Reflections on Integrating MATLAB Grader across a Mechanical Engineering Curriculum

Dr. Patrick M Comiskey, Milwaukee School of Engineering

Patrick Comiskey is an Assistant Professor of Mechanical Engineering at the Milwaukee School of Engineering. He received his B.S. from that institution and his Ph.D. from the University of Illinois at Chicago, both in mechanical engineering. His teaching and research interests are in the area of transport phenomena and engineering education.

Dr. Prabhakar Venkateswaran, Milwaukee School of Engineering

Prabhakar Venkateswaran is an Associate Professor of Mechanical Engineering and the Department Vice Chairperson at the Milwaukee School of Engineering. He received his Master's and PhD in Aerospace Engineering from the Georgia Institute of Technology, and his Bachelor of Science degrees in Mechanical and Aerospace Engineering from the University of Miami, Coral Gables. His teaching and research interests are in the area of thermal-fluid sciences, gas turbines, gas dynamics, and engineering education.

Dr. Michael Christopher Sevier, Milwaukee School of Engineering

Michael Sevier is currently an assistant professor of Mechanical Engineering at the Milwaukee School of Engineering. After finishing his doctorate degree at the University of California, Santa Barbara, Michael took a position at ATA Engineering where he worked as a structural analysis engineer for nine years. During this time, he both took and taught multiple professional courses and realized how many technically brilliant instructors struggled to convey information in a way that could be readily absorbed by the students. Now in his eighth year in academia Michael is researching how various teaching methods and study habits affect the absorption and long-term retention of class material in the hopes of best preparing students for their future as engineers.

Reflections on Integrating MATLAB Grader Across a Mechanical Engineering Curriculum

Abstract

In this paper, the authors document their experiences implementing MATLAB Grader in several mechanical engineering courses spanning from 1st year core courses to 4th year electives at Milwaukee School of Engineering, a primarily undergraduate institution (PUI). MATLAB Grader is an online browser-based version of MATLAB where instructors can create, share, and automatically assess MATLAB exercises with their students. The sustained effort by the authors to incorporate MATLAB Grader in their courses was motivated by an intent to help students develop an analytical mindset so that they can ultimately successfully engage in more realistic design-type problems. MATLAB Grader facilitated this goal by presenting students with opportunities to analyze trends with parametric variations as opposed to focusing on a single answer for one set of conditions. In addition, problems can also be scaffolded so that students receive instantaneous feedback on intermediate stages of a complex problem. Furthermore, the authors have often grappled with how to best grade programming-based assignments. MATLAB Grader is an excellent platform that addresses these objectives and desires.

This work is unique in that, to the best of the authors' knowledge, the use of MATLAB grader in mechanical engineering courses is not well-documented. There is more evidence of its use in the electrical/computer engineering discipline, but evidence of its use in mechanical engineering is more limited. As a result, this paper represents an effort to share the pedagogical and practical benefits as well as the challenges and drawbacks of using this platform. Based on their experiences, the authors also seek to share guidelines that they have developed for instructors considering the use of MATLAB Grader in their mechanical engineering courses. These guidelines are tailored to the cohort of students i.e. 1st year, 2nd year, etc. as well as the type of course taught i.e. programming course versus a problem-solving course like statics/dynamics versus a numerical methods course such as finite element methods.

Background

There are several merits to integrating computational activities into an engineering curriculum. Computer programs can help automate repetitive tasks, visualize important trends, analyze large data sets, perform parametric sensitivity studies to support design decisions, and students cultivate a logical problem-solving process. Furthermore, as the industrial landscape continues to rapidly advance with increasing penetration of automation and big data analysis, students will need to graduate ready to utilize modern tools in their engineering practice. In

mechanical engineering courses, computer-based assignments have significant potential to increase experiential learning for students. For instance, in thermodynamics, instead of a student analyzing a Brayton cycle at a fixed set of conditions to generate one set of values for the power output and efficiency, they can repeat the analysis for multiple compressor pressure ratios to analyze its impact on the power output and efficiency. Or in dynamics, instead of calculating the range and maximum height of a projectile, they can plot and visualize the trajectory of the projectile for various initial conditions.

While there are significant advantages for helping students develop fluency using computer programs to solve engineering problems, there are some challenges to consider. For instance, when teaching computer programing to first-year engineering students, the class sizes are typically large, making it difficult to provide timely and relevant feedback. For second- and third-year students, the computer assignments should help students explore the relevant engineering concepts in more detail, as opposed to focusing on the coding aspects. In upper-level numerical methods courses such as Finite Element Analysis (FEA) and Computational Fluid Dynamics (CFD), programming-based assignments can provide students with valuable insight into what is physically happening inside the black-box so they are not just learning to push buttons within the graphical user interface (GUI). Some of these issues can be avoided by an automatic grader with careful scaffolding and assessments. Considering these potential issues, MATLAB Grader presents some unique advantages. MATLAB is a ubiquitous programming environment that students are exposed to in many engineering colleges and programs. Furthermore, MATLAB is also utilized in many industrial settings. MATLAB Grader is an online browser-based version of MATLAB where instructors can create, share, and automatically assess MATLAB exercises with their students. MATLAB Grader can also be integrated into various Learning Management Systems (LMS) so that students do not need to navigate several different websites and grades are automatically input into the gradebook.

The primary advantage of the MATLAB Grader platform is its automated grading feature. Autograding of computer programs have been available and investigated for many decades, primarily in the computer science domain [1]. However, the use of autograders of computer programs in other disciplines has only received attention more recently. This is attributed to the different goals of autograders in computer science and other engineering disciplines. In computer science, the focus of the autograders is on code correctness, quality, and efficiency. In other engineering disciplines, the primary focus is to reinforce course concepts and develop modeling skills with code quality being a secondary concern [2].

