Matlab Grader

Michele McColgan
Department of Physics & Astronomy

My Experience

- Used Cody Coursework aka Matlab Grader
- 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.

Why Matlab Grader?

- Students don’t need a Matlab license
- It works in any browser
- Grading is done automatically!

Why not Matlab Grader only?

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

Why Matlab Grader in detail?

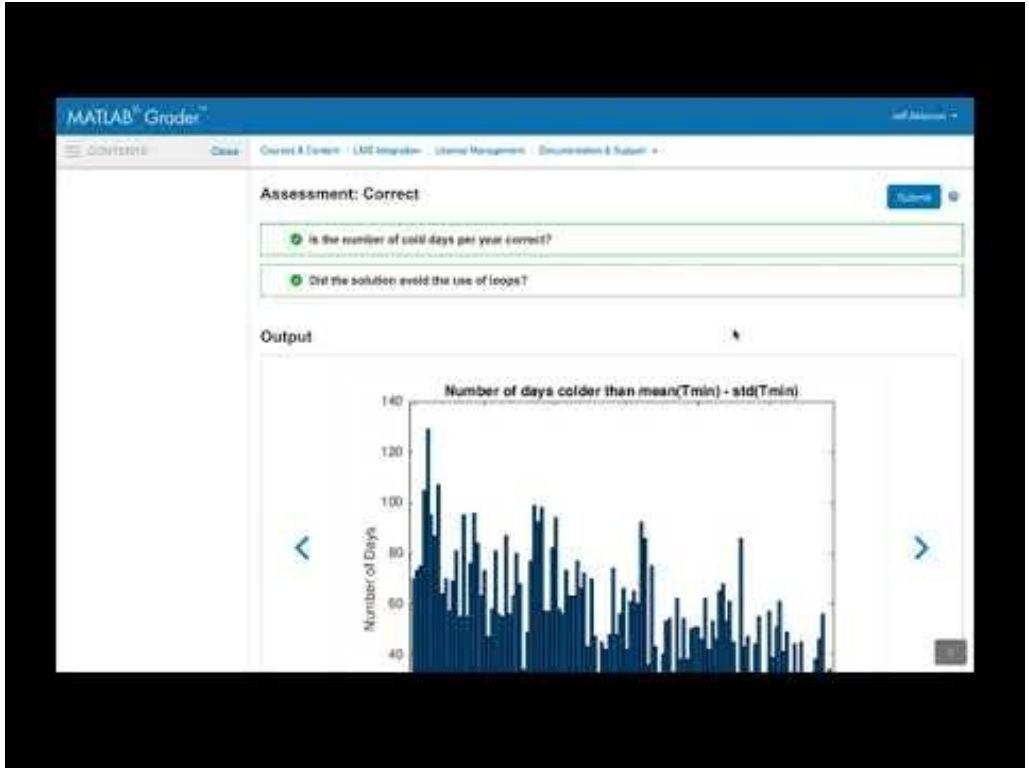
- an instructor can gradually increase difficulty
- Introduce computational thinking without teaching it/spending alot of time on it
- Introduce basic computational skills (symbols, vectors, functions, graphing, etc.)
- Demonstrate how to convert a word problem into a computational problem and understand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

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

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)
```

Examples of My Problems

- Function - Ohm’s Law
- Script - Temp in Kelvin
  - Function in a Script
- Plotting - Ohm’s law multiple resistors
- DIfficult - Viral & van der Wahls - modeling