Utilizing the Full Range of MATLAB Capabilities in the Classroom

James E. Toney and Adithya Jayakumar
Department of Engineering Education
The Ohio State University
Columbus, OH 43210
Email: toney.35@osu.edu, jayakumar.5@osu.edu

Abstract

Since its origin as a platform for performing matrix calculations efficiently, MATLAB has developed into a sophisticated general-purpose programming language. It is now the platform of choice for introducing engineering students to computer programming, due to its flexibility and ease of use. Recent enhancements to MATLAB capabilities have created new opportunities for improving the efficiency of classroom instruction and for promotion of active and mastery learning. In this paper we discuss our experience in adopting a number of recent MATLAB enhancements in a second-year-level programming course for engineers at Ohio State. MATLAB Grader is used for automatic submission and grading of basic in-class exercises, while MATLAB Online's folder-sharing capability serves as a vehicle for submission of assignments involving graphical user interfaces or other interactivity. For more extensive assignments that are not amenable to auto-grading, we have developed custom checker codes for students' use in validating their code prior to submission. Interactive live scripts, in tandem with the Top Hat classroom platform, have supplanted static Powerpoint presentations to increase student engagement.

In conjunction with these innovations, the course content has been re-designed to emphasize a more mature approach to application development based on modular programming. Increased emphasis has been placed on leveraging MATLAB's unique capabilities for array processing, as well as less commonly used data structures, including cell, structure, and string arrays.

Introduction

This paper discusses the design of a second-level MATLAB programming course for engineering majors who have had previous exposure to the subject in the first-year program. The students are nearly evenly divided among sophomores, juniors and seniors, with a small number of freshmen in the spring semester. The objective of this course differs from most of those that have been described in the literature, which typically fall into three categories:

- using MATLAB simulations as a vehicle to teach an engineering subject, such as signal processing or mechanics[1,2]
- introducing basic concepts of programming using MATLAB as a foundation for further computer science courses, which may utilize C++ or Java[3]

- combining an introduction to programming with coverage of MATLAB's numerical analysis capabilities for linear algebra, differential equation solving, etc.[4]

This course, by contrast, is a terminal programming course for students in less computation-focused engineering fields. Its emphasis is on general programming, rather than numerical analysis using built-in functions.

All engineering majors at Ohio State are introduced to programming in MATLAB as part of the first-year Fundamentals of Engineering sequence. The honors version of the first-year course progresses from MATLAB to C / C++ programming, which satisfies the computer science requirement for an engineering degree. Non-honors students must take an additional programming course; computer science & engineering (CSE) majors typically take C++ or Java, while students in less computer-oriented majors may take Programming in MATLAB for Engineers. Consequently, the majority of students in the latter class are civil, chemical, or aerospace engineering majors, with most of the remainder being in welding engineering, food, agricultural, and biological engineering, environmental engineering, or biomedical engineering. Relatively few electrical and computer engineering (ECE) or mechanical engineering majors take the course, and virtually no CSE majors.

In 2018 the course was re-designed to place greater emphasis on general principles of structured programming, increase the use of active and mastery learning, and leverage recently released MATLAB capabilities. The revised learning objectives are stated as follows:

Upon successful completion of the course, students will be able to:

- Design, implement and debug a moderately complex MATLAB program using a modular approach
- Incorporate universal structures such as branching and looping into programs
- Utilize MATLAB-specific features to perform operations on large data sets efficiently

In conjunction with restructuring of the content, the pedagogical approach was revised to increase emphasis on active and mastery learning and to use classroom time more efficiently. In accordance with a modern "flipped classroom" model[5], students watch lecture videos before class; in class they answer review questions, follow along with demonstration examples, and complete programming exercises.

Structure of the Course Content

Since the programming background of the students varies widely – some have had only the Fundamentals of Engineering course, while others have used MATLAB extensively in major courses – the course begins with the most basic elements of MATLAB. The course content has the following major components:

- Array operations and data analysis
- Input/Output and plotting
- User-defined functions and program structure
- Elements of programming (loops, branching)
- Data structures
- Graphical user interfaces
- Simulation

Compared to the previous edition of the course, increased emphasis has been placed on modular programming, in contrast to unstructured, linear scripting. There is also increased coverage of less commonly used data structures, including string, cell and struct arrays. The last quarter of the course deals with discrete-time simulation, primarily applied to kinematics, with the last day providing a rudimentary introduction to Simulink.