The problems developed in the platform can be set up so that students have multiple attempts to correctly solve the problem. In addition, problems can also be scaffolded so that students receive instantaneous feedback on intermediate stages of a complex problem. These two

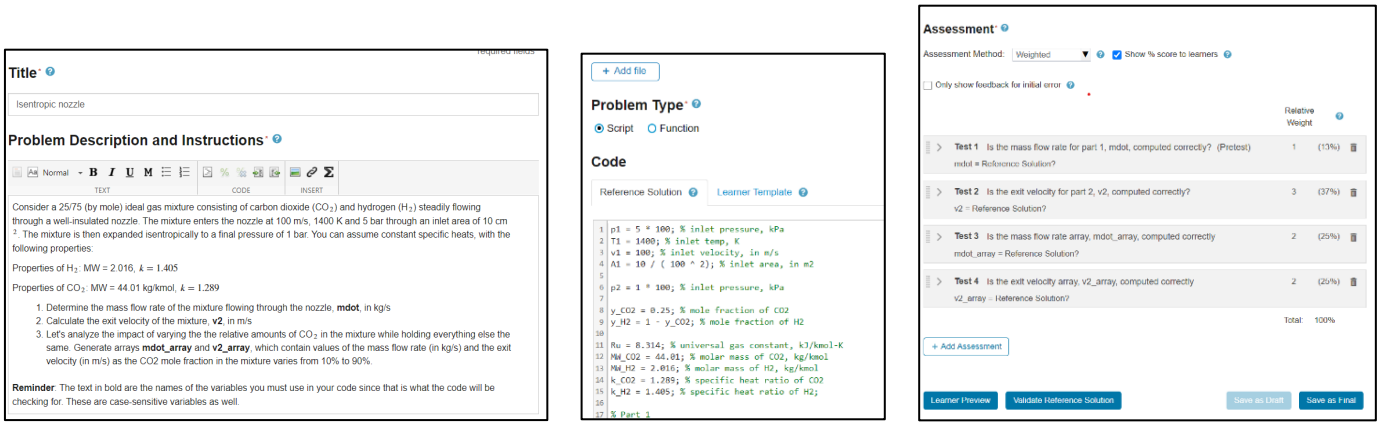
features in conjunction create a permission structure for students to learn through failure without worrying about negative impacts on their grade.

The automated grading feature of MATLAB Grader will free up time for instructors to work with students to help them understand the underlying concepts and debug their code. In addition, automated grading can ensure uniform grading and feedback for all students. Moreover, by carefully selecting the intermediate assessments for each problem, students may be encouraged to correctly solve complete parts of a problem. There is a lot of discussion around the concept of partial credit in engineering courses with many different philosophies [3], [4]. Simplistically, if a problem has n steps, and the student correctly performs k steps, then they would reasonably receive k/n percent of the total credit for problem [3]. In such a paradigm a student may potentially earn a high grade on an assignment or on an assessment without ever having solved a complete problem correctly because they were able to correctly work through a subset of the total number of steps. However, correctly solving a complete problem demonstrates a student's ability to accurately apply underlying principles and execute all phases of an analysis. MATLAB Grader problems can help achieve these outcomes because of the unforgiving nature of computing software; the answer is either correct or not with no in-between. For example, even if a correct equation is utilized, if the incorrect treatment of units results in an incorrect final answer, then MATLAB Grader can be set up to award no credit for that part of the problem. However, by carefully designing the intermediate milestones/checkpoints, the problem can be appropriately scaffolded so that students can utilize the instant feedback to determine the source of their error and rectify it before moving on to the next part of the problem. Furthermore, this practice can help students develop their own debugging/troubleshooting strategies which will help their general problem-solving abilities.

This paper describes the efforts by a group of mechanical engineering faculty at Milwaukee School of Engineering, a primarily undergraduate institution (PUI) to integrate MATLAB Grader based exercises throughout the undergraduate curriculum. The development and deployment of these exercises is described at various stages throughout the curriculum in addition to challenges and drawbacks encountered. All this is used to inform guidelines offered to other instructors interested in utilizing MATLAB Grader in their courses. While efforts to utilize MATLAB Grader are documented in the literature, they primarily focus on programming courses [5] [6], or in individual courses across different engineering and mathematics curricula [2] [7] [8] [9] [10]. To the best of our knowledge, this paper represents the first description of an intentional effort to integrate MATLAB Grader exercises across an undergraduate mechanical engineering curriculum.

Structure of MATLAB Grader

Any MATLAB Grader problem has two parts. The first is where the instructor develops the problem statement and instructions for the student (Figure 1 (a)), as well as supply the model solution (Figure 1 (b)). Below the model solution is where the instructor develops various assessments with corresponding comments to provide intermediate feedback should a student fail an assessment (Figure 1(c)).



(a) (b) (c)

Figure 1: Instructor view of a MATLAB Grader Problem showing the problem statement (a), model solution (b), and assessments (c).

The second part is the instructor view of the student-facing page which is where the students type out their code (Figure 2). On this page, the instructor can prescribe certain things which the students cannot change. For instance, if the problem asks students to generate an array of values corresponding to an array of input values, the instructor can prescribe the input array and lock that code so that a student cannot edit it. That way, if a student solves the problem correctly their output array should be identical to the one generated by the instructor.

