MA #3 Guide

Overall Goal: In Task 1, you will calculate the deflection (downward movement) of a beam. In Task 2, you will create functions to analyze resistors. You will build on your prior MATLAB knowledge and practice your knowledge of logical arrays, conditional statements, and data validation while completing this MA.

What is a .p file?

Included in the MA #3 files you will see several .p files. These are locked files (you cannot open or edit them) that contain the correct function code. Place the files in your current folder. Then, you can call these functions in your main script the same way you would call any other function.

Why would you use the .p file?

If your code is not matching the sample output and you are not sure why, use the .p file. If you run your code and call the p.file function and you get the correct sample output, then you have identified that the error is in your function file and not the main script. For the MA, you get separate credit for a correct main script and a correct function.

MA Requirements:

Algorithm: Submitted to Blackboard
MATLAB Code: Submitted to Zybooks (5 total submissions)

PROBLEM 1

Task 1: Function – Calculate the beam deflection based on x location

- Create a new function named **FindDeflection.m**
  - Inputs: (1) Variable storing the location on the beam (this can be a single value or a vector of values)
  - Outputs (1) Variable storing the calculated deflection (this can be a single value or a vector of values depending on the input)
- Within your function, use the given piecewise function to calculate the deflection, $v(x)$.
  - No unit conversions are required.
  - HINT: In a piecewise function, a different equation is used depending on the x-value.

Task 2: Main Script – Calculate the deflection at a user-specified x location

- Prompt the user to enter a location on the beam between 0 and 360 [in].
- Call your **FindDeflection.m** function to calculate the corresponding deflection.
  - Inputs: (1) Variable storing the location on the beam
  - Outputs (1) Variable storing the calculated deflection
- Output to the command window the x-location and corresponding deflection value.

Task 3: Main Script – Calculate a deflection profile

- A deflection profile means you are calculating $v(x)$ for all locations from x=0 to the user inputted value.
- Create a vector of x location values from 0 to the user inputted value.
  - HINT: Think about what spacing in your x-vector would be me most useful.
- Call your **FindDeflection.m** function to calculate the corresponding deflections.
  - Inputs: (1) Variable storing the locations on the beam
  - Outputs (1) Variable storing the calculated deflections
- Plot your deflection profile. Your plot must include: title, axis labels, gridlines, and smooth solid line (theoretical data) as individual points
- Plot 1: **x-axis:** x-locations on the beam; **y-axis:** theoretical deflections, $v(x)$ [in]

PROBLEM 2

Background: The colors on a resistor indicate the resistance. Here, you will develop functions to report the resistance when given the color bands, or the color bands, when given the resistance.

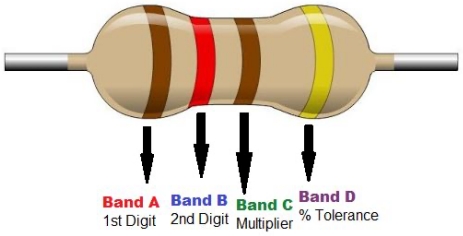
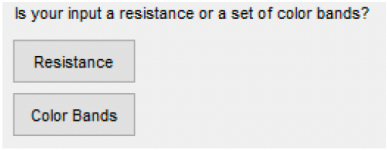


Image Source: circuitbasics.com

Task 1: Main Script – Obtain user input from a menu

- Using menu, ask the user to select if their input with be a resistance or color bands:



- Complete data validation on the user selection. If the user exits out of the menu without making a selection, an error should be given, and the program terminated.

Task 2: Function – Determine the resistance band colors

- Create a new function named **Resist2Color.m** that will convert a numerical resistance in [ohms] to the corresponding color bands
  - ➢ Inputs: (1) Numerical value for resistance in [ohms] as a vector
    - ✧ HINT: Each digit should be considered a unique element and entered with a space
    - ✧ Example Input: `R = [1 0 0 0]`
  - ➢ Outputs (1) 1 x 3 string vector that contains the corresponding color bands in order
    - ✧ Example Output: `C = ["Brown" "Black" "Red"]`
- Use the information provided in the **ColorCode** and **Multipliers** variables to determine the corresponding color bands for the input resistance. The last page of this guide provides a reference example!
  - ➢ HINT: Load the **ColorGuide.mat** file inside your function so that you can use the **ColorCode** and **Multipliers** variables inside your function.