Major changes from the previous edition of the course include:

- Moving the coverage of relational and Boolean operators, logical arrays and logical indexing much earlier. This change was motivated by our experience that if students are taught to iterate over data with loops and `if` statements first, they never become completely comfortable with MATLAB array operations and tend to fall back on the less efficient approach.
- Introducing user-defined functions earlier and expanding the coverage of program structure. In the past, when this topic was introduced much later, students became habituated to unstructured, linear scripting. They tended to regard user-defined functions as a special technique to be used rarely, rather than a fundamental approach to designing programs.
- Adding a unit on debugging and testing. Previously, students were not formally introduced to the debugger, and no emphasis was placed on testing of code. This left them inadequately prepared to construct and debug the final team project.
- Greater emphasis on problem solving techniques of relevance to engineering, including iterative solution of mechanics problems and cellular automata algorithms, exemplified by Conway's Game of Life.[6] This increases the integrative nature of the course, with mutually reinforcing computer science and engineering science content.[7]
- Expanded coverage of `char`, `string`, `cell`, and `struct` arrays. Since string arrays were introduced into MATLAB recently, there is still much confusion about the difference between a `char` array and a `string`, and when it is best to use a `cell` array of `char` arrays to represent strings, rather than simply a `string` array. The course materials and assignments have been revised to clarify these issues.

Structure of Activities and Assessments

The latest version of the course has a multitude of activities and assessments geared towards active learning and maximizing class participation. They are as follows:

- Pre-class videos
- Student polling
- In-class examples
- Exercises
- Application Assignments
- Projects
- Exams

Of the above, the Pre-class videos and In-class examples don't contribute to the students' grade and only serve as a mechanism to convey the content. An overview of each is provided below:

Pre-class videos: The pre-class videos were incorporated into this course to maximize the amount of class time where students could work on programming and problem-solving tasks. This 'flipped' classroom approach provided more time for examples and in-class exercises, while getting rid of the traditionally long presentation going over the subject matter. This also means that students have the opportunity to get more questions answered immediately. Each class typically has 2 to 4 short videos which should take students about 15 minutes in total.

In the interest of not re-inventing the wheel, high quality videos readily available through Mathworks were used. The authors plan to create customized videos for more advanced topics such as discrete- time simulation.

Student polling: The Top Hat platform is used to pose review or checkpoint questions. Students are typically allotted 1 minute to complete each question, which can vary depending on the complexity of the problem. Results of the polling are then shared followed by a quick discussion about the correct answer. When there is a significant diversity of answers, students are given one minute to discuss the problem with their neighbors. Those who are persuaded by their peers that their initial answer was incorrect may change their response, as in Mazur's peer instruction approach.[8]

In-class examples: MATLAB Live scripts containing brief text explanations, explanatory figures and code examples have largely supplanted the static Powerpoint presentations that we used previously. Live scripts are a new addition to MATLAB which enable the integration of text, images and equations into MATLAB script. Students download the live script from the LMS and can run the ready-made examples along with the instructor and observe the output. The files also include 'You Try It' sections, each of which gives students an opportunity to try a practice problem themselves before the instructor provides the solution.

Exercises: Following the in-class examples, students work on the for-credit exercises. The exercises are typically 2–5 short, application-centered questions, where students are required to write the correct lines of code to solve each question. The majority of students are able to complete the exercises within class, but they can be submitted until the beginning of the next class without penalty, and until the next exam with a 20 % late penalty.

Students complete the exercises using MATLAB Grader, another recent add-on to MATLAB. MATLAB Grader is an online tool, originally developed for use in massive open online courses (MOOCs), which enables instructors to create their own questions and have students answer them using an online interface. The system provides the opportunity for Mastery, as it allows students to repeatedly try until they enter the correct solution.

Application Assignments: The application assignments are weekly programming assignments that cover the week's topic in more depth than the exercises. Since the students in the class are from a wide range of majors, the topics covered by the applications are also diverse. This type of assessment is needed to further reinforce the material that was covered and requires students to think about material in a broader sense.

The application assignments are done in MATLAB and students submit their script file (.m file) and the output of the program (command window messages and plots). These assignments, unlike the exercises, are graded by the teaching assistants.

For application assignments that deal with graphical user interfaces (GUI), an alternate submission process was devised using the folder sharing capability of MATLAB Online. After developing their program on the desktop using GUIDE, students upload their code and figure to a folder on their MATLAB Drive, which they then share with the teaching assistant. The TA can evaluate their program by running it, without the need to download it.

Projects: There are 2 major projects that are a part of this course – the mid-semester project and the final project. The mid-semester project is an individual project in which students visualize and analyze a large, real-world data set. They do this by performing statistical analysis on the data and figure out appropriate ways to visualize the information through different plots.

The final project is a significantly more complicated team project. In this project, teams model the take-off and landing phases of the Falcon 9 rocket. Prior to the start of the final project, students are introduced to discrete time simulation and spend three weeks modelling systems with gradually increasing complexity. While certain assumptions are made to make this a feasible task for students to complete in 3 weeks, the project serves as a comprehensive and summative assessment of the entire course.