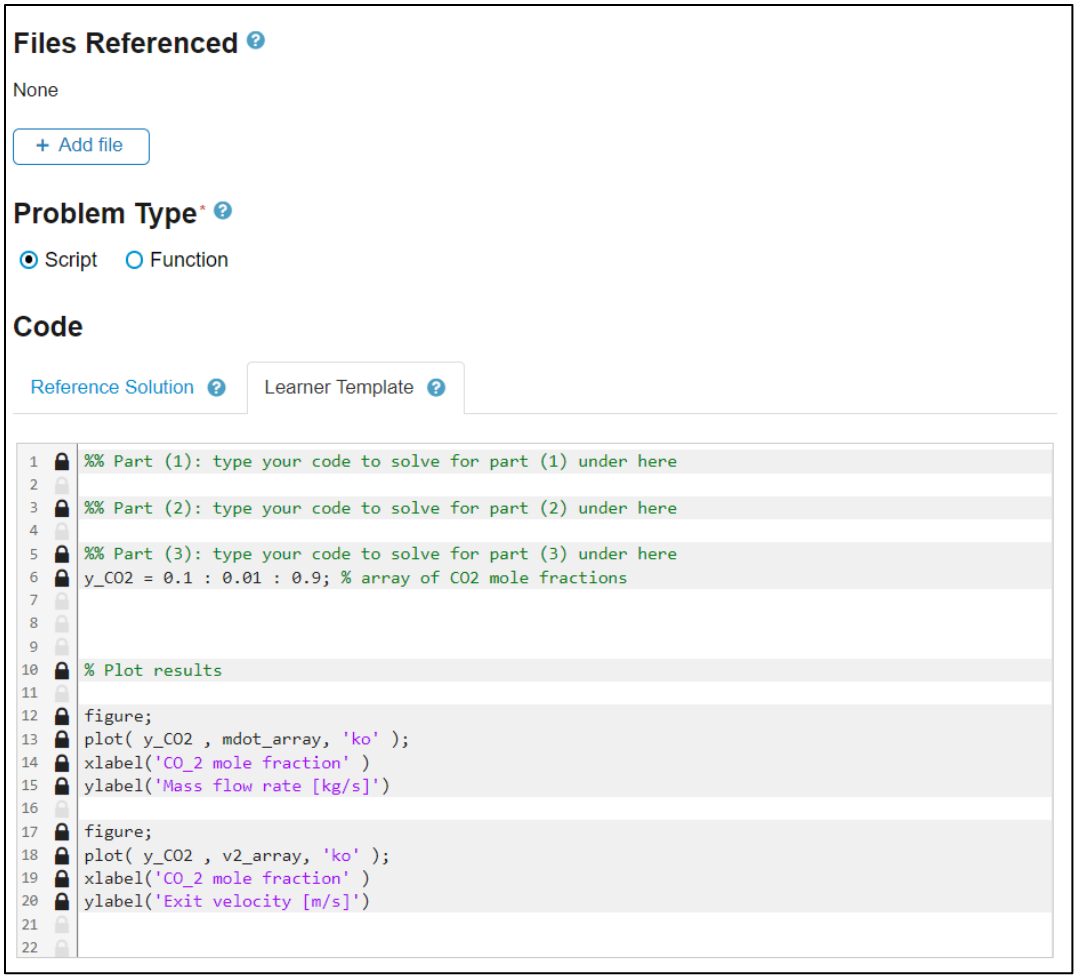


Figure 2: Instructor view of the student code template of a MATLAB Grader problem.

Problem Archetypes

MATLAB Grader offers two types of problems which can be assigned to students, script-based and function-based. Script-based problems are like traditional pen-and-paper problems, in that numerical values for inputs are prescribed and students input their code to generate the prescribed output variables. An example of such a problem is shown in Figure 1, where the desired output is an array because an input variable is being varied. The students are tasked with generating arrays of mass flow rate and exit velocities as the mixture composition entering a nozzle is varied. Setting up problems this way serves two purposes. First, it circumvents the issue of students completing the problem by hand and hardcoding the value to the desired output variable and defeating the purpose of the platform. Second, by having students generate arrays and then providing them with the code to plot these arrays (such as the locked code towards the bottom of Figure 2), students can start developing a “feel” for the subject matter by visualizing the sensitivity and directionality of output variables relative to changes in input.

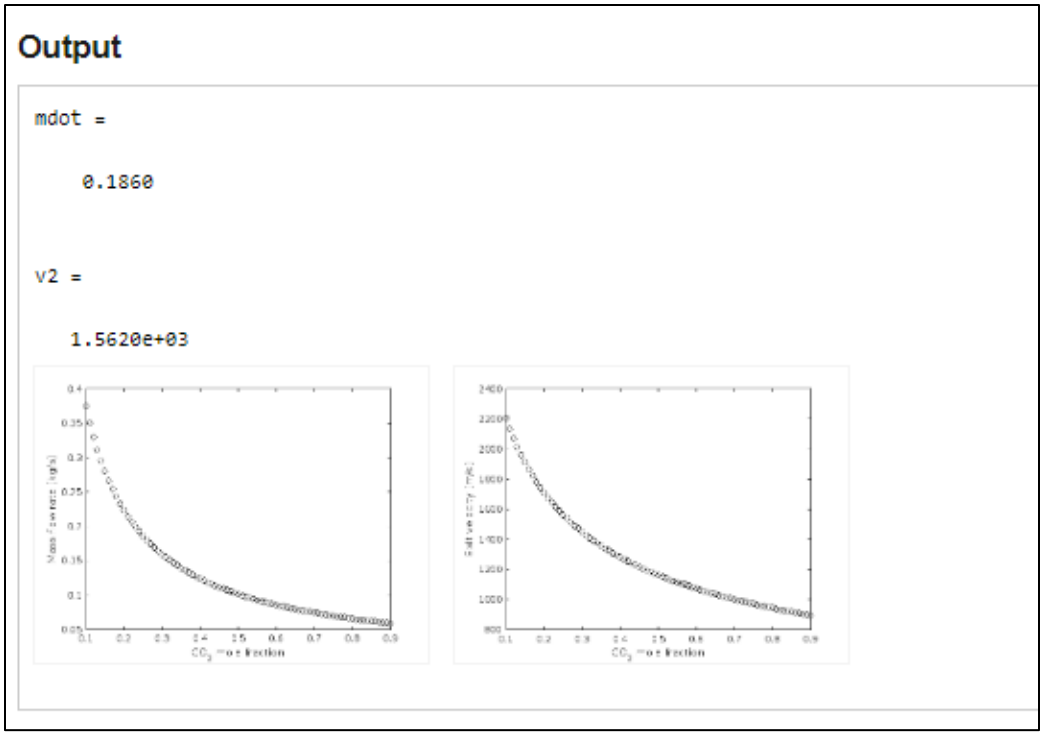
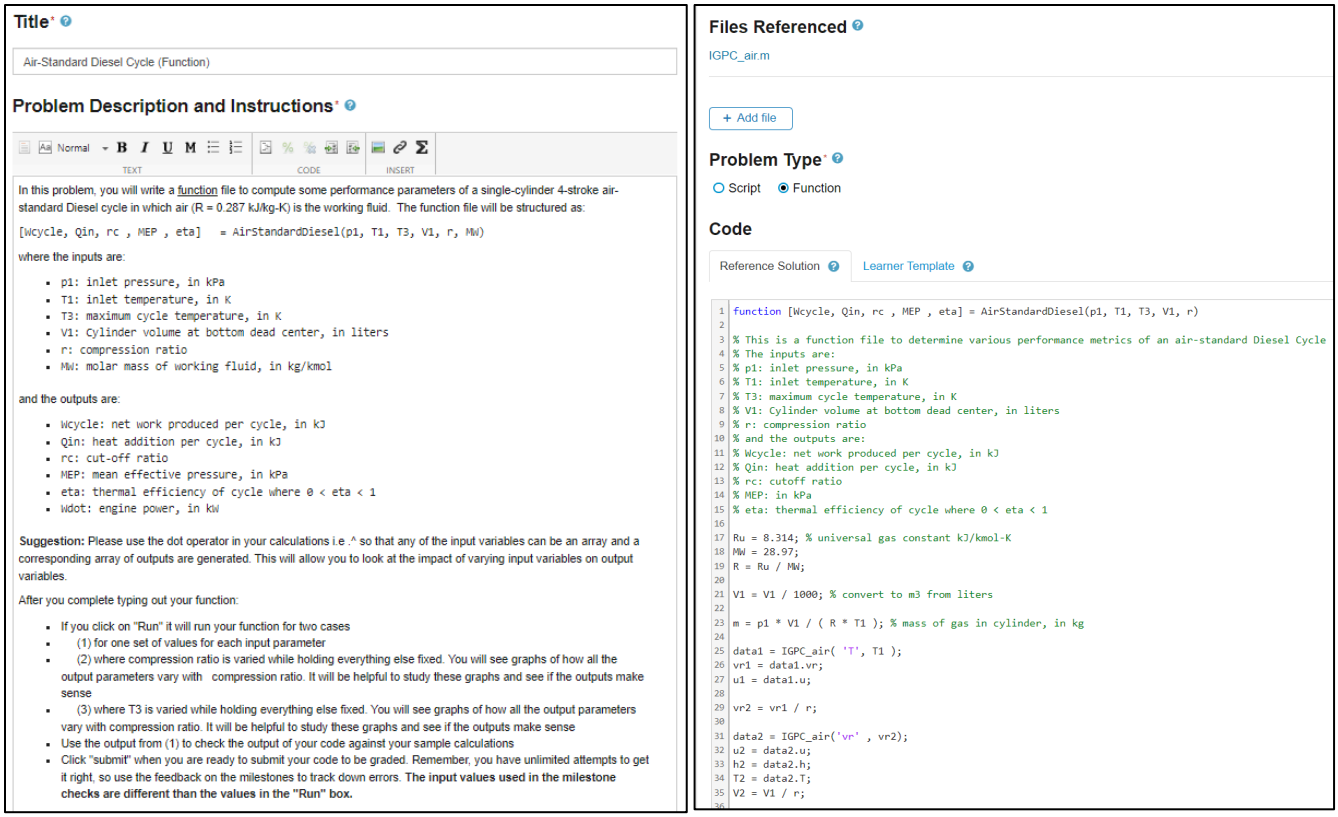


