Development of a MATLAB App for Improving Student Learning of the Use of Arrays

Thomas Murphy and Christopher Williams
Georgia Southern University/Georgia Southern University

Abstract

Students taking introductory programming courses commonly have problems understanding and utilizing fundamental concepts. One particular topic, arrays and their use in solving engineering problems, is consistently a trouble area for many students in our introductory MATLAB programming course. A MATLAB app was developed to help students learn the structure and proper use of arrays. MATLAB apps consist of a GUI and underlying code which provides functionality that is packaged together. The app allows users to select among the array topics: creating arrays, indexing, comparisons on arrays, and operations on arrays. For each topic, the app provides examples, practice problems, syntax, and solutions. Students can enter MATLAB statements and receive solutions and immediate feedback based on the solution. MATLAB apps are generally run from the MATLAB IDE but can be stand alone. The paper will discuss the design considerations for the app, the development process, and the testing of the app.

Keywords

Array, Introductory Programming, MATLAB

Introduction

Arrays and array indexing often cause students problems in introductory programming classes. Students regularly confuse the location of elements in an array with the value stored at the location and also commonly have difficulty using the result of an operation on an array as part of a larger solution to a problem. The use of arrays is even more confusing in MATLAB due to MATLAB's support of most operators and functions being defined for arrays (element by element operations) and the support of indexing arrays using either integer indices or an array of logicals.

A MATLAB app was developed to help students learn the structure and proper use of arrays. MATLAB apps consist of a GUI and underlying code which provides functionality that is packaged together. The app allows users to select among the array topics: creating arrays, indexing, comparisons on arrays, and operations on arrays. For each topic, the app provides examples, practice problems, syntax, and solutions. Students can enter MATLAB statements and receive solutions and immediate feedback based on the solution. MATLAB apps are generally run from the MATLAB IDE but can be made stand alone. Software tools such as MATLAB GUIs for practice and self-assessment have been in use since the mid 1990s[1, 2,] and have been shown to be effective as a learning tool and can be used to support both face-to-face and online courses[3, 4, 5].

It is intended that the use of the app will be incorporated into MATLAB programming courses and labs in subsequent semesters. Data will be collected to determine its efficacy and guide the development of apps for other topic areas.

Rationale for the App

Electrical and computer engineering students at Georgia Southern University take Computing for Engineers (ENGR 1731) typically in their first or second semester. Students have a wide range of programming backgrounds from no prior programming experience to formal programming courses using several languages. Arrays and their use in solving engineering problems is consistently a trouble area for many students in the course. Arrays are typically introduced much earlier in MATLAB programming courses than in courses using other high-level languages. Student performance on the first course exam is often very poor if the student is having difficulty understanding the concept and use of arrays. Students who have a poor grasp of arrays struggle throughout the course.

The authors have seen similar difficulty in introductory programming courses taught using other languages such as C++ and Java but MATLAB has some features that makes arrays more confusing to students. Most MATLAB operators and functions support operations on arrays of data (element by element operations) and MATLAB supports indexing arrays with a single index, an array of indices, and an array of logicals (logical true means the value is extracted). The array of logicals is often obtained as the result of an array comparison and students confuse the array location (logical or index) with the array values and will perform operations on array locations rather than array values and try to index arrays using array values. For students who do not properly name variables denoting what they contain and comment relevant sections of code, the confusion is compounded. Students in the introductory programming classes often cannot answer what their program variables hold and whether or not the array operation they have coded makes sense.

To identify which array concepts students were grasping, ENGR 1731 project submissions involving arrays and array operations assigned during the first month of class and the first course exam were examined. Fall 2018 and fall 2019 course sections were used; the course is currently taught once a year on the Armstrong campus of Georgia Southern University. Lab results were not used as an indicator as lab submissions reflect instructor feedback and corrections before the submission. Project submissions also have some amount of instructor or peer tutor feedback before submission but many students do not seek any help on projects before submitting them.

Short answer type questions on array creation and indexing arrays from the exams had student averages of 91.1% and 83.5% of the total points respectively. The answers to these questions were a single MATLAB statement. More involved problems from the exams involving arrays such as performing comparisons to identify a portion of data and then operating on the identified data or evaluating a formula over a time range and extracting a portion of the data had student averages of 60.8% of the total points. Problems from both well and poor performing students were examined to ensure that the student mastery of the topic was reflected by their numeric problem score. Students typically performed the first operation or two in the more complex

problems correctly, but then often incorrectly used the initial results for the subsequent portions of the problem. Similar results were seen in the student projects.

