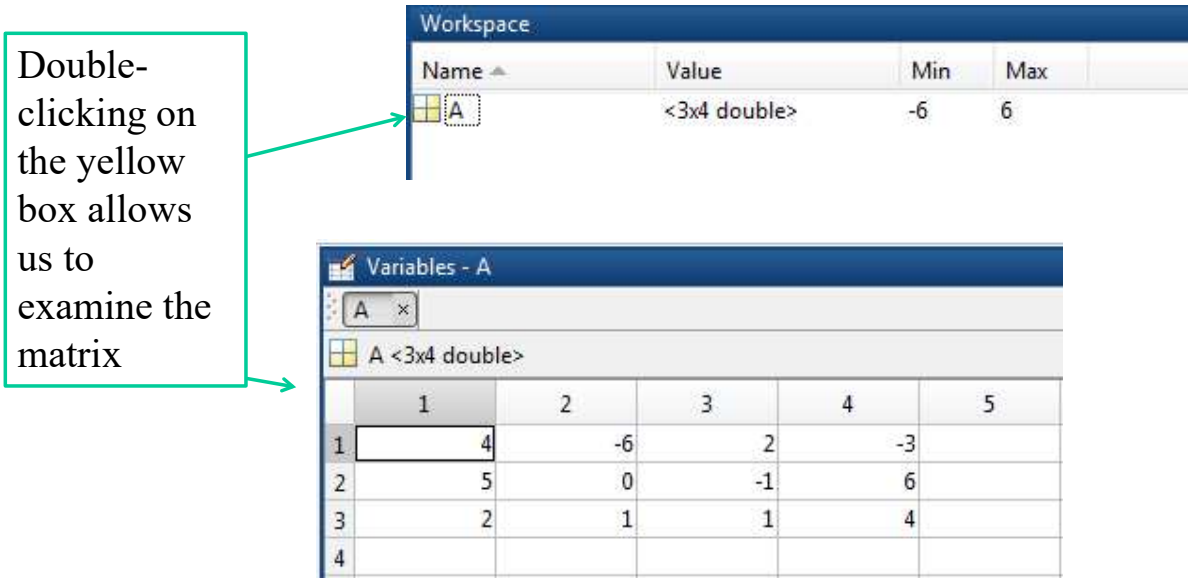
We can define a matrix by typing it into Matlab

```
Command Window
>> A = [4 -6 2 -3; 5 0 -1 6; 2 1 1 4]

A =

     4    -6     2    -3
     5     0    -1     6
     2     1     1     4
```


Defining matrices: component by component

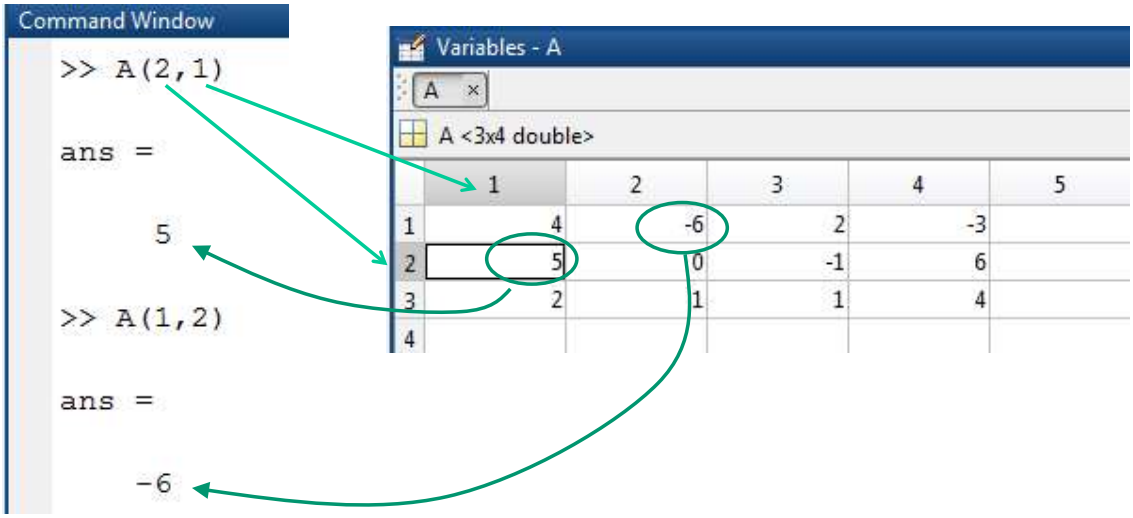
The matrix variable appears in the Workspace.