Figure 3: Student view of their code output when run through MATLAB Grader.

The sample output shown in Figure 3 represents the correct execution of a student script written to address the problem in Figure 1 with the provided code template shown in Figure 2. In addition to specific output values, a key output are the graphs generated to help students see trends and variations. This is a powerful way for students to attach meaning and understanding to equations and the subject matter in general.

In function-based problems, students develop function files which accept arbitrary values to a pre-defined set of inputs and generate a desired set of outputs. An example is shown in Figure 4. These problems are helpful in guiding students think about problems in a parametric fashion in which any input variable can be changed, rather than “hard-coding” values to certain parameters.



(a) (b)

Figure 4: Instructor view of a problem statement (a) and model solution (b) in a function-based MATLAB Grader problem.

Once the function is written, students are provided with a variety of test cases that they can run to check if their code is functioning correctly. This can be a single set of input parameters for students to debug their code by checking the output against sample calculations. Or similar to before, pre-written plotting code can also be supplied, which call the student's function to generate and plot the variation of outputs with variations in inputs.

Classroom Applications

The following are specific use cases to demonstrate how MATLAB Grader can be used to support student learning throughout their education starting with an introduction to programming course, proceeding through the core curriculum, and ending with upper-level computational and elective courses.

In the initial stage when students are learning programming logic, it is assumed that students are unfamiliar with coding and need significant practice to gain fluency. For courses within the core curriculum, students should have already been exposed to MATLAB and the goal of MATLAB Grader is then to reinforce their coding capability and apply that knowledge towards solving more complex problems as compared to those done on a calculator alone. Finally, the types of problems students encounter in upper-level computational and elective courses frequently require code to solve. MATLAB Grader not only helps facilitate this process,

but it can also support student-built functions to solve similar problem types (e.g., evaluating the response of any single degree-of-freedom spring mass damper system to any transient base excitation).

Computer Applications in Engineering

Computer Applications in Engineering is a required course for mechanical engineering students and is typically taken in their first year. The intent of the course is to familiarize students with computational tools, methods, and techniques, as well as their application towards the solution of practical engineering problems. This is accomplished by covering computer programming fundamentals such as writing pseudo-code and creating flowcharts before progressing to foundational logic like conditional statements and looping logic. To facilitate these topics, MATLAB is employed as the programming language of choice because of its robust integrated development environment (IDE), extensive documentation, relative ease of programming, and wide acceptance. In addition to computer programming fundamentals, hardware interfacing is covered as students learn how to collect and interpret sensor information and interact with various devices.

As this is an introductory engineering course, student preparedness varies widely depending upon content exposure in their secondary education. Some students also perceive the material as outside of their chosen major which may result in a lack of interest. Moreover, student reception of the material appears to be bimodal; they either greatly enjoy the course content or despise it. To add to the difficulty, student satisfaction of the material is not mutually exclusive to their performance in the course. All of this frequently leads to the unfortunate outcome of students who are not properly prepared to utilize the immensely important and relevant computational tools and applications in their engineering field of choice. The issue percolates to more advanced upper-level courses where numerical techniques are inherently required to adequately describe various physical phenomena or in laboratories where data collection hardware interfacing is paramount.

