Introduction to programming

Last tutorial we used Matlab as a calculator. Matlab is also a programming language. You can save a series of commands you find yourself repeatedly typing as a text file ending in .m to create a program.

Example 1

For instance, the formula to find the equivalent resistance of two resistors in parallel is shown below

\[
\frac{R_1 \times R_2}{R_1 + R_2}
\]

To find the equivalent resistance of a 10k resistor in parallel with an 18k resistor, type from Matlab’s command line:

\[
\text{R1 = } 10e3; \quad \text{R2 = } 18e3; \quad \text{Requiv = } (\text{R1}\times\text{R2})/(\text{R1}+\text{R2})
\]

That’s fine for a relatively simple problem with two resistors, but what if you had to do that calculation many times? You would write a program to do it. Just like you can type

\[
\text{sin(1)}
\]

to evaluate what sin(1) is, you could write a program

\[
\text{Requiv}(10e3, 18e3)
\]
to evaluate the parallel combination of two resistance values.

Do it now. Open up a text editor by typing, at the Matlab command prompt

\[
\text{edit}
\]

Next enter these lines:

```matlab
definition result = Requivalent(R1, R2)
result = (R1*R2)/(R1+R2);
```

Save the file in the current directory as “Requivalent.m” and close the editor.

Now from the Matlab prompt, run it by typing:

\[
\text{Requivalent}(10e3, 18e3)
\]

If it returned 6.428e3 = 6.43k then you just wrote your first Matlab program. Note that you could assign the result to a variable in one step like

\[
\text{R = Requivalent}(10e3, 18e3)
\]
Analyzing this program line-by-line:

Line 1: `function result = Requivalent(R1, R2)`
   - The first word of any Matlab program must be “function”
   - `result = Requivalent(R1, R2)` means the program is stored as a file named Requivalent.m and run from the Matlab prompt by typing Requivalent.
   - The two variables R1 and R2 in the parentheses means the program takes two numbers; the one number to the left of the equals sign means it returns one number. The first number it takes is called R1 and the second is called R2. “Taking” these numbers means the program is called with this numbers, like this:
     - `Requivalent(10e3, 18e3)`
     - Internally, this sets R1 = 10e3 and R2 = 18e3.

   - `result` is the name of the variable that the program will return to the command line.

Line 2: `result = (R1*R2)/(R1+R2);`
   - This completes the calculation, and stores the answer in the variable called “result”.

**Example**

Design a program called `Multiply` that takes three numbers and returns their product

```matlab
function result = Multiply(num1, num2, num3)
    result = num1 * num2 * num3;
end
```

**Problem 1**

Write a program called `F2C` that takes a temperature in Fahrenheit and converts it to a temperature in Celsius. You may need to quickly search the web to find the conversion factor.

- The program takes one variable called “f” and returns one variable called “c”. To test it, type
  ```matlab
  >> result = F2C(70)
  ```
  - It should set the variable `result` to about 21.1.

When done, write your program code (it should be two lines) on the solution sheet I provided.

**Problem 2**

Write a program called `GradeNeeded` that takes 3 numbers; your current course grade (call it `CurrentGrade`), the percent weight of your final exam in your total grade (call it `ExamWeight`) and your desired course grade (call it `DesiredGrade`). Have it return one number: `ExamGrade`, the score you need to make on the final exam to achieve this grade. The equation is:

```
ExamGrade = \frac{DesiredGrade - CurrentGrade(1-ExamWeight)}{ExamWeight}
```

To test, try asking what grade you would need on a final exam if your current grade is 93, you want a 95, and the final exam is worth 0.33 (that is, 33%) of your final grade.

When done, write your program code (it should be two lines) on the solution sheet I provided.
**Problem 3**

A common electrical engineering circuit fragment is a *high pass filter*, shown below.

![High Pass Filter Circuit](image)

A high pass filter takes an input voltage signal \( V_{in} \) and removes high frequency content before passing it to the output voltage \( V_{out} \). Specifically, it allows frequencies lower than \( f_o = \frac{1}{2\pi RC} \) Hz to be passed, and starts blocking signals higher than \( f_o \) Hz.

Create a program called MyFilter that takes just one argument, \( f_o \), and returns the resistance value \( R \) required to make a high pass filter with cutoff \( f_o \) assuming you are using a 10uF capacitor. Write the program code on your assignment sheet.

---

**If/Else/End Statements**

Matlab allows one to test a variable to see if it is equal to, greater than, or less than another variable, and then run appropriate code.

**Example**

Design a program called IsPositive that takes one number \( x \) and returns a 1 if \( x \) is greater than zero, and -1 if it is equal to or less than zero.

```matlab
function result = IsPositive(x)
    if x > 0
        result = 1;
    else
        result = -1;
    end
end
```

Allowable tests are \( >, \geq, <, \leq, \sim \) (that means “not equals”) and \( == \) (means “equals”). Note that you don’t need to indent your code, but it makes it more readable.

