Assessing computation skills with MATLAB

Teaching Computational Thinking Skills using MATLAB

Dan Burleson, University of Houston

October 21, 2019

Course Setup: ENGI 1331

- Freshmen engineering students of all disciplines
- Common curriculum for 4-7 sections of 60 students (250 - 400 total)
- 4-5 Undergraduate Teaching Assistants (UTAs) in class

Assessment grounded in feedback

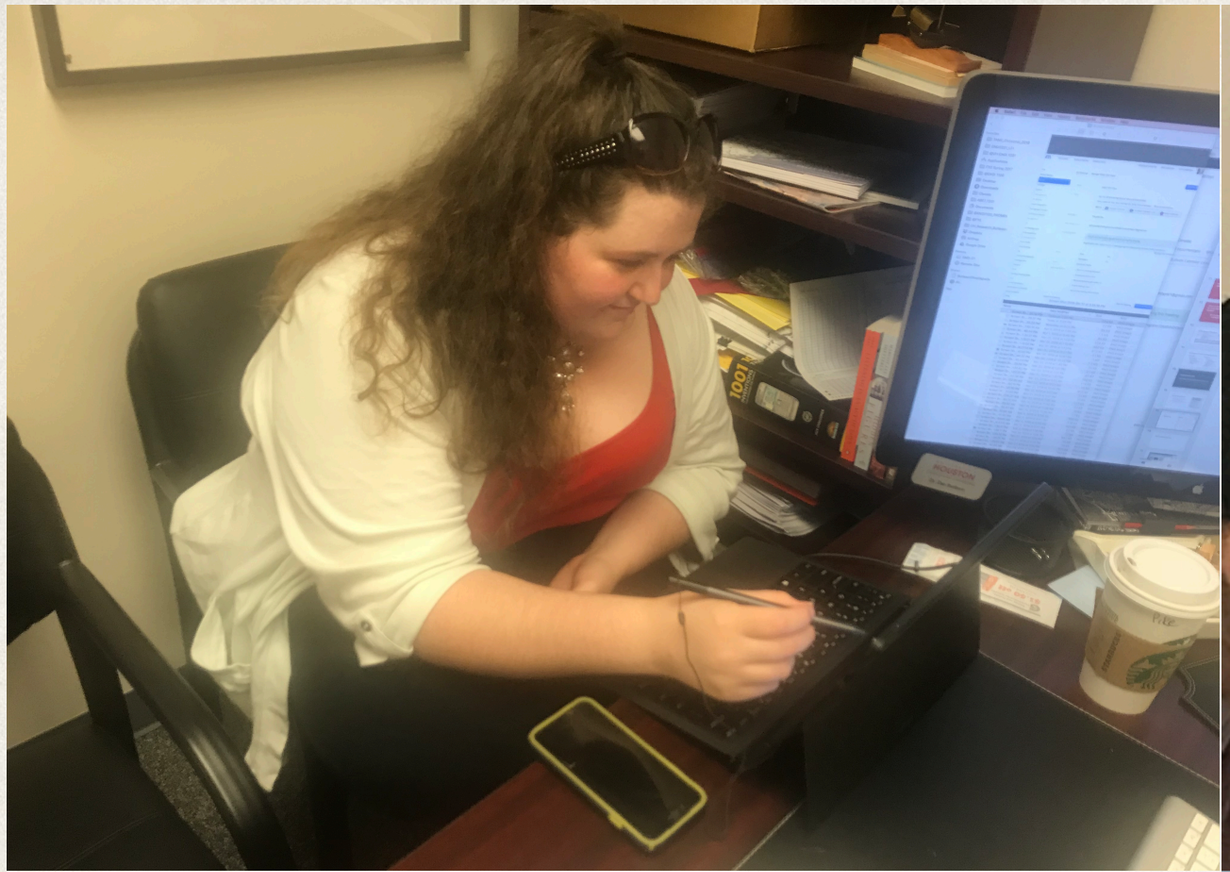
Blended Assessment Approach

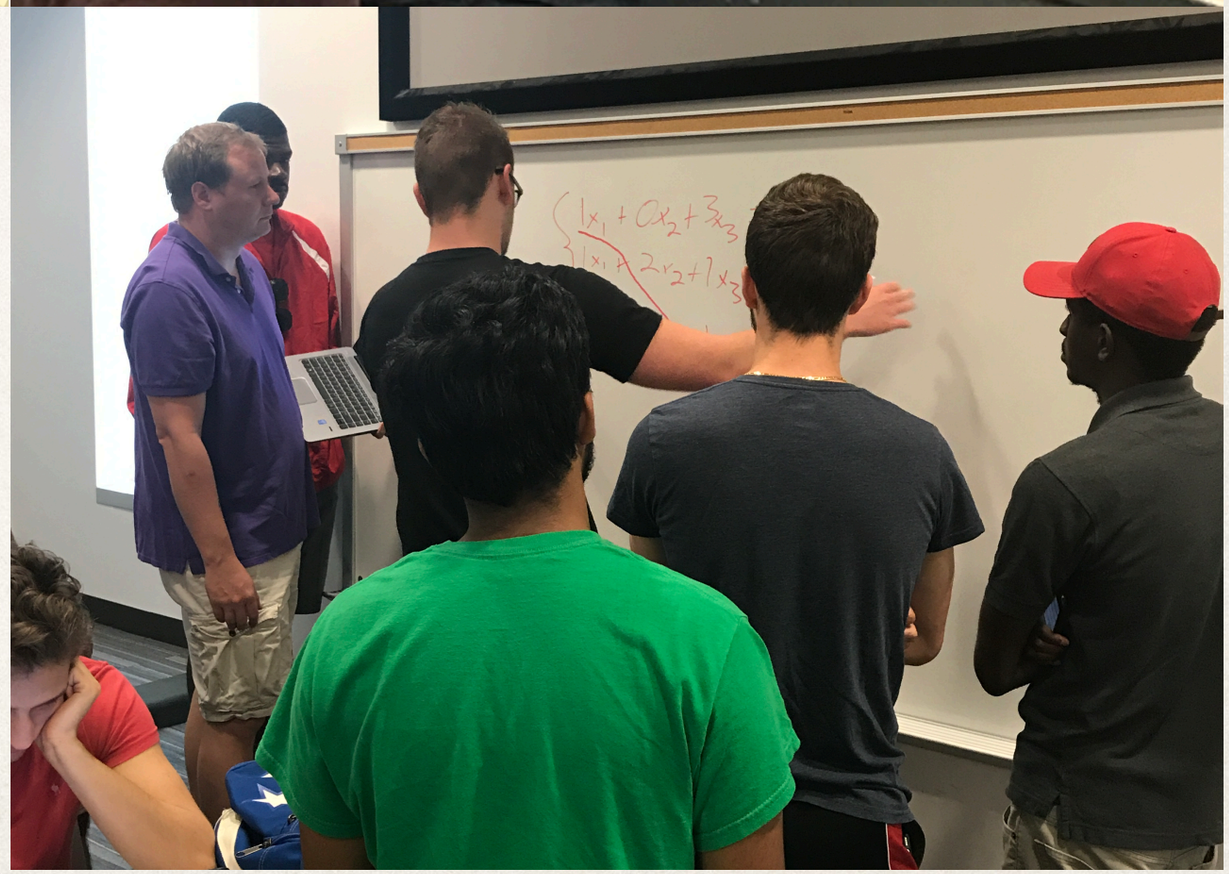
Blended by technology

- Written (hand solution or pseudo code)
- MATLAB coding (scripts and functions)
- Interactive textbook (Zybooks)

Blended by who

- Self assessment and reflection
- Peer-to-Peer (daily)
- UTA-to-Student (daily and weekly)
- Instructor-to-Student (weekly and monthly)





Undergraduate Teaching Assistants

Peer mentorship leading to genuine feedback

Interactive Textbook (Zybooks)

PARTICIPATION ACTIVITY 8.1.2: For loops.

Complete the following for loop using a start index, increment value, and end index.