In an attempt to mitigate the difficulties of learning programming logic and therefore increasing content interest and ultimately course performance, MATLAB Grader was employed in two different instances of the course. The first was during the Winter 2021-22 term where the bulk of the homework consisted of MATLAB Grader problems (about four to six problems per week), but other homework problems were also assigned requiring in-person check-ins (one or two per week). The MATLAB Grader problems were used to build the conceptual framework for other, more difficult problems. Whereas requiring some homework problems to be reviewed in-person ensured that the student had to complete the assignment correctly in order to earn credit. This is similar to MATLAB Grader but with the added benefit of “forcing” student-instructor interaction to clear up confusion and prevent bad habits from developing. While this combination worked well for most students within the course, some found the assignment load to be overwhelming. Upon reflection, this issue could be mitigated by being more transparent with problem difficulty and creating more scaffolding for introductory MATLAB Grader problems.

The second instance of the course was taught during the Fall 2022 term where a MATLAB Live Script or active session of MATLAB was projected to the front of the class

during each lecture. This projection of MATLAB was used to actively teach students the course material in real time. After specific content was discussed, a "lecture coursework" topic was unlocked on the LMS, and in-class time was allocated so students could work through several MATLAB Grader problems. Students were allowed to submit their MATLAB Grader problems an infinite number of times and they were due at the start of the next class period. This afforded students the ability to engage in the course material immediately after it was presented to them and ask questions as they arose while still providing adequate time to complete should the student need it. In this manner, there were a total of 32 MATLAB Grader problems assigned and worked on during lecture throughout the term.

Due to the nature of the course material, the MATLAB Grader problems were relatively straightforward to create and setup. Several problems used were even written by MathWorks in a MATLAB Grader problem repository, although the majority of the problems were custom-tailored for each instance of the course. As most of the students were learning programming for the first time, it did not appear that there was a steep learning curve when comparing MATLAB Grader to the standard desktop-version of MATLAB. This was especially true for the students who previously had a background in programming. Furthermore, students appeared to rapidly comprehend new material, but it is unclear if that can be attributed to using MATLAB Grader. Upon reflection, the biggest drawback from a learner's perspective was separating MATLAB Grader technical issues versus programmatic flaws. For example, it was found that students often had difficulty with variable assignment, an issue compounded by the command "clear" behaving differently between MATLAB Grader and the standard version of MATLAB (see the discussion section herein). It is hypothesized that issues such as this occur because students are learning programming and MATLAB Grader concurrently and therefore simply have a difficult time separating the root of an issue.

Core Curriculum

Following the student's introduction to coding in "Computer Applications in Engineering," MATLAB Grader can be used within core mechanical engineering courses to reinforce their computational capability, visualize trends, and solve more complex problems. This section discusses the implementation of MATLAB grader within dynamics, mechanics of materials, fluid mechanics, heat transfer, and thermodynamics.

Dynamics and Mechanics of Materials

Dynamics and Mechanics of Materials are both second-year courses in the mechanics sequence with learning objectives that are primarily conceptual (e.g., applying principles of work and energy to hypothetical systems). For each of these courses, students are expected to draw free-body diagrams (and kinetic diagrams for dynamics) and write out their equilibrium equations. Difficulties observed in these courses primarily occur when a student skips steps or rushes the problem-solving process. The use of MATLAB Grader in both courses was intended to enforce the correct problem-solving process by only accepting correct results while giving tips where common errors are made. It was also intended to help reinforce basic coding skills such as

utilizing a for-loop to create an output array which could be subsequently plotted. For example, one of the Mechanics of Materials MATLAB Grader problems involved plotting the principal stress distribution through a short beam undergoing bending and transverse shear.

The major benefit of assigning MATLAB Grader problems in these courses was an increase in office hour attendance and student-instructor interaction. However, it quickly became apparent that many students continued to rush through the problem-solving process. It would not be uncommon for many students to skip the hand-written portion and go straight to coding. When errors occurred, students often approached the problem with a haphazard series of quick changes instead of slowly working through the process with incremental checks. For this reason, a checklist was provided to students hinting at the approach they should take. The solution process also needed to be clearly written on paper prior to asking an instructor for help.

One interesting observation was students insisting on using their calculator to solve problems as opposed to using MATLAB. Sometimes they would get the correct answer, but not in a computationally efficient way. The easiest method to mitigate this issue is to create problems which involve an array instead of a single scalar value. This would make the problem so computationally inefficient to do by hand that the student would need to use MATLAB. The side benefit of this approach is that the results are easier to view, and a relationship between variables may be observed.

It should be noted that students who had MATLAB Grader experience in their Computer Applications in Engineering course seemed to adjust more easily to the MATLAB Grader problems in their core curriculum courses that also contained MATLAB Grader and hence had a more positive learning experience. It is believed this is primarily because the automated feedback allows the student much quicker iteration through many more problems. It may also be that the emphasis on pseudocode prior to coding in those courses may translate into better problem-solving approaches within their core curriculum courses. Similar to the use of MATLAB Grader in “Computational Applications in Engineering”, it is likely worthwhile to be very careful and transparent with problem difficulty and incorporate appropriate scaffolding.

Thermodynamics, Fluid Mechanics, and Heat Transfer

Thermodynamics I is a core course in the curriculum typically taken by students in their second year, while Thermodynamics II is an elective course typically take in the 3rd year. Thermodynamics I is a traditional first course in mechanical engineering thermodynamics where the learning outcomes include applying 1st and 2nd laws of thermodynamics to closed and open systems, evaluating properties of real fluids, applying ideal gas and incompressible models to substances, and analyzing basic versions of power and refrigeration and heat pump cycles. Thermodynamics II is a continuation where concepts introduced in Thermodynamics I are applied to more advanced power and refrigeration/heat pump cycles, ideal gas mixtures, psychrometrics applications, reacting flows, and flows where compressibility effects are significant.

The Spring 2024 term is the first semester in which MATLAB Grader problems have been used in this course in Thermodynamics I. In this course, the goal of the MATLAB Grader problems is to help students continue to appreciate the utility of computing to automate repetitive calculations, and to help foster discipline when handling units. As a result, each weekly problem set typically contains one MATLAB Grader problem to supplement the written problems, and the problems are designed to gradually rise in coding complexity. Since this is often the first course where students must work extensively with English customary units, many of the initial problems focus on working in that system of units. The problems at the beginning of the term are relatively basic and designed to help students visualize individual processes and analyze individual steady-flow devices, while continuing to gain comfort with the MATLAB environment. Property evaluation using steam tables are still done using hardcopy versions of the tables to allow students to develop their property evaluation skills. In the second half of the course where various cycles are discussed, the problem complexity increases and are designed to help students appreciate the sensitivity of overall performance parameters to changes in the operating parameters. For instance, one problem is designed to help students explore how varying the pressure ratio in a Brayton cycle affect the power output, required heat addition, and thermal efficiency. At this point students would have hopefully demonstrated mastery of utilizing the property tables, and so, MATLAB-based property evaluation functions such as XSteam [11] can be integrated into the Grader platform.