Double-clicking on the yellow box allows us to examine the matrix



Defining matrices: component by component

We can also refer to a single value in the matrix



Defining matrices: single `for` loops

We can use for loops to create matrices

```
Day9_inClassExample2.m
1 -   clc
2 -   clear variables
3 -   column=0;
4 -   for theta=0:10:90
5 -       column=column+1;
6 -       M(1,column)=theta;
7 -       M(2,column)=sind(theta);
8 -       M(3,column)=cosd(theta);
9 -   end
```

This code creates a matrix with angles in the first row, sines in the second row, cosines in the third row.

M <3x37 double>

	1	2	3	4	5	6	7	8	9	10
1	0	10	20	30	40	50	60	70	80	90
2	0	0.1736	0.3420	0.5000	0.6428	0.7660	0.8660	0.9397	0.9848	1
3	1	0.9848	0.9397	0.8660	0.7660	0.6428	0.5000	0.3420	0.1736	0
4										

Defining matrices: single `for` loops

Day9_inClassExample2.m

```
1 -    clc
2 -    clear variables
3 -    column=0;
4 -    for theta=0:10:90
5 -        column=column+1;
6 -        M(1,column)=theta;
7 -        M(2,column)=sind(theta);
8 -        M(3,column)=cosd(theta);
9 -    end
```

row 1: angles

row 2: sines of angles

row 3: cosines of angles

Defining matrices: single `for` loops

Day9_inClassExample2.m

```
1 -    clc
2 -    clear variables
3 -    column=0;
4 -    for theta=0:10:90
5 -        column=column+1;
6 -        M(1,column)=theta;
7 -        M(2,column)=sind(theta);
8 -        M(3,column)=cosd(theta);
9 -    end
```

This script uses a recursive assignment for the variable "column" to put the values in the correct columns of the matrix

Defining matrices: nested `for` loops

Some matrices have a structure that needs a more complicated loop structure in order to generate them:

$$C = \begin{bmatrix} 2 & 3 & 4 \\ 3 & 4 & 5 \\ 4 & 5 & 6 \\ 5 & 6 & 7 \end{bmatrix}$$

Notice that $C_{ij} = \mathrm{i} + \mathrm{j}$

Defining matrices: nested `for` loops

We can use nested for loops (one for loop inside another) to generate this type of matrix

Day9_inClassExample3.m

```
1 -    clc
2 -    clear variables
3 -    for row=1:4
4 -        for column=1:3
5 -            C(row,column)=row+column;
6 -        end
7 -    end
```

