**Calculating and Plotting Resultant Force**

You will be writing a program in MATLAB that allows the user to input any number of forces in either rectangular form or polar form or even a mixed combination of these forms (some forces in rectangular and some in polar). The program will calculate the resultant force and will also plot the individual forces and the resultant force.

1. Create a new function file by clicking on New then selecting function.
2. The first line of your function file should look like this:

function [ output\_args ] = Untitled( input\_args )

Replace Untitled with a name for your function (ResForce).

***Note: when naming a function, you must follow the same rules as for naming variables and script files (any combination of letters, numbers, and underscores but must begin with a letter).***

1. Your function will have one input argument which will be the number of forces the user wishes to enter. Pick an appropriate variable name for this and use it in place of input\_args in the first line of your function.

***Note: this input argument is a replacement for prompting the user for the number of forces using an input statement; that is, it is equivalent to this: Number = input(‘Enter Number of Forces’);***

1. Your function will have two output arguments, the magnitude of the resultant force and the angle of the resultant force. Pick appropriate variable names for these two output arguments and enter them (separated by commas) in place of output\_args in the first line of your function.

***Note: the output arguments will take the place of fprintf statements. However, we will only get numbers with no fancy formatting or text accompanying the values.***

1. At this point, the first line of your function should look something like this (variable names for input and output arguments and the name of your function will of course vary from mine):

function [ Res\_Mag, Res\_Angle ] = ResForce(NumberOfForces)

***Note: The next section of your function should contain comments about what the function does and how to run the function.***

**Now add the following to your function. Each part can be tested as you go by leaving off semicolons or by setting a breakpoint and stepping through the function using Debug.**

1. Use the ***zeros*** command to create two vectors, an X-Component Vector and a Y-Component Vector. These vectors should each have a length equal to the number of forces the user wishes to enter and initially be filled with zeros.
2. Now create a ***for*** loop that will prompt the user to enter each of the forces one at a time. Your ***for*** loop should do the following:

* First ask the user if the force he/she wishes to enter is in rectangular form or polar form (use a ***menu*** statement for this).
* Set up a conditional statement (***switch*** or ***if …. else***) dependent on user’s response for the format of each force.
* If the user selected rectangular form, prompt the user for the x and y components of the force (***input*** statements) and enter the results into your X-Component Vector and your Y-Component Vector. The first force entered would be the first entry in these vectors, the second force entered would be the second entry in these vectors, …
* If the user selected polar form, prompt the user for the magnitude and angle of the force (***input*** statements). Use these values to calculate the x and y components for the force, then enter the results into your X-Component Vector and Y-Component Vector.
* Don’t forget to ***end*** your for loop.

1. **Test your Program:**

* Either leave the semicolons off your X-Component Vector and Y-Component Vector or set a Breakpoint at the first line of your function and step thru with the Debugger.
* Run your function at the command prompt by simply typing the first line of your function without the keyword ***function*** and supplying the appropriate input arguments. For example, if your first line of code looks like this:

function [ Res\_Mag, Res\_Angle ] = ResForce(NumberOfForces)

You would run your function at the command prompt (assuming 2 forces) as follows:

>> [ Res\_Mag, Res\_Angle ] = ResForce(2)

* Your program should now be displaying a menu statement asking if the force is in rectangular form or polar form. Choose rectangular and enter an x-component of 2 and a y-component of 3.
* Your program should now once again be displaying a menu statement asking if the force is in rectangular form or polar form. Choose polar and enter a magnitude of 10 and an angle of 60 degrees.

***Note: you will get error about your output arguments not being defined properly. This error is OK at this point because you haven’t written the code yet to determine the output arguments.***

* Verify that the X-Component Vector = [2 5] and the Y-Component Vector = [3 8.66]. If not, there are errors in your code that need to be fixed. Fix the errors and repeat the test before moving on.

1. After your ***for*** loop, when all forces have been entered by the user, compute the resultant force:

* First compute resultant force in the x and y directions (Hint: ***sum*** is a useful).
* Then compute magnitude and angle. Remember, if the x-component is negative you need to adjust the angle by 180 degrees.

1. **Test your program**

* First complete the following table by hand (MATLAB command prompt or calculator):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function** | **Magnitude** | **Angle** | **X-component** | **Y-Component** |
| **F1** |  |  | 2 | 5 |
| **F2** | 10 | 135o |  |  |
| **F3** | 5 | 205o |  |  |
| **Resultant Force** |  |  |  |  |

* Now type clear at the command prompt to clear your workspace. Note: you don’t have to clear the workspace to run a function – I’m having you clear the workspace so you can see exactly what the function will add to the workspace.
* Run your function and see if you get the same results. Make sure you enter each of the forces using the original information – don’t use any calculated values from the table. In other words, enter F1 in rectangular form and F2 and F3 in polar form when prompted.
* If your program doesn’t produce the same result you calculated, check your code and calculations. Then clear your workspace and run the function again.

1. Look at your workspace. What variables are now in your workspace? What do these variables represent in your function?

***Note: When you run a script file, every single variable in that script file will appear in the workspace. Your script file can also access values already in the workspace. With a function file, the only variables that will appear in the workspace are the output arguments. The only values the function can access are the input arguments.***

1. Now add a plotting feature to your function. On the same plot, plot each individual force in one color (your choice) using dashed lines and plot the resultant force in a different color using a solid line style.

Hint #1: Try these commands at the MATLAB command prompt:

>> hold on

>> plot([0 2],[0 3],'k-s','LineWidth',2)

>> plot([0 10],[0 -5],'r--s','LineWidth',2)

>> hold off

Hint #2: For Loop

1. **Test your program**

Use the same three forces from the table in Step 10 to see if your plot features works correctly. Paste your resulting plot in the space indicated.

**PASTE PLOT HERE:**

1. Now, you will add one final feature to help ensure that the user enters the correct values and doesn’t have to start the program all over again if he/she makes an error in entering one of the forces.

* Go back to the conditional statement where the user is prompted for the x and y components of the force.
* Add an ***fprintf*** statement that tells the user the values that were entered for the x and y components.
* Ask the user if these values are correct.
* Add a ***while*** loop that will run if the values are not correct, prompt the user for new values, inform the user of the values entered, and ask again if the values are correct.
* Add exactly the same features described above to the section of the conditional statement where the user is prompted for the magnitude and angle of the force.

1. **Test your program**

Test your program and make sure you are able to re-enter new values whenever you indicate that the values entered were not correct.