In Thermodynamics II, the problems start off at a high level of complexity since students should have already demonstrated their mastery of the foundational concepts and skills in the pre-requisite Thermodynamics I course. Similar to Thermodynamics I, one MATLAB Grader problem was assigned with each weekly problem set. Since the number of property evaluations increase when considering more complex cycles, ideal gas mixtures, and reacting mixtures, many of the problems utilize MATLAB-based property evaluation functions to avoid tedious lookups and interpolations. Problems focus more on analyzing the impact of gas mixture composition on device/system performance, inspecting the sensitivity of initial conditions and fuel composition on adiabatic flame temperatures while accounting for variable specific heats, and evaluating the characteristics of HVAC systems comprised of multiple psychrometric processes.

A suite of problems has also been created for the undergraduate fluid mechanics and heat transfer courses. The structure of the problems is similar to the ones designed for the thermodynamics problems. However, the fluid mechanics problems have not been deployed yet, and the Spring 2024 semester is the first time the heat transfer problems are being utilized. The heat transfer problems cover topics in steady-state and transient conduction, one-dimensional numerical conduction, free and forced convection, and radiation. The convection problems in particular were difficult to develop because of the fluid property evaluations that were required and the availability of multiple valid empirical correlations for Nusselt numbers [12]. As a result, a parallel effort to locate or develop MATLAB-based fluid property evaluation functions was also undertaken, and MATLAB Grader problems were written to accept answers with multiple valid correlations. Again, by eliminating the tedium of interpolating for fluid properties, the goal

was to create time and space for students to explore in more depth various heat transfer phenomena and concepts.

Upper-Level Computational and Elective Courses

Upper-level courses build upon the foundational material covered previously to address problems more similar to those in industry and/or academia. At this point, computational tools are not only valuable in their ability to demonstrate concepts but are often required to solve problems. It is imperative to ensure that students use computational tools appropriately since there is a tendency to blindly trust whatever results the computer gives. It is especially important to address haphazard approaches to computing at this stage because they will soon be solving problems with real consequences. For this reason, MATLAB Grader is especially useful because it enforces correctness before credit is given. The scaffolding of the code helps students work their way through complicated algorithms, and the instantaneous feedback helps nudge them in the right direction. For these courses, another substantial benefit to using MATLAB Grader which is that the problems can more easily relate to experiments and/or commercial simulations, giving an added dimension to the student's learning.

Finite Element Analysis

The "Introduction to Finite Element Methods" course is primarily for fourth-year students and is designed to introduce fundamental theory and practical application via the use of commercial finite element analysis (FEA) software. Traditionally, this course would include hand-written assignments pertaining to theory and workshops to address aspects of the analysis workflow (i.e., pre-analysis, pre-processing, validation, and result-interpretation). The course concludes with a final project where student groups present their analysis process. Students are allowed to choose the topic for their final project with guidance from the instructor. Often these projects pertain to their capstone design courses or student clubs such as SAE formula hybrid.

MATLAB Grader was first introduced in Fall 2020 to the "Introduction to Finite Element Methods" course primarily for the purpose of simplifying the linear algebra computations previously performed on hand-written assignments so that students could more easily focus on the conceptual process therein [13] [14]. While this could have been done simply within the desktop version of MATLAB or another coding language, there is a danger that errors would be too difficult to diagnose by the student. This reduces the likelihood of connections the student makes between theory and practice. MATLAB Grader is advantageous in this regard because it provides checks that enforce correct output thereby ensuring a complete learning experience. At this point, each MATLAB Grader problem corresponded to a specific two-dimensional structure.

In Fall 2021, the existing suite of MATLAB Grader problems was dramatically expanded such that upon completion, the student would have created their own FEA code that could solve most two-dimensional static structural problems [10]. At the outset, students were provided a suite of background MATLAB scripts and functions that contained important functionality but

would be of little pedagogical benefit for students to create (e.g., formatting text and graphical output of results). Upon completion of each MATLAB Grader function, students were instructed to save their code within the same folder as the provided scripts and functions. The addition of each student-created function would expand the functionality of the code. For example, once the student completed the function for creating the stiffness matrix for a frame element, their code's capacity would then include the ability to solve problems with frame elements in addition to truss and spring elements. To run the code, students would first create “input files” via Excel workbooks that mimicked the form of Nastran input files. The students would then run a MATLAB Live Script that referenced both the input Excel file and the folder containing the necessary functions and scripts. The Live Script would then produce a text output file, contour plots, and output structures in the MATLAB workspace that provided both final results (e.g., nodal displacements) and intermediate information (e.g., the global stiffness matrix).

This problem sequence was intended to help students see the power and flexibility of the theory learned in class. When the code capacity increased to include plane stress isoparametric elements, the students could begin to see the resemblance between what they produced and results provided by commercial software such as Ansys. Another benefit of the two-dimensional FEA code was that simple models could be used to address FEA concepts such as convergence, singularities, and the consequence of poorly formed elements.

Results from term-end surveys in the Fall 2021 section of the course were mixed. Some students perceived this to be a positive learning experience while others found it to be overwhelming and/or irrelevant. One difficulty appears to be aligning assignments to a wide range of student motivation. For example, a graduate school-bound student would likely be very interested in learning how the coding process works since they may one day be creating their own subroutines. However, a student primarily concerned with becoming proficient with commercial FEA software would likely perceive MATLAB Grader problems as a burden and irrelevant towards their goal. It is also likely that the former student possesses greater coding proficiency which lowers the hurdle for completing the MATLAB Grader problems.