Students generally showed mastery of creating arrays, performing a comparison on an array of data, and indexing an array as long as they operation was done in isolation. Students generally could:

- Create arrays using MATLAB's colon operator, MATLAB's linspace function, and through concatenation.
- Operate on arrays using MATLAB built-in functions and element by element operations (+, -, .*, ./, .^).
- Perform comparisons on arrays using MATLAB element by element comparisons (<, <=, >, >=, ==).

Students had more difficulty combining the operations into a coherent solution to a problem. Students often had difficulty performing the tasks below.

- Using the result of a comparison to operate on a portion of array data that passed a test.
- Using the result of a comparison to extract a portion of data that passed a test.
- Interpreting what kind of result (a Boolean/logical, an index, or a value) they had after a comparison or operation.

Common examples of students' misuse of arrays and the results of operations on arrays is shown below. For this particular problem students were to create an array t, evaluate a formula g(t), plot g(t) versus t, extract a subrange of g(t) and plot the subrange.

```
% Example of Proper Code
% evalaute g(t) for 0.0 <= t <= 20.0 s
t = 0.0 : 0.1 : 20.0;
g = 16./(4.0*sqrt(t.^2 - t + 1));
% extract g(t) for 0.0 <= t <= 4.0 s
loc = t >= 0.0 & t <= 4.0;
tSub = t(loc);
gSub = g(loc);

% examples of student code for second part
% a) incorrect
t2 = (t >= 0 & t <= 4);
g(t) = 16/sqrt(4*(t2.*exp(2) - t2 + 1));

% b) incorrect
t2=t(0.0:.05:4.0);
gt2=(16./(sqrt(4*(t2.^2-t2+1))));

% c) correct but misleading names
tt=find(t<=4);
t2=t(tt);
```

```
g2=16./(sqrt(4*(t2.^2-t2+1)));
```

Code sample a) is incorrect, the student is misinterpreting a logical result from a comparison as a set of values. Code sample b) is also incorrect, the student is indexing an array with an array of time values (MATLAB requires array indices to be integers). Code sample c) generates the correct result, however the variable names are misleading. The variable tt which one might assume is time is storing an array of indices. It is difficult to tell if the student understood what they were doing and used poor programming style or if they did not understand how to solve the problem and were lucky.

Development of the App

Starting with the module level objectives for the array module, objectives 2 – 6 (listed below) were identified as objectives that lent themselves to problems that could be assessed automatically as well as provide meaningful automated feedback. Automatic verification of numerical results works well where automatic verification of conceptual answers is more problematic. Objectives 1, 6, and 8 would be difficult to assess automatically. The current GUI was limited to the problem types that could be reliably automatically assessed.

The objectives for the array module are given below.

1. Be able to explain the purpose of variables for programming.
2. Be able to create arrays in MATLAB.
3. Be able to index arrays in MATLAB.
4. Be able to properly use MATLAB element by element and matrix operations with arrays.
5. Be able to perform arithmetic and logic operations arrays.
6. Be able to apply MATLAB built in functions to arrays.
7. Be able to explain when to use vector and matrix operations versus element by element operations on arrays.
8. Be able to write MATLAB programs to solve problems using arrays.

Problems related to module level objectives were developed with a focus on problems that could be reused with different values for practice/assessment, required one or two MATLAB statements, and produced logical or numerical results that could be automatically assessed.

The design of the application's interface started with notes of needed functionality. Once those were established, sketches were drawn up of possible interfaces and controls placement. An initial mockup was created using MATLAB's GUIDE Layout Editor to test types of controls and placement of those controls. Multiple iterations were made to determine the final type and placement of controls. Initially, the RadioButtons were dropdown menus that proved difficult to see all possibilities available at a glance. Those were pulled into ButtonGroups with labels to provide context to the possible types of questions, as well as the style of those questions. This had the added benefit of showing all available types of problems (whether topic or style) to the user without any additional exploration required.

Additionally, the placement of all controls and buttons, as well as their styles, were iterated over, including having student tutors examining the program to determine usage patterns. Knowledge from usability, interface, and data visualization texts by Schneiderman[6], Norman[7], Tufte[8, 9], et. al. were applied to make sure that the mutual exclusion provided by types/styles of problems was clear as well as having buttons near the functionality and actions they would activate upon use. Color-coding was used to more closely mimic the functionality a user might be accustomed to while using MATLAB. Specifically, any errors were colored to be red in the app and the error text from MATLAB was used directly.

Before settling on a final design, four engineering peer tutors (sophomore or junior level engineering students) tested the app (not all functionality was implemented at this stage) and provided feedback. The students played around with the app for approximately 10 - 15 minutes each and were asked to rate the overall appearance of the GUI on a five point Likert scale with 5 being very appealing and 1 being very unappealing and asked to rate the overall layout of the GUI with 5 being excellent layout and 1 being poor layout. There was also two sections where students could offer suggestions on how to improve the appearance and layout of the GUI. The appearance was rated as 4.0/5 and the layout was rated 5.0/5. Their comments are summarized below.

- Appearance is straightforward, use slightly larger text (mentioned in 3 out of 4), and abundance of unused space.
- Everything appears to be readily available, add a list of helpful functions in addition to syntax, allow user to scroll in the code results box if the answer is long, and make the problem say whether the user is right or wrong at the top of the code results box.

The feedback was incorporated into later versions of the app interface through larger text throughout and reducing larger gaps between controls after the resizing.

Interface and App Operation

As seen in Figure 1, the interface opens with all user options visible. A number of RadioButtons exist for the user to choose the type and style of question the user would like to study as well as Buttons to activate functionality, and text UI controls to enter or view text from the app. The RadioButtons are all arranged into ButtonGroups with each group labeled to describe their functionality. At the top-left, there is a group to choose which 1D array topic the user would like to attempt. The currently available topics are Creation, Indexing, Comparisons, and Operations. Below that is another group that allows the user to choose the style of question they will be presented; learning or practice problems. Learning problems come with associated helpful text that is visible once the question is displayed. This helpful text is shown under the button group for choosing the style of question in a text box UI control. On launch of the app, Array Creation, specifically of the Learning style, questions are selected and displayed by default.

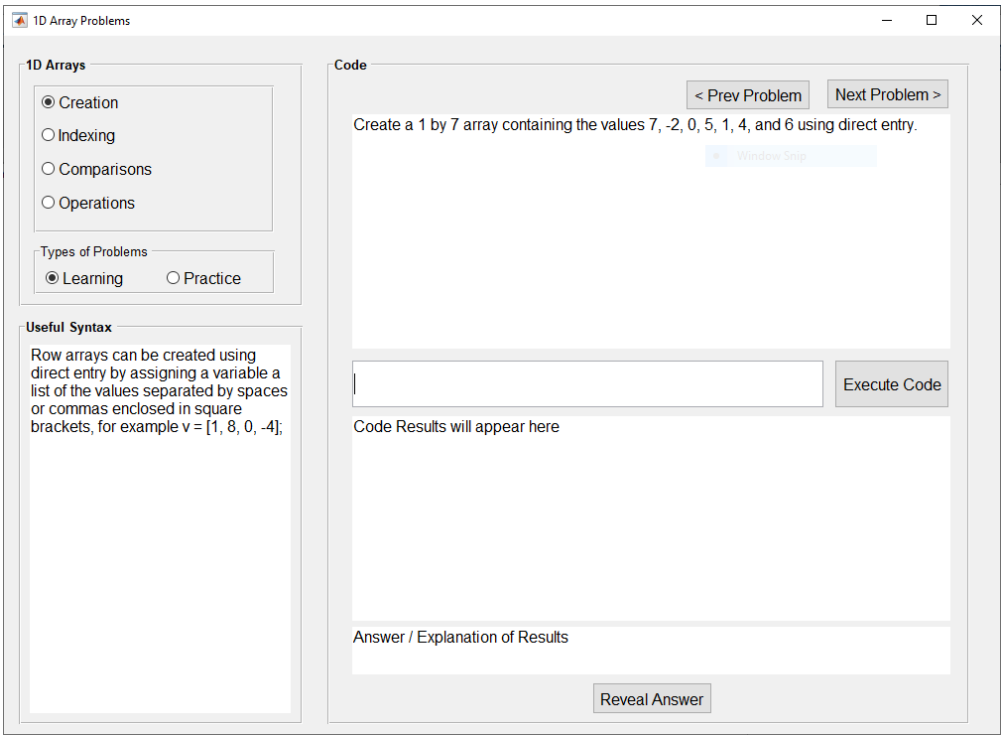

Figure 1. App Interface, Problem and Hint

The rest of the interface contains the items used to control most of the operation of the app. There are buttons to navigate through problems, a text box UI control that contains the current question, an area for the user to enter MATLAB code, a button to execute user-entered code, a text box to show the result of executing that code, and a text box where the question's answer can be displayed. Additionally, there is a button that allows the user to reveal the question answer without needing to answer the problem correctly. The correct answer is revealed automatically upon receiving a successful attempt from the user.