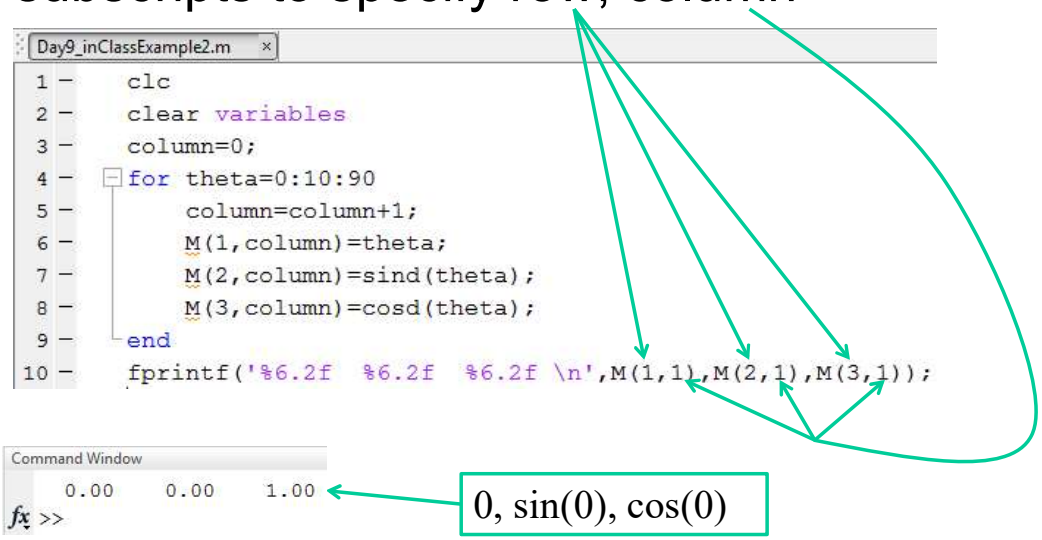
C <4x3 double>

	1	2	3
1	2	3	4
2	3	4	5
3	4	5	6
4	5	6	7

The inner "column" loop runs completely for each value of row

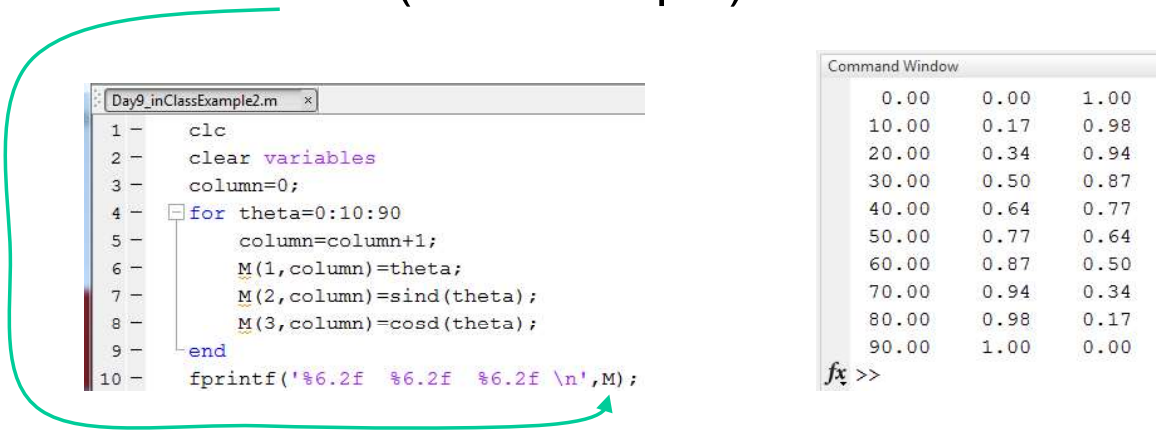
Printing matrices

To print a specific value from a matrix, use subscripts to specify row, column



Printing matrices

To print the entire matrix at once, just use the matrix name (no subscripts)



Notice that Matlab transposed the matrix (traded rows and columns) when it printed!

Printing matrices

To print out the entire matrix as you created it, you must print the transpose.

Day9_inClassExample2.m

```
1 -    clc
2 -    clear variables
3 -    column=0;
4 -    for theta=0:10:90
5 -        column=column+1;
6 -        M(1,column)=theta;
7 -        M(2,column)=sind(theta);
8 -        M(3,column)=cosd(theta);
9 -    end
10 -   fprintf('%6.2f %6.2f %6.2f %6.2f %6.2f %6.2f %6.2f %6.2f %6.2f %6.2f\n',M');
```

Command Window

```
    0.00   10.00   20.00   30.00   40.00   50.00   60.00   70.00   80.00   90.00
    0.00    0.17    0.34    0.50    0.64    0.77    0.87    0.94    0.98    1.00
    1.00    0.98    0.94    0.87    0.77    0.64    0.50    0.34    0.17    0.00
fx >>
```

