Basic MATLAB continued; Using functions

Variable names

valid variable names	invalid variable names	reason invalid
`x`	`$`	• does not begin with a letter • `$` is not allowed in variable names
`x6`	`6x`	• does not begin with a letter
`lastValue`	`if`	• `if` is a keyword
`pi_over_2`	`pi/2`	• / is not allowed in variable names

Advice on choosing variable names

- use short, meaningful names
  - a name that conveys the purpose of the variable is often useful for others who need need to read your code, e.g., use

    `massEarth` instead of **`mE`**
    `massSun` instead of **`mS`**

  - exceptions to the rule:
    - if you are solving a problem that contains variable names, you should try to use the same names, e.g., in physics the following would likely be common:

      `g`, **`c`**, **`v0`**, **`h`**, **`hBar`**

Advice on choosing variable names

- use lowerCamelCase for most variable names, e.g., use

  `thetaRad` instead of **`thetarad`**

- avoid long names, e.g., use

  `filteredData` instead of **`measurementsFilteredToRemoveOutliers`**

Advice on choosing variable names

- be careful when using **`i`** and **`j`** as variable names
  - **`i`** and **`j`** are often used as loop variables (see Week 06)
  - in MATLAB **`i`** and **`j`** are actually names of functions that return the square root of -1

More on variable assignment

- remember that the statement:

```
z = 1 + 2
```

means:

1. evaluate the expression on the right-hand side of **=**
2. store the result in the variable on the left-hand size of **=**

More on variable assignment

- what is the result of the following assignment statements?

```
z = 1 + 2;
y = z;
y = 4;
```

- is the value of **z** **3** or **4**?

More on variable assignment

- the statement:

```
y = z;
```

means:

1. evaluate the expression on the right-hand side of **=**
2. store the result in the variable on the left-hand size of **=**

Operator precedence

- all operators in MATLAB follow a set of precedence rules ("order of operations")

operator	name	precedence
`( )`	parentheses	highest
`^`	exponentiation	
`-`	negation	
`*, /, \`	multiplication and division	
`+, -`	addition and subtraction	lowest

Logical expressions and operators

- the textbook introduces relational and logical expressions and operators in Chapter 1
  - these are expressions and operators involving the values **`true`** and **`false`**
- the relational operators are:

operator	name
>	greater than
>=	greater than or equal to
<	less than
<=	less than or equal to
==	equal to
~=	not equal to

Logical expressions and operators

- the logical operators are:

operator	name
&&	and
||	or
~	not

- logical expressions and operators are not useful us until Week 05

Functions

- most MATLAB is provided through functions
- a function in MATLAB accepts a set of inputs and (usually) calculates a set of outputs
  - there can be 0 or more inputs
  - there can be 0 or more outputs
- the user of the function provides the inputs
  - the input values are called *arguments* to the function
- the function provides the outputs
- the user uses the name of the function to use the function
  - we say that the user *calls* the function

Functions

- you can find the names of elementary mathematical functions using the following command:

  >> **`help elfun`**

- this produces a long list of functions…
  - try it in MATLAB if you missed this lecture; the list doesn't fit on a lecture slide
- you can also use Help browser:

  >> **`doc elfun`**

Rounding functions

```
Rounding and remainder.
  fix           - Round towards zero.
  floor         - Round towards minus infinity.
  ceil          - Round towards plus infinity.
  round         - Round towards nearest integer.
  mod           - Modulus (signed remainder after division).
  rem           - Remainder after division.
  sign          - Signum.
```

Rounding functions

- **`round`** rounds the input value to the nearest integer and returns the rounded value

```
>> x = round(2.9)

x =

     3
```

call the function **`round`** with the argument **`2.9`**

- try other input values; use **`help round`** or **`doc round`** for information about the function

Rounding functions

- **`ceil`**, **`fix`**, and **`floor`** also round but in a different way
  - try them out to see the differences
  - use **`help`** or **`doc`** for more information

Trigonometric functions

- there are many trigonometric functions
- **`cos`**, **`sin`**, and **`tan`** compute the cosine, sine, and tangent of the input value *in radians*

```
>> y = cos(pi)

y =

     -1
```

call the function **`cos`** with the argument **`pi`**

Trigonometric functions

- **`cosd`**, **`sind`**, and **`tand`** compute the cosine, sine, and tangent of the input value *in degrees*

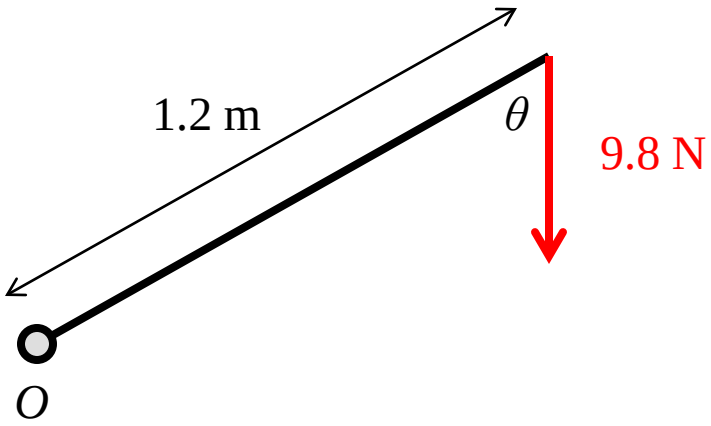
```
>> y = sind(90)

y =

     1
```

call the function **`sind`** with the argument **`90`**

- what is the value of **`tand(90)`** ?

Trigonometric functions

- calculate the magnitude of the torque about O when $\theta = 60^{\circ}$



Functions with multiple inputs

- a MATLAB function can have multiple inputs
- to call a function with multiple inputs, supply the arguments separated by commas
- consider the plot function which can be called with 1, 2, or more arguments

```
>> help plot
```

Functions with multiple inputs

```
>> x = -180:5:180;
>> y = sind(x);
>> plot(y);
>> plot(x, y);
>> plot(x, y, 'r.');
>> plot(x, y, 'r.', x, cosd(x), 'b:');
>> line(xlim, [0 0]);
>> line([0 0], ylim);
```

try this in MATLAB to see what the different versions of **`plot`** do

Function with multiple outputs

- many MATLAB functions have multiple outputs
- to store the multiple outputs, assign the outputs to a vector of comma separated variable names

```
>> help sort
```

Functions with multiple outputs

```
>> x = [2 5 4 1 3];
>> y = sort(x);
>> [y, idx] = sort(x)
>> [y, idx] = sort(x, 2, 'descend')
```

try this in MATLAB to see what the different versions of **`sort`** do