1) Iterate with i from 1 to 100 (inclusive).

```
for

    % Loop body statements

end
```

Check Show answer

CHALLENGE ACTIVITY 4.7.1: Element-wise division: Convert pounds to kilograms.

This tool is provided by a third party. Though your activity may be recorded, a page refresh may be needed to fill the banner.

0 / 2

View Problem Details

Element-wise division: Convert pounds to kilograms

Assign row array weightKilo with the corresponding weight in kilograms.

Use the following conversion: kg = lb / 2.2

Function

Reference Solution Save Reset MATLAB Documentation

```
function  weightKilo = ConvertToKilo(weightPounds)
% weightPounds: Row array of weights in pounds

    % Assign weightKilo with the corresponding weight in kilograms
    weightKilo = weightPounds;

end
```

Code to call your function

Reset

```
ConvertToKilo([100, 190, 163, 211])
```

Weekly Assignment

Student given:

- Background
- Tasks
- Sample output
- Rubric
- Learning Objectives

Tasks

TASK 1: Data validation
Check that the image of the MRI scan is larger than the image of the tumor (both rows and columns). Also check that both images are viewable images (three-dimensional, three-layered matrices). If either of these conditions are not met, produce an error and terminate the program.

TASK 2: Find the tumor
Write a script which finds a brain tumor in a given MRI scan. When the brain tumor is found, output to the command window the pixel coordinates that the tumor is between. If the brain tumor is not found, output to the command window that the brain tumor cannot be found on this MRI scan. Your code should find the tumor, or determine the tumor is not on the MRI scan, in no more than three minutes of runtime.

NOTE: Determining where the tumor is on the MRI scan will require the entire tumor image to be checked. Checking only a portion of the image may lead the program to output a false positive.

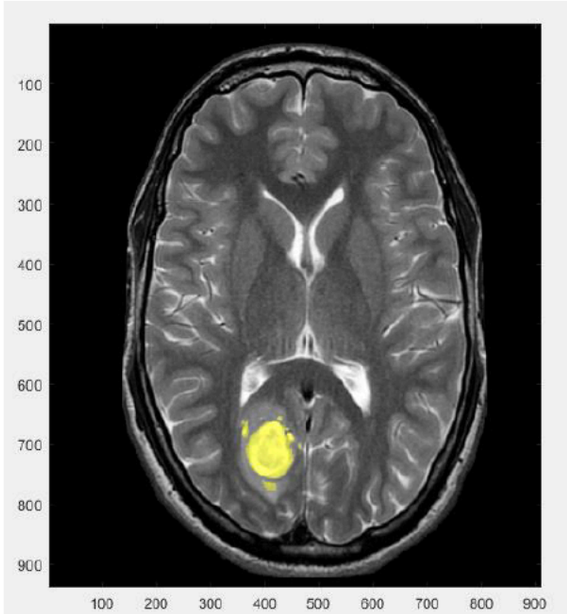
TASK 3: Highlight the tumor
If the brain tumor is found, display the MRI scan with the site of the brain tumor highlighted in yellow. To determine which portion of the tumor image is to be highlighted, assume that the tumor will always be within the region of the tumor image that is brighter than the average brightness of the tumor image. You may choose the method used to highlight the tumor.

HINT: Pay attention to the data type of the image when determining average brightness.

Sample Output:

Sample output for given data **MRI_Scan.png** and **Tumor.png.** Additional provided data files are for self-testing of code:

```
Command Window
The tumor is located between the coordinates [640,352]
and [777,465] starting from the top left corner of the MRI scan.
>>
```



Weekly Assignment Rubric

Does it work if the data changes?