In the final project, students use material from the entire course and are encouraged to look into commands, in-built functions and techniques that were not covered in the class. They create multiple user defined functions to determine various physical parameters such as temperature and pressure at various altitudes. They also create a Graphical User Interface (GUI) of the entire system, plotting useful graphs from both the take-off and landing phases. The deliverable of this project includes a report, with an extended abstract, pseudocode or flowchart, results and conclusions. Students also submit their GUI and all code related to the project for testing.

Exams: The course has 2 Midterms which require students to create appropriate script files to solve 'word problem' type questions. These questions mimic problems that students are likely to encounter while using MATLAB in their other classes or in industry. They typically involve extracting a subset of data, perform some mathematical analysis on it using the programming techniques that they have learned in the class, visualize the data using appropriate graphs and plots and write the conclusions to a file. The second midterm tests students understanding of certain advanced MATLAB components such as GUIs and problem-solving methods such as interpolation etc.

Tools and Resources

The students who are a part of this course have access to an abundance of additional tools and resources to help with their overall learning process.

Instructional Team: The instructional team is a core part of the support structure for this course. For a class of 36 students, there are 2 undergraduate teaching assistants (UTAs) and one instructor. There are 6 to 7 sections of this course offered every semester. The instructors and the TAs answer questions in class, on the discussion board, through email and during office hours. Students have the opportunity of visiting the office hours of any of the 12 to 14 UTAs in addition to the office hours of the instructor. This means that students can get help on their application assignments or have their questions answered at most times during the week.

Learning Management System: This course has a learning management system (LMS) which houses all the course materials such as the syllabus, presentations and videos for each topic, application questions and associated checkers, access to additional resources, a calendar of things to come and discussion board for each topic. In addition, students use the LMS to submit their application assignments, complete periodic journals and see their gradebook. Instructors can also post announcements on the LMS to alert students about any changes in the material or remind them about upcoming deadlines.

Custom Checkers: Prior to the release of MATLAB Grader, custom checker scripts were developed that students could use to validate their code prior to submission. The checkers are still used for the application assignments, but only as an aide for the students, not for grading. As in the case of all auto-grading schemes, constraints (such as precise variable names and the argument list of functions) must be placed on students' code. In the early assignments, templates are provided to help students comply with the checker's requirements.

Since the checkers are only for validation, not grading, their use is optional. However, our experience has been that most students take advantage of the checkers to validate their answers, despite the inconvenience of having to perform an additional step and accepting constraints on their code.

Conclusions

This paper describes changes to "Programming in MATLAB for Engineers", a terminal programming course for students in less computer-oriented engineering majors, such as civil, chemical and aerospace engineering. Adoption of the latest capabilities from Mathworks – live scripts, MATLAB Grader, MATLAB Online folder sharing, tutorial videos – has improved the efficiency and effectiveness of instruction in our second-level MATLAB programming course. Additional resources – Top Hat polling, custom checker scripts – have been implemented to improve classroom interactivity and provide rapid feedback to students. Changes in the structure and content of the course have been made to increase the emphasis on structured programming and better prepare students to design, implement, and debug complex programs. The initial offering of the revised course is in progress during the Spring 2019 semester.

References

1. U. Rajashekar, A. C. Bovik, "Interactive DSP education using MATLAB demo", *1st Signal Processing Education Workshop*, Hunt, TX, Oct. 15-18, 2000.

2. S.H. Song et al., "Developing and Assessing MATLAB Exercises for Active Concept Learning," *IEEE Trans. Education* 62 (1): 2 – 10 (2019).

3. A. Azemi and L. Pauley, "Teaching the Introductory Computer Programming Course for Engineers Using MATLAB," 38th ASEE/IEEE Frontiers in Education Conference, Saratoga Springs, NY, Oct. 22-25, 2008.

4. M. Herniter, D. Scott, and R. Pagasa, "Teaching Programming Skills with MATLAB," Proceedings of the 2001 ASEE Annual Conference and Exposition, 6.954.1-9.

5. R. Talbert, "Learning MATLAB in the Inverted Classroom," *Computers in Education Journal* 23(2):89-100, 2013.

6. M. Gardner, "Mathematical Games – The fantastic combinations of John Conway's new solitaire game "life"", *Scientific American*. 223 (1970), 120–123.

7. L. Pruski and J. Friedman, "An Integrative Approach to Teaching Mathematics, Computer Science, and Physics with MATLAB," *Mathematics and Computer Education,* ISSN: 0730-8639 (Winter 2014), 6 – 18.

8. C. Crouch and E. Mazur, "Peer Instruction: Ten Years of Experience and Results," *Am. J. Phys*. 69 (9), 2001, 970-977.