During a normal session, the user can move between questions with no restrictions. They can use learning questions with helpful text and attempt those or choose the practice style of question which provide no hints. Users can attempt answers with no navigation restrictions or can choose to go through questions and reveal the answer for each one as a means of study. This multitude of options allows the user to study the questions using whatever method is most comfortable for them. Since this is intended to be used as a supplement to MATLAB Marina[10] (our personally-developed Virtual Learning Environment), the user also has the option of watching videos, reading the provided text, or looking through existing code samples.

Questions, answers, helpful text, and other metadata about the questions are currently stored in a Microsoft Excel file and loaded on the app's startup. Using this data storage technique was chosen for the ease of adding questions for other people who want to attempt to use this program in their own classes with modifications. This was chosen in favor of a plain-text system that, while faster and less storage intensive, would have required a fair bit of documentation and standardization to use effectively. While the Excel format will still require documentation if opened up for other users, the authors felt that the spreadsheet format would be easier for others

to work with. It should be noted that this storage approach is currently being evaluated for possible later issues with scalability and program distribution. There is also consideration being made to provide a separate administrative GUI application to create, edit, update, and delete questions so that the user would not need to be familiar with how the data is stored for the program.

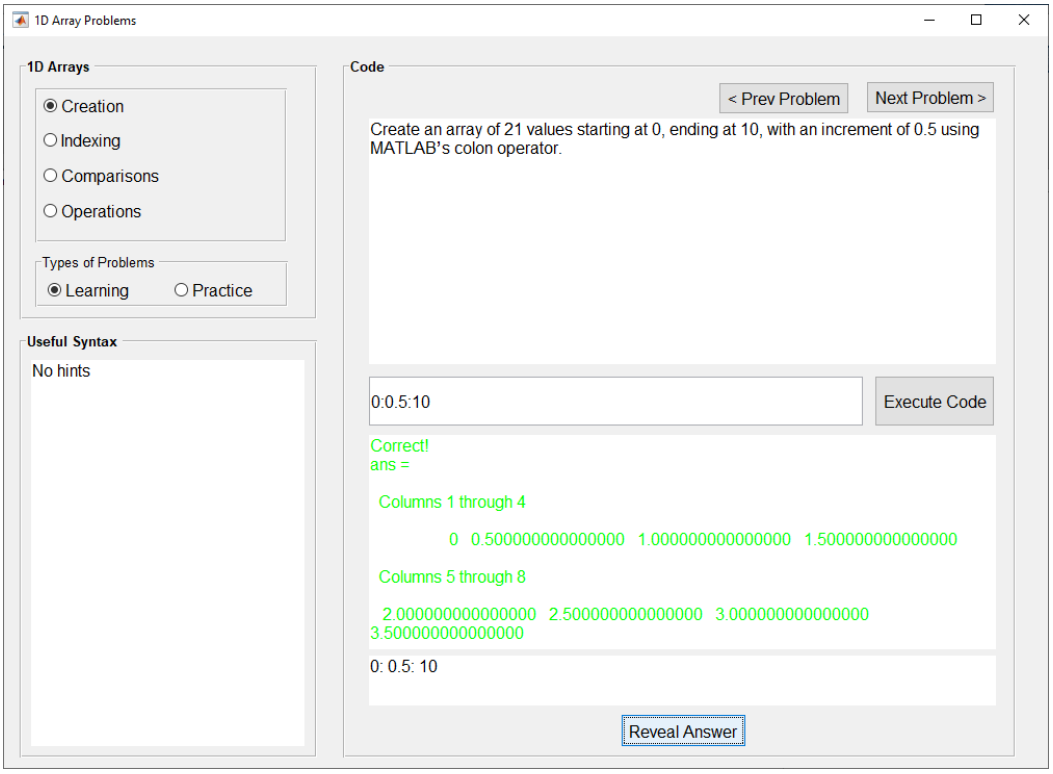

Figure 2. App Interface, Entered Statement, Verification of Result

Conclusions and Future Direction

The MATLAB app was developed to provide students with a way to practice working with arrays, see the results of array operations, and get immediate feedback. It is hoped that a better grasp of the array operations and their results will allow for students to apply these concepts/operations to solving for complex problems. The authors plan to have the app used by a sample of introductory programming course students in the electrical/computer engineering and mechanical engineering departments in spring 2020. The current version of the app is being used as a MATLAB function/GUI and requires MATLAB to be installed on the computer the application is run on. A stand-alone MATLAB app will be created for distribution to classes.