Problem #1

General Script Documentation		6
Section Comments:		
Item	Pts	Pts Rec'd
Purpose of Problem defined	1	1
Variables defined (inputs, outputs, intermediate values)	2	2
Housekeeping Commands (clc, clear, close)	2	2
Headers for each task	1	1
Algorithm and Test Case Worksheet (zero for all if not included in submission)		**30**
Section Comments:		
Item	Pts	Pts Rec'd
Algorithm includes all necessary user inputs and data loads	3	3
Algorithm is logical and matches efforts made in code	4	4
Test Case - Task 1 correct	3	3
Test Case - Task 2 correct	3	3
Test Case - Task 3 correct	5	5
Test Case - Task 4 correct	4	4
Test Case - Task 5 correct	4	4
Test Case - Task 6 correct	4	4
Test Cases Scenarios		**64**
Section Comments:		

Test Case	Output	Pts	Pts
(1) TempData.csv has new values with less days and less temperature readings per day. Overall.csv has new values.	Task 1: Output based on selection correct	2	2
	Task 1: MA3Task1.csv is correct	2	2
	Task 2: Output correct for value and location of lowest average & highest average and lowest minimum and highest maximum.	4	4
	Task 3: MA3Task3.mat is correct	6	6
	Task 4: MA3Task4.csv is correct	4	4
	Task 5: Output correct for expected value and comparison to expected value	4	4
	Task 6: MA3_Final.csv correct	4	4
	Task 7: Plots all on same graph, correct, and include all required data and formatting.	6	6
	Extra Credit: Horizontal lines for long-term values included on graph	2	0
(2) TempData.csv has new values with more days and less temperature readings per day. Overall.csv has new values.	Task 1: Output based on selection correct	2	2
	Task 1: MA3Task1.csv is correct	2	2
	Task 2: Output correct for value and location of lowest average & highest average and lowest minimum and highest maximum.	4	4
	Task 3: MA3Task3.mat is correct	6	6
	Task 4: MA3Task4.csv is correct	4	4
	Task 5: Output correct for expected value and comparison to expected value	4	4
	Task 6: MA3_Final.csv correct	4	4
	Task 7: Plots all on same graph, correct, and include all required data and formatting.	6	6
	Extra Credit: Horizontal lines for long-term values included on graph	2	0

Collaborative Project Rubric

Checkpoints for feedback

Rubric for Deadline B

Section (Total Points)	Exemplary	Above Average	Developing	Score
Detailed Inputs (25)	Every input is defined with appropriate units. Data loads and user inputs used appropriately. (25)	Missing a few inputs and/ or units. Missing one or two instances where data loads or user inputs are needed. (19)	Missing most inputs and/ or units. Many data loads or user inputs are not specified. (0)	
Detailed Outputs (25)	All outputs are described. All formatted output to command window and file exports shown. (25)	Missing a few outputs to the command window and/ or file exports. (19)	Missing most outputs to the command window and/ or file exports. (0)	
Detailed Workflow (25)	Algorithm is logical and includes appropriate decision-making and loops where required. Includes all data validations, calculations, manipulations, and results. All items defined prior to use. (25)	Algorithm is illogical in one or two places. It may be missing a few calculations or show them incorrectly. (19)	Algorithm is illogical or incomplete in more than two places. (0)	
Complete	Algorithm	Algorithm is	Algorithm is	

Exams

Written and coding

MATLAB Programming (50 points) – *Approx. 50 minutes + 10 minutes for planning*

Background: *Aluminum alloys are made by adding other elements to aluminum to improve its properties, such as hardness or tensile strength. Table 1 shows the composition of a set of commonly used alloys which are known by alloy number (2024, 6061, and so on).*

Table 1. Aluminum Alloy Composition

Alloy	Composition of Aluminum Alloy (%)				
	Cu	Mg	Mn	Si	Zn
2024	4.4	1.5	0.6	0	0
6061	0	1	0	0.6	0
7005	0	1.4	0	0	4.5
7075	1.6	2.5	0.2	0	5.6
3560	0	0.3	0	7	0

Data is provided that contains the information in the table above:

- **Alloy.csv**: contains the alloy number of each composition
- **Data1.csv** and **Data2.csv**: Each contain a different combination of the percent of each element for the alloy numbers

TASK 1 (4 pts): Load **Alloy.csv**, and **Data1.csv.** This data will be used for Task 2 through Task 4. Prompt the user to enter a string array as a variable named Element for the chemical elements listed in Table 1: ["Cu", "Mg", "Mn", "Si", "Zn"].