To mitigate this issue, subsequent terms have only assigned MATLAB Grader problems to students who are motivated to earn higher grades in the course. In the Spring 2024 term specifically, MATLAB Grader problems are only required for those wishing to pursue an “A” in the course since these students would likely gain the most from it. The additional effort would likely be unappreciated and less useful for other students whose time could be better spent producing quality workshops where commercial software is used to address given problem statements. Starting in the Spring 2024 term, all students have access to the complete two-dimensional code at the onset of the course so that they can still complete homework assignments addressing FEA concepts. The functions which represent MATLAB Grader problems are provided as MATLAB p-code to obscure its content while maintaining functionality of the two-dimensional code.

Computational Fluid Dynamics

Computational Fluid Dynamics exposes students to fundamental methodology while also familiarizing them with industry leading commercial CFD software. Foundational numerical methods are discussed and used to develop algorithms for solving a variety of partial differential equations which are then extended to the application of standard CFD algorithms. Students concurrently learn Ansys Fluent in workshop-structured laboratory sessions by solving a variety of real-world problems and gradually bridge the connection between the CFD algorithms discussed during lecture and specific options chosen in Fluent as the term progresses.

Similar to FEA, a concern of enrolling undergraduate students in an advanced numerical topic important in industry and serviced with commercial software is that students do not have enough depth to comprehend why a specific option or method is used in a particular application. Instead, they simply learn a rote procedure under the guise of understanding the subject which can lead to a potentially catastrophic interpretation of results. Additionally, variances in numerical methods comprehension and the complexities of partial differential equations presents a uniquely difficult challenge for teaching and learning the subject. To emphasize the importance of understanding the depth of the numerical algorithms and options available in a commercial CFD solver such as Fluent, coding projects are relied upon.

To facilitate a thorough investigation of CFD at a level in which undergraduate students can be successful, MATLAB Grader was used to build an understanding of the standard CFD numerical algorithms during the Fall 2023 term. Lectures consisted of a combination of writing on a board usage, PowerPoint slides, and MATLAB Live Scripts or active sessions of MATLAB projected to the front of the class. Laboratories consisted of active sessions of Ansys Fluent with PowerPoint handouts provided as a rough procedure and summary of important points. The usage of MATLAB Grader was confined to three coding projects each consisting of two parts, and were used to showcase important aspects of various CFD algorithms. The first part was a conceptual quiz which was administered via the LMS and contained a variety of multiple choice, fill-in-the-blank, true/false, and free response type questions on important conceptual aspects of the project. The second was the MATLAB Grader portion which was where the students built their CFD algorithm to solve an application-oriented problem. The conceptual quiz was worth a quarter of that project's grade and the MATLAB Grader portion was worth the rest.

The three coding projects were relatively involved due to the nature of the course material. As MATLAB Grader can only quantitatively assess students, it was deemed paramount to qualitatively assess content comprehension through the conceptual quizzes. This minimized the possibility of students happening upon a correct solution in MATLAB Grader by luck or consistent trial-and-error. Even if a student attempted to solve the project in this manner, they would still need to understand the important concepts or risk losing a quarter of their grade. The coding portion of the project itself would initially appear very involved and convoluted but

through persistent attempts by the student, afforded with the ability to infinitely submit attempts until the due date, the overarching theme of the numerical algorithm would gradually become clear for a majority of the students. Anecdotally, the students in sections of CFD in which coding projects were assigned seemed to have a significantly better grasp of foundational elements of CFD as compared to sections in which there were no coding projects. This manifested itself by successfully utilizing the commercial CFD code more efficiently (and ultimately more correctly) probably because a deeper conceptual understanding was reached. It is unclear if this is because the conceptual quiz portion of the coding projects, or the MATLAB Grader portion. Particular difficulty exists in determining the root cause because the correct solution of the MATLAB Grader part of the coding project goes hand-in-hand with a thorough understanding of the key concepts.

Aerodynamics

Aerodynamics is an introductory course on important flight concepts such as the generation of lift on lifting surfaces, the physical manifestation of drag and its effects, and the importance of various aircraft system components. The discussion encompasses a flow regime ranging from fundamental subsonic flight to elementary supersonic flight and because a substantial amount of theoretical aerodynamics relies upon numerical algorithms to evaluate various aspects of the flow phenomena, numerical projects are a natural fit for the course.

Undergraduate students do not typically connect the need for numerical methods in aerodynamics before they enroll which is often a cause for consternation. An inherent difficulty with an upper-level course is that content is frequently built off prior topics. This becomes a problem when students either forget prior course content or were exposed to the material in a different manner masking the applicability of their previous knowledge. A course such as aerodynamics is especially keen to these issues because a student must rely upon numerical methods, fluid mechanics, and thermodynamics to succeed. Moreover, when the topic features numerical algorithms to properly evaluate, the student's familiarity and comprehension of numerical techniques is intertwined with the physical realities of that topic. This presents complications in determining if student difficulties with the project are due to the numerical aspect or a lack of understanding of the physical phenomena.

As a result of these difficulties, projects were assigned throughout the Spring 2023 term with the intent of testing content comprehension qualitatively with quizzes and quantitatively with MATLAB Grader. The projects consisted of three parts spread over a duration of two weeks where the first part was due after one week, and the second and third part due after two weeks. The first part of both projects was a conceptual quiz administered via the LMS and contained multiple choice and true/false questions. The second part used MATLAB Grader and was constructed such that the student could continuously test their numerical algorithm with a similar analytical solution that was either shown in the course textbook or they had to calculate on their

own. The third part was using their code that they submitted to MATLAB Grader and validated with their analytical comparison to explore an application of their algorithm with a slightly different physical application. This application required an analysis of the results in an extension of what they created as the validated code. Each part of the project was a third of the weight of the entire project to indicate to the student that qualitatively understanding a concept, designing and building a numerical algorithm with proper testing, and using the created code is of equal importance in proper problem evaluation. This distribution also ensured that students who wrote code to simply pass the MATLAB Grader assessments cannot get full points without understanding what physically is going on.

Upon reflection, most of the students appreciated the compartmentalization of the project because each part helped the students understand components of the subsequent parts. This appeared to result in a thorough understanding of the material, but student comprehension cannot be compared to the project with only a coding component because the project has always been run with several different parts. As expected, the MATLAB Grader numerical algorithm development portion allowed students to apply their algorithm towards the third part of the project more completely since the infinite submission attempts afforded students the ability to correct their mistakes. Comparing their code to a known analytical solution also appeared helpful because the analytical solution procedure highlighted a general path towards the solution. Besides a cohort of students who do not like projects involving coding, the ease of grading, student reception, and comprehension, resulted in an expansion of this project structure for the Spring 2024 term. Instead of a comprehensive final examination, there's a comprehensive final project which relies upon the numerical algorithms created in the first two projects and then expands them. This should build cohesion among the projects throughout the entire term and the projects progressively get more complex in line with the material throughout the term.