MATLAB adopted a different system for creating GUI applications in R2016a named App Designer. They currently plan on removing GUIDE but have not provided an exact timeline for that change only stating that it will be removed in a future release. Once that occurs, any existing GUIDE applications will not be able to have their interfaces edited through the currently existing tools. Because of this, initial examination of moving to an App Designed-based application have begun. The GUI interface does not appear to have any problems being

automatically migrated, but the functionality is not automatically converted and will require additional development work. Additionally, the feasibility of porting the application and any work that may need to happen to update application behavior has not been examined. The current plan is to migrate the application and rewrite the functionality to work with the newer design system for future maintenance and additional functionality.

The current version of the application has been designed around the idea of flexibility, reuse, and modularity. With the current question setup, adding additional questions, or topic categories, is not difficult with only minor modifications to the user interface. Adding question styles could easily be done by moving the horizontal radio buttons to be vertical, although the current interface is already a rather "tall" application. Considering the possible use of smaller resolution screens, growth in height without some restrictions would be undesirable. Finally, clutter must be considered when trying to extend functionality as different content topics might require additional items in the GUI that are not currently present. There must be some examination done when adding sections to avoid the "one-size-fits-all" style of thinking that can cause issues with interface design and organization.

References

1 James H. McClellan, Ronald W. Schafer, Jeffrey B. Schodorf, and Mark A. Yoder, "Multi-Media and World Wide Web Resources for Teaching DSP," 1996 IEEE International Conference on Acoustics, Speech, and Signal Processing, Atlanta, GA, May 1996

2 Educational MATLAB GUIs, http://dspfirst.gatech.edu/matlab/, Jim McClellan, Georgia Institute of Technology

3 Gabriel Heredia Acevedo, Bernardo Restrepo, and Jonathan Holguino, "MATLAB GUI for Elementary Flows as an Educational Tool," 2015 American Society of Engineering Education Southeast Annual Conference, Gainesville, FL, April 2015.

4 Lina Battestilli, Larry Silverberg, Jeffrey Eischen, and Caleb Thomas, "CADApps: MATLAB Apps for Core Courses in Aerospace and Mechanical Engineering Curricula," 2019 American Society of Engineering Education Southeast Annual Conference, Raleigh, NC, March 2019.

5 Subhan Khan, Mujtaba Hussain Jaffery, Athar Hanif, and Muhammad Rizwan Asif, "Teaching Tool for a Control Systems Laboratory Using a Quadrotor as a Plant in MATLAB," IEEE Transactions on Education, vol. 60, issue 4, pp. 249 – 256, November, 2017.

6 Ben Shneiderman, Designing the user interface (2nd ed.): Strategies for effective human-computer interaction, Addison-Wesley Longman Publishing Co., Inc. Boston, MA, 1992.

7 Donald Norman, The Design of Everyday Things, Doubleday, New York, 1990.

8 Edward Tufte, The Visual Display of Quantitative Information, Graphics Press, Cheshire, CN, 2001.

9 Edward Tufte, Envisioning Information, Graphics Press, Cheshire, CN, 1990.

10 Goeser, P.T., Murphy, T., and Williams, C., "A Virtual Learning Environment as an Open Educational Resource for an Engineering Programming Course", University System of Georgia Teaching and Learning Conference, Athens, GA, April 5 – 7, 2017.

Thomas Murphy

Thomas Murphy is an Associate Professor of Electrical Engineering at Georgia Southern University. He received his PhD and M.E. in Electrical and Computer Engineering from the University of Florida and his B.S. in Electrical Engineering from the University of Notre Dame. His research interests are primarily in the areas of digital systems, control systems, signal processing, engineering education, and computer applications in these areas. Dr. Murphy is a

member of several professional societies including The American Society for Engineering Education (ASEE) and The Institute of Electrical and Electronics Engineers (IEEE).

Christopher Williams

Christopher Williams is a Lecturer of Computer Science at Georgia Southern University. He received his M.S. and B.S. in Computer Science from Armstrong Atlantic State University. His research interests are primarily in the areas of mobile computing, usability, and education. Mr. Williams is a member of the Association for Computing Machinery (ACM).