

MATLAB and Simulink in a Hybrid Electronics Course during COVID

Michele McColgan

Department of Physics & Astronomy



My Matlab Grader Experience In General Physics

- Created 100 General Physics Problems
- Piloted in my general physics course with freshman physics majors - 2017
 - 3 of 10 in each problem set were computational
 - If students passed the assessments, they received full credit
 - Most students tended to do these problems first, some not at all
 - Students persisted with multiple attempts
 - You must practice with students to get everyone to feel confident they can be successful.

MATLAB Grader in Electronics Course during COVID

- Created ~30 problems
 - Ohm's Law, Voltage Dividers, Transistors, Op Amps, RLC circuits
- Motivation
 - Novelty and variety in online assignments in case we go remote
 - Each unit has 4 homework assignments
 - MATLAB Grader problems, Canvas Quiz, Gradescope written work, Team Project that includes design work, building circuits, and MATLAB and Simulink computational tasks.
- In combination with MATLAB Live Scripts and Simulink circuit models
 - MATLAB Onramp and Simulink Onramp

SCDV 230 Electronic Instrumentation

Reorder Content

> DC Circuits

> AC Circuits

> Transistors

> Op Amps

> Assignment 1 Ohm's Law

> Assignment 2 Series and Parallel Circuits

> Assignment 3 Voltage Divider

> Assignment 4 - Transistors as switches

> Assignment 5 Transistors as Amplifiers

> Assignment 6 - Op Amps

> Assignment 7 - Op Amp Applications

> Video Examples

ADD ASSIGNMENT

Manage People

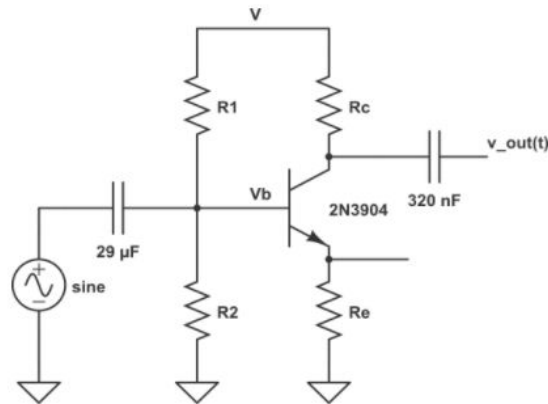
SCDV 230 Electronic Instrumentation

Duration (EDT): Not Specified - Not Specified

Course Description

This course includes assignments on the topics of ohm's law, series and parallel circuits, voltage dividers, npn and pnp bipolar junction transistors as switches and amplifiers, inverting and non-inverting op amps and op amp circuit applications, capacitive and inductive reactive and impedance in AC circuits.

The problems here supplement the Canvas questions, Gradescope HW assignments, and team assignments on these topics.



MATLAB Grader Problem

npn emitter amplifier - find Vc

[Actions](#)[Back to Instructor View](#)

Write a function to calculate the collector voltage, V_c , labeled in the circuit below for the npn transistor acting as an emitter amplifier. The supply voltage is 12 V, the voltage at the base is $V_b = 1.8$ V, $R_b = 300\Omega$, and $R_e = 1000\Omega$. Assume the diode voltage for the transistor is 0.6 V and the current through the collector is approximately equal to the current through the emitter.

Matlab Grader will test your function with the above values. In the "How to call your function section" test your function with the given voltage and resistance values. In addition to these values, in the assessment section Matlab Grader will test your function with a supply voltage of 12V, $V_b = 5$ V, $R_b = 220\Omega$, and $R_e = 600\Omega$.



Function

[Reset](#) [MATLAB Documentation](#)

```
1 function Vc = co1Vol(V,Vb,Re,Rc)
2     Ve = ??;
3     ie = ??;
4     ic = ??;
5     Vc = ??;
6 end
```

Code to call your function

[Reset](#)

```
1 V = ??;
2 Vb = ??;
3 Re = ??;
4 Rc = ??;
5 vC = co1Vol(??,??,??,??)
```

[Run Function](#)

Assessment:

[Run Pretest](#) [Submit](#)

> V = 12 V, Vb = 1.8 V, Re = 300 ohms, Rc = 1000 ohms (Pretest)

V = 24 V, Vb = 5 V, Re = 220 ohms, Rc = 600 ohms

Assessment

Assessment* ?

Assessment Method: Correct/Incorrect ▼ ?

Test 1: V = 12 V, Vb = 1.8 V, Re = 300 ohms, Rc = 1000 ohms

Test Type



MATLAB Code ▼ ?





MATLAB Code* ?



```
1 % Run learner solution.
2 V = 12;
3 Vb = 1.8;
4 Re = 300;
5 Rc = 1000;
6 vc = colVol(V,Vb,Re,Rc);
7
8 % Run reference solution.
9 vcReference = reference.colVol(V,Vb,Re,Rc);
10
11 % Compare.
12 assessVariableEqual('vc', vcReference);
```

Feedback on Incorrect (in addition to default feedback) ?

Normal

B *I* U **M**  

TEXT

CODE

INSERT

☒ Pretest ?

Test 2: V = 24 V, Vb = 5 V, Re = 220 ohms, Rc = 600 ohms

Test Type

MATLAB Code ▼ ?

MATLAB Code* ?

```
1 % Run learner solution.
2 V = 24;
3 Vb = 5;
4 Re = 220;
5 Rc = 600;
6 vc = colVol(V,Vb,Re,Rc);
7
8 % Run reference solution.
9 vcReference = reference.colVol(V,Vb,Re,Rc);
10
11 % Compare.
12 assessVariableEqual('vc', vcReference);
```

Student Solutions

Beginning

Progresses from
how to use a
function and
passing variables

Solution 2: All tests passed
Submitted on 17 Sep 2020 at 16:57 by t

```
1 function y = myFunction(R1,R2)
2     y = R1 + R2;
3 end
```

Solution 21: All tests passed
Submitted on 14 Sep 2020 at 18:11 by t

```
1 function y = myFunction(R1,R2)
2     y = R1 + R2;
3 end
```

Solution 17: 1 of 2 tests passed
Submitted on 14 Sep 2020 at 17:10 by t

```
1 function y = myFunction(R1,R2)
2     y = 50 + 100;
3 end
4
5
```

Solution 13: 1 of 2 tests passed
Submitted on 14 Sep 2020 at 16:54 by t

```
1 function y = myFunction(R1,R2)
2     y = 150;3000
3 end
```

Solution 2: 1 of 2 tests passed
Submitted on 14 Sep 2020 at 14:16 by t

```
1 function y = myFunction(R1,R2)
2     y = 150;
3 end
```

Now

To understanding
how the equations
work and identifies
gaps in content
knowledge

Solution 8: All tests passed
Submitted on 7 Oct 2020 at 23:26 by Micha

```
1 function Vc = colVol(V,Vb,Re,Rc)
2     Ve = Vb - .6;
3     ie = Ve / Re;
4     ic = ie;
5     Vc = V - Rc*ic;
6 end
```

Solution 7: 0 of 2 tests passed
Submitted on 7 Oct 2020 at 23:26 by Micha

```
1 function Vc = colVol(V,Vb,Re,Rc)
2     Ve = Vb - .6;
3     ie = Ve / Re;
4     ic = ie;
5     Vc = Vc - Rc*ic;
6 end
```

Solution 6: 0 of 2 tests passed
Submitted on 7 Oct 2020 at 23:25 by Micha

```
1 function Vc = colVol(V,Vb,Re,Rc)
2     Ve = Vb - .6;
3     ie = Ve / Re;
4     ic = ie;
5     Vc = V + Ve;
6 end
```

Solution 5: 0 of 2 tests passed
Submitted on 7 Oct 2020 at 23:25 by Micha

```
1 function Vc = colVol(V,Vb,Re,Rc)
2     Ve = Vb - .6;
3     ie = Ve / Re;
4     ic = ie;
5     Vc = V - Ve;
6 end
```

Improving MATLAB Grader

- For unlimited attempts, report # of attempts and # lines of code changed
- Random #s in the text - Canvas has this feature
- My own improvements:
 - More tests - use random #s
 - Add more variety - my default is to create functions
 - Find a way to use MATLAB Grader to lead students to create Live Scripts

How I use MATLAB Grader

- Review Material in Physics
- Start with simple problems
- Easy to practice scripts and functions and scripts that call functions
- Transitioning review/beginner physics topics to Matlab Grader
- Provide videos of:
 - How to use MATLAB Grader with problems from my General Physics course
 - How to create a MATLAB live script
 - How to create a Simulink Model

Why MATLAB Grader?

- Students don't need a Matlab license
- It works in any browser - some students have Chromebooks
- Grading is done automatically!

Why not MATLAB Grader only?

- Students still need to learn the Matlab IDE to use Matlab as a tool
- Limited functionality - Live editor

Simulink

- Applied Physics Majors going on to Engineering
- Designing, predicting, troubleshooting

Why Matlab Grader in detail?

- an instructor can gradually increase difficulty
- Introduce computational thinking without teaching it/spending a lot of time on it
- Introduce basic computational skills (symbols, vectors, functions, graphing, etc.)
- Demonstrate how to convert a word problem into a computational problem and understand the difference between algebra symbols and computational symbols.
- There's a place to begin - not starting from scratch

Matlab Grader Resources

- Documentation
- Example Problems
- Other Courses
- Videos



Assessing Scripts

Write Assessments for Script-Based Learner Solutions

For script-based solutions, you can easily create the most common assessments without writing code. Create an assessment by selecting the **Test Type** and specifying the solution code you are testing:

- **Variable Equals Reference Solution** — Check whether a variable in the learner solution equals the same variable in the reference solution within tolerance.
- **Function or Keyword Is Present** — Check for the presence of specific functions or keywords in the learner solution.
- **Function or Keyword Is Absent** — Check that certain functions or keywords are not present in the learner solution.
- **MATLAB Code** — Write the assessment using MATLAB[®] code.

The code behind the first three actions uses the same assessment functions as the functions used to check a function-based solution. You can click **Convert test to code** to see the code.

Execution Model

- When the learner submits a script-based solution for assessment, both the learner solution and the reference solution are run first. Your assessments then evaluate the learner solution.
- Each assessment runs sequentially and independently of the other assessments. If one assessment fails, the subsequent assessments still run.
- Variables created in one assessment are not available in the next one. Define all the required variables in each assessment.
- An assessment can refer to variables in the reference solution by referring to `referenceVariables.variable_name` in your code.
- If code terminates without errors, the assessment result shows a passed status. Otherwise, the assessment results show a failed status.

If the test is a pretest, the learner can view information about the assessment test by clicking the arrow to the left of the test name, regardless of whether the test passed or failed.

Assessing Functions

Write Assessments for Function-Based Learner Solutions

For function-based solutions, you can write MATLAB[®] code using the built-in functions that check for variable equality and keyword or function presence:

- `assessVariableEqual` — Check whether a variable in the learner solution equals a specified value within tolerance.
- `assessFunctionPresence` — Check for the presence of specific functions or keywords in the learner solution.
- `assessFunctionAbsence` — Check that certain functions or keywords are not present in the learner solution.

Execution Model

- Each assessment you write for a function-based solution typically includes a call to the learner solution. You can provide inputs to the function and evaluate any returned values. You can also call the reference solution to compare its output with the learner solution output.
- Each assessment runs sequentially and independently of the other assessments. If one assessment fails, the subsequent assessments still run.
- Variables created in one assessment are not available in the next one. Define all the required variables in each assessment.
- If code terminates without errors, the assessment result shows a passed status. Otherwise, the assessment results show a failed status.

If the test is a pretest, the learner can view information about the assessment test by clicking the arrow to the left of the test name, regardless of whether the test passed or failed.

- When you use `assessVariableEqual` with function-based problems, use the same name for any output variable created when calling the learner function as you would use in the learner function declaration. The default feedback messages reference the output variable name created in the assessment, and the learner may not recognize the output variable if it does not match the declaration.

```
b = 5;  
h = 3;  
area = triangleArea(b,h);  
areaCorrect = reference.triangleArea(b,h);  
assessVariableEqual('area',areaCorrect)
```