Now we see 3 rows and 10 columns

Printing matrices

If you accidentally print the ENTIRE matrix inside the for loop you will get a LOT of output

Day9_inClassExample2.m

```
1 -    clc
2 -    clear variables
3 -    column=0;
4 -    for theta=0:10:90
5 -        column=column+1;
6 -        M(1,column)=theta;
7 -        M(2,column)=sind(theta);
8 -        M(3,column)=cosd(theta);
9 -        fprintf('%6.2f %6.2f %6.2f\n',M);
10 -   end
```

Command Window

```
    0.00    0.00    1.00
    0.00    0.00    1.00
   10.00    0.17    0.98
    0.00    0.00    1.00
   10.00    0.17    0.98
   20.00    0.34    0.94
    0.00    0.00    1.00
   10.00    0.17    0.98
   20.00    0.34    0.94
   30.00    0.50    0.87
    0.00    0.00    1.00
   10.00    0.17    0.98
   20.00    0.34    0.94
   30.00    0.50    0.87
   40.00    0.64    0.77
    0.00    0.00    1.00
   10.00    0.17    0.98
   20.00    0.34    0.94
```

Printing matrices

If you don't give enough entries in the format you will get strange-looking results

```
Day9_inClassExample2.m
 1 -    clc
 2 -    clear variables
 3 -    column=0;
 4 -    for theta=0:10:90
 5 -        column=column+1;
 6 -        M(1,column)=theta;
 7 -        M(2,column)=sind(theta);
 8 -        M(3,column)=cosd(theta);
 9 -    end
10 -    fprintf('%6.2f %6.2f \n',M);
```

only 2 places for numbers, but 3 rows in the matrix

```
Command Window
    0.00    0.00
    1.00   10.00
    0.17    0.98
   20.00    0.34
    0.94   30.00
    0.50    0.87
   40.00    0.64
    0.77   50.00
    0.77    0.64
   60.00    0.87
    0.50   70.00
    0.94    0.34
   80.00    0.98
    0.17   90.00
    1.00    0.00
fx >>
```


Reading spreadsheet data as matrices

Matlab lets you read in Excel spreadsheets. The data in the spreadsheets is stored as a matrix inside Matlab.

```
Day9_inClassExample4.m
1 -    clc
2 -    clear variables
3 -    thermocouple_data=xlsread('data');
```

Name of the matrix

Name of the spreadsheet. Don't forget the quotes.

thermocouple_data <200x2 double>

	1	2	3
1	0.0107	35.4524	
2	0.0202	37.3325	
3	0.0311	35.5455	
4	0.0405	36.5118	
5	0.0507	41.4824	
6	0.0617	42.1343	
7	0.0711	43.6220	
8	0.0815	42.5057	
9	0.0921	45.1604	
10	0.1015	44.5312	
11	0.1123	49.1657	
12	0.1222	46.9676	
13	0.1321	47.3782	
14	0.1428	52.0346	
15	0.1525	53.3840	
16	0.1627	54.3223	
17	0.1727	55.0224	
18	0.1829	56.5516	

Reading spreadsheet data as matrices

The ‘size’ command lets us determine how many rows and columns a matrix has.

Day9_inClassExample4.m

```
1 -    clc
2 -    clear variables
3 -    thermocouple_data=xlsread('data');
4 -    [n_rows, n_cols]=size(thermocouple_data);
```

Workspace

Name	Value
n_cols	2
n_rows	200
thermocouple_data	<200x2 double>

any valid variable names for the two answers

This will be important when we are automating our work with a script, and we don’t know how much data will be in each spreadsheet

Reminder:
Matlab uses standard precedence of operators

Standard precedence of operators:

1. ( )
2. ^
3. * /
4. + -

Command Window

New to MATLAB? See resources for Getting Started.

```
>> 6+4*3^2

ans =

    42

fx >>
```