Task 3: Function – Determine the resistance in ohms

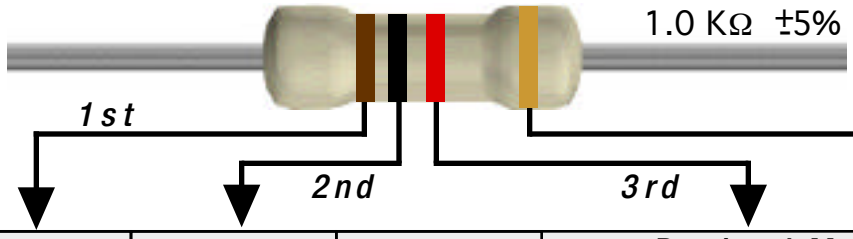
- Create a new function named **Color2Resist.m** that will convert a set of color bands to a numerical resistance in [ohms]
  - Inputs: (1) 1 x 3 string vector that contains the corresponding color bands in order. You may assume the text capitalization will always match the information provided in the **ColorCode** and **Multipliers** variables.
    - Example Input: `C = ["Brown" "Black" "Red"]`
  - Outputs (1) Numerical value for resistance in [ohms] as a regular number (not a vector!)
- Use the information provided in the **ColorCode** and **Multipliers** variables to determine the corresponding resistance for the color bands. The last page of this guide provides a reference example!
  - HINT: Pay careful attention to indexing!
  - HINT: Load the **ColorGuide.mat** file inside your function so that you can use the **ColorCode** and **Multipliers** variables inside your function.

Task 4: Main Script – Obtain user input and convert values

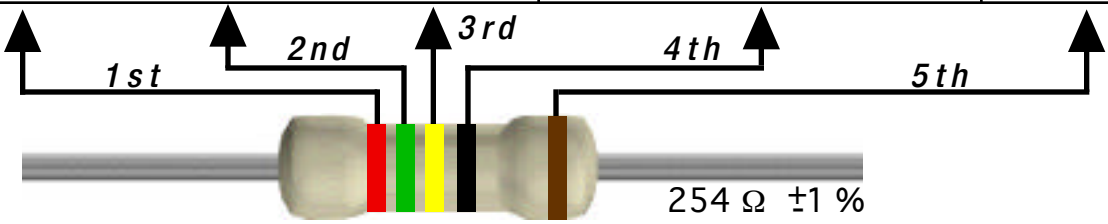
- Ask the user to entire either a string vector with three color bands or a resistance as a vector. The appropriate question should be asked based on the user-input from **Task 1**.
- Perform Data Validation on the user entry:
  - If the user is entering a resistance:
    - Check that all the digits after the first 2 are zeros. If not, create an error message and terminate the program.
    - If a correct entry was inputted, call your **Resist2Color.m** function to determine the color bands.
    - Output the color bands to the command window.
  - If the user is entering color bands:
    - Check that the string vector input contains exactly three entries. If not, create an error message and terminate the program.
    - If a correct entry was inputted, call your **Color2Resist.m** function to determine the resistance in [ohms].
    - Output the resistance to the command window.
      - HINT: If you want to use the Ohms symbol in your output, the placeholder in your `fprintf` statement is `%c` where you want the character symbol to show up and the corresponding variable should be: `char(937)`.

RESISTOR COLOR CODE GUIDE

4- Band Code

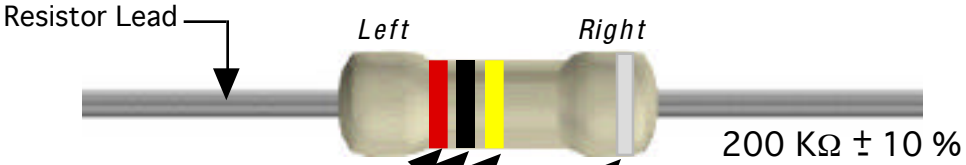


Color	1st Band	2nd Band	3rd Band	Decimal Multiplier		Tolerance
Black	0	0	0	1	1	
Brown	1	1	1	10	10	± 1 %
Red	2	2	2	100	100	± 2 %
Orange	3	3	3	1K	1,000	
Yellow	4	4	4	10K	10,000	
Green	5	5	5	100K	100,000	
Blue	6	6	6	1M	1,000,000	
Violet	7	7	7	10M	10,000,000	
Gray	8	8	8		100,000,000	
White	9	9	9		1,000,000,000	
Gold					0.1	± 5 %
Silver					0.01	± 10 %
None						± 20 %



5- Band Code

Calculation



First Band Red ---------- 2
Second Band Black ---------- 0
Multiplier Band Yellow -- x10,000
Tolerance Band Silver -------10 %

The Gold or Silver band is always placed to the right.
The resistor value is read from the left to right.

If there is no tolerance band, then find the side that has a band closest to a lead and make that the first band.

Equation

20 x 10,000 = 200,000

1,000 = 1K

Resistor = 200 KΩ
with a ± 10 % Tolerance