TASK 2 (9 pts): Create a menu for the user to select an element. For the selected element, determine the average percent composition for each alloy and range of values (minimum to maximum value). Save the results as a row vector named **Task2.csv** in the following order: (1) minimum, (2) average, and (3) maximum.

TASK 3 (12 pts): Create a menu for the user to select an alloy number. Based on the selection, determine the two element that makes up the highest percent composition and output the values and elements as shown in the sample output.

ENGI 1331 – Exam 1 – Fall 2019
Grading Rubric

Section	Script Item	Possible	Given
TASK 1 (4 pts)			

Housekeeping, Load, inputs (4 pts)	`clc;clear all % 1pt`
	`Alloy = csvread('Alloy.csv'); % 1pt`
	`Data1 = csvread('Data1.csv'); % 1pt`
	`Element = input('Enter a … '); % 1pt`
		4	

	Task 1 Total	4	
TASK 2: (9 pts)			
Menu (1 pts)	`[Select = menu('Select an element', Element); % 1pt`	1	

Statistics (6 pts)	`AVG = mean(Data1(:,Select)); % 1pt mean, 1pt Data1 indexed`
	`MIN = min(Data1(:,Select)); % 1pt min, 1pt Data1 indexed`
	`MAX = max(Data1(:,Select)); % 1pt max, 1pt Data1 indexed`
		6	

Export (2 pts)	`csvwrite('Task2.csv', [MIN, AVG,MAX])` `% 1pt csvwrite, 1pt vectorize results`	2	
	Task 2 Total	9	
TASK 3: (12 pts)			
Menu	`Select2 = menu('Select an alloy number', string(Alloy)); % 1pt`	1	

Considerations for Assessment

- Who are you assessing?
- Who are you assessing?
- Who is doing the assessment?
- How often do you need to assess?
- What are your assessment constraints?

Alignment

Transparency

Consistency

Teaching Computation in the Sciences Using MATLAB®

MathWorks®

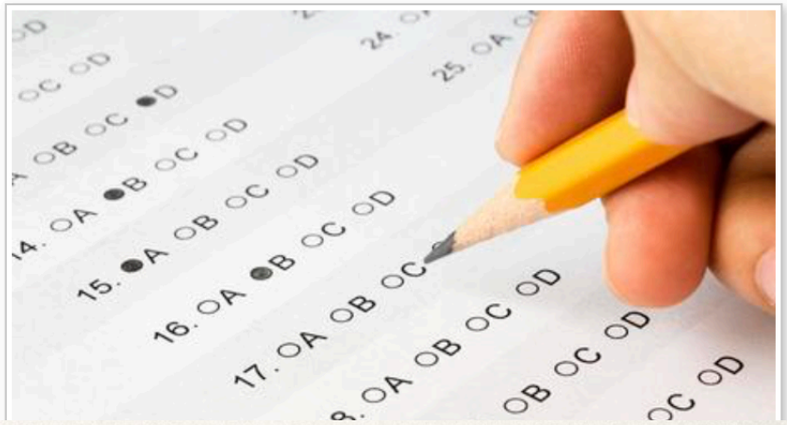
Teaching Computation in the Sciences Using MATLAB

- Why Teach Computation?
- **Skills and Techniques**
  - Computational Thinking
  - Data Analysis
  - Modeling
  - Visualization
- **Teaching Strategies**
  - Building Self-Efficacy
  - Cooperative Learning
  - Assessment

Assessment

Content on this page is derived from participant presentations, discussions, and breakout groups at the 2015, 2016, 2017, and 2018 workshops, as well as assessment resources from Pedagogy in Action.

In order to develop student competency in computation and modeling, STEM faculty need methods of assessment that accurately capture student understanding and growth (learn more about assessment from Pedagogy in Action). Assessment of student computation and modeling is a non-trivial task: the format and volume of material associated with exercises can be unwieldy to create and evaluate. From a student's perspective, assessment-related anxiety common in many STEM fields is often exacerbated by the real and perceived difficulty of computation and modeling. Overcoming these barriers and producing effective assessments is crucial to successfully teaching computation and modeling to students.



There are a variety of ways in which faculty can approach assessing student computation and

Highlights types of assessments and teaching activities that exemplify them.