Here’s another more complex example:
Example
Design a program called WillIGetAnA that takes 3 exam grades and returns a string that is either ‘you earned an A!’ or ‘not this time’.

```matlab
function result = WillIGetAnA(grade1, grade2, grade3)
    average = (grade1 + grade2 + grade3) / 3;
    if average >= 90
        result = 'you earned an A!';
    else
        result = 'not this time';
    end
end
```

You can perform multiple tests using && and ||. && means “and”, for instance to see if $x$ number is greater than zero but less than one you could say

```matlab
if x > 0 && x < 1 ...
```

Similarly, to run some code if $x$ is either less than zero or greater than 1 you could say

```matlab
if x < 0 || x > 1 ...
```

for instance look at the example below for several coding techniques:

Example
```matlab
function HowDidIDo(grade)
    if grade >= 90 && grade <= 95
        disp('not perfect, but very, very good')
    end
    if grade > 95
        disp('Essentially a perfect grade.')
        disp('You need to get some sleep!')
    end
    if grade < 90
        disp('not an A')
    end
end
```

Note several things about this program:
- You can run it by typing >> HowDidIDo(75, 100, 95)
- There is no output variable for this example. There doesn’t have to be. If you were to type >> a = HowDidIDo(75,100,95) it would give you an error because there’s no result variable that HowDidIDo returns. Instead, the program just prints answers directly to the screen using the command disp (short for “display”).
- You don’t need an else statement.
- You can have multiple statements between an “if” statement and its associated “end”.

Note several things about this program:
**Problem 4**

Modify the above program so that it takes 4 numerical grades and prints your letter grade to the screen. Call it MyGrade. For instance, MyGrade(88, 97, 82, 75) should return a ‘B’ since the average is 85.5.

**Loops**

The bread-and-butter of any program language is its ability to perform repeated calculations in a “loop”. The simplest of examples:

```
function ArmyNavy
for i=1:10
    disp('Go Army, Beat Navy')
end
```

Note that this program has no inputs AND no outputs…it just prints ‘Go Army, Beat Navy’ to the screen (although it seems doubtful Army will follow this advice.)

Line-by-line analysis:
First, recall that to Matlab 1:10 creates this vector: [1 2 3 4 5 6 7 8 9 10]

for i=1:10 creates a variable called i and sets it equal to 1, the first number in the sequence. It then runs whatever code follows until the “end” statement (i.e. it runs the disp statement). At the “end” statement Matlab jumps up to the for i = 1:10 line, sets i = 2 (the next number in the sequence 1:10) and the program starts running statements downwards until it reaches the “end” statement and restarts again, this time with i=3. It does this until i=10. After i=10, when it hits the “end” statement, it doesn’t go back up to the i=1:10 statement because its has used all the values in the sequence; rather it continues executing downwards from the “end”. Since that was the last statement, the program ends.

Here’s another example that shows what’s happening inside the loop clearly:

```
function CountUpToN(N)
for i=1:N
    disp(i)
end
```

Run it. What does it do?
Variables are often incremented within loops. The statement
\[ x = x + 5; \]
doesn’t make sense mathematically, but it makes perfect sense to Matlab that evaluates the statement to the right of the equals sign first, and then makes the variable on the left equal to this new value. In other words,
\[
\begin{align*}
x &= 0; \\
\text{for } i &= 1:10 \\
& \quad x = x + 1;
\end{align*}
\]
will start off setting \( x \) equal to 10, and then will execute a loop 10 times, where each time through it will increment \( x \) by 1. When it finishes, \( x \) will equal 10.

**Problem 5**

Modify the program in the boxed example above so that it returns the sum of the numbers from one to N. Hint: you’ll need to use the material in the paragraph above too. Call the program `PyramidSum`, and test that `PyramidSum(10)` returns \( 1+2+3+4+\ldots = 55 \). Copy the program code into your assignment sheet and print it off.

**Problem 6**

You can call functions from within functions. For example, `PyramidSum(PyramidSum(4))` will cause the inner `PyramidSum` to add the numbers \( 1+2+3+4 = 10 \), and pass that as the argument to the outer `PyramidSum` which will then add the numbers \( 1+2+3+4+5+6+7+8+9+10 = 55 \). It works. What is the `PyramidSum(PyramidSum(PyramidSum(50)))`? (That’s three of them)

**Congratulations**

While there are many details left to learn about programming in Matlab, you now know the essentials about functional programming, branching (the technical term for the if-else-end statements), and looping. As you progress through the electrical and computer engineering curriculum you will learn how to combine your ever-increasing programming knowledge with your knowledge of matrix and vector algebra to turn Matlab into a natural extension of your mind for solving numeric engineering problems.