Gas Turbines and Aircraft Engines

Gas Turbines and Aircraft Engines relies upon a synthesis of thermodynamics, fluid mechanics and heat transfer in order to model, analyze and design various aircraft engine cycles as well as individual components. The course covers a review of conservation equations, elements of compressible flow, power generation engines, aero-propulsion engine cycles, turbomachinery analysis and design, and combustor design.

A key learning outcome of this course involves students explaining the sensitivity of different performance parameters of engines and components to variations in design parameters. MATLAB Grader was ideally suited to meet these learning outcomes because students could develop scripts to investigate these parametric sensitivities. As a result, all of the homework assignments during the Fall 2023 section of the course had one MATLAB Grader problem in which students could perform these calculations. The instructor typically supplied the plotting

commands so that upon running their scripts, the platform generated the desired plots, so that students could study them to develop a deeper understanding of the trends.

The final project for course involved selecting specific components and operating parameters for an engine configuration that could meet specific thrust requirements. Unfortunately, this project could not be deployed through MATLAB Grader because of the complexity. The optimal strategy to tackle this assignment was to develop function files for each engine component that could then be connected in a main analysis script. Such functionality is not supported by the MATLAB Grader environment.

Discussion

Student Outcomes

A reliable performance metric to quantify how MATLAB Grader has affected student outcomes is not currently known. As a result, only anecdotal observations are presented which tend to generally support positive student outcomes. The following reflects these observations.

A benefit of MATLAB Grader is that it allows for much quicker iteration and exposure to the coding experience than possible with manual grading which is of particular importance for first year students in an introductory programming course. The students subsequently gain more practice and hence better proficiency in coding than otherwise possible. Within core curriculum courses, the authors noticed that students who have used MATLAB Grader before tend to struggle less with their computational assignments. This applies to their “Introduction to Computer Applications” course as well as any other prior core curriculum courses which may have assigned MATLAB Grader.

Courses adopting the use of MATLAB Grader saw a significant increase in office hour attendance due to these assignments requiring a correct answer in order to receive credit. This increase in student-instructor interaction probably positively affects their conceptual understanding by catching nuances that might otherwise go unnoticed if partial credit were allowed or if answers were copied from an online resource. An additional benefit is that this increased interaction helps foster student-faculty relationships which can substantially benefit a student’s collegiate experience [15].

While many of the courses utilizing MATLAB Grader have created problems with a graphical solution showcasing the influence of various inputs, it is not well understood how much conceptual benefit students receive. That said, such problems do enforce good programming practice because it would be too overwhelming to perform hand calculations. One possible method that was employed in CFD and Aerodynamics to further ensure conceptual

understanding is to ask related follow-up questions in a quiz administered via the LMS in addition to the MATLAB Grader problems. Moreover, assessment/exam questions can also draw on the trends that are explored in the MATLAB Grader problems to incentivize studying the output of their programs.

The strength of MATLAB Grader for upper-level computational and elective courses is that it offers a method to apply concepts in a way that could not easily be done otherwise. MATLAB Grader is highly encouraged for these types of courses because it emphasizes complete, correct solutions of complex problems over partial credit. As students transition to industry or graduate school, it becomes increasingly important to emphasize perseverance towards completion of these problems which can be aided when partial credit is irrelevant.

Potential Difficulties and Pitfalls

Many of the difficulties experienced with MATLAB Grader are not necessarily endemic to MATLAB Grader but are common difficulties observed in higher education today. These difficulties commonly present themselves with frustrating student interactions, substandard student performance, and bitter course evaluations. While it would be easy to blame student preparedness, motivation, or discipline for these difficulties, it is the responsibility of a professional educator to use this feedback to improve the student learning experience. With this in mind, the difficulties presented in this section are provided to caution and inform those looking to use MATLAB Grader in their own courses.

At the outset, it is important to be intentional about course objectives such that they align with the overall goals of the curriculum and with industry expectations for graduates. When course objectives have been established, it is important that all assessment tools, including MATLAB Grader, appropriately align with those objectives. It is important to be transparent about the motivation for MATLAB Grader so that students understand the purpose for each problem, the intended level of difficulty, and that difficulties encountered are an important part of the learning process. Finally, it is important to be consistent with multiple reminders about the problem's purpose, recommended solving approach with justification, when to ask for help, and how to ask for help.

One significant difficulty observed is how many students struggle with basic coding concepts. This issue often presents itself in the form of complaints about the course being a "coding" course instead of the engineering topic it is intended to address. In reality, the student struggles most with a prerequisite tool which makes them feel as if the course is overly difficult when it comes to computational requirements. This would be similar to taking a dynamics course in a foreign language and complaining that the course is about the language of instruction instead of dynamics. Approaching this issue requires two steps. The first step is to remind the students that while mechanical engineering is not primarily about coding, it is an important tool relied upon in the discipline. This step often requires an intervention to address inaccurate student

assumptions about mechanical engineering which frequently over-emphasis CAD and technician work and under-emphasize analytical work. At the very least, instructors can emphasize that the development of coding skills enforces algorithmic thinking which is critical in all forms of mechanical engineering even if the engineer does not specifically code in their career. Additionally, it is important to have consistent computational assignment demands across sections of similar courses so that students see this as the viewpoint of the institution and not just of individual professors (and therefore more easily dismissed).

Once the student understands the importance of coding in the mechanical engineering profession, the second step is to provide resources so that the student may begin to acquire the appropriate coding skills. MATLAB Grader problems can act not only as a vehicle for teaching programming, but also as reinforcement to the important concepts in subsequent courses. Understanding this, it may be worthwhile to scale problem difficulty in an assignment set and throughout the course such that introductory problems have significant scaffolding that the students may then use as a template for the rest of the assignment set.

It should be emphasized that the authors believe students will be better prepared for computational assignments within their core curriculum courses if MATLAB Grader is utilized in their introduction to programming course. The instantaneous feedback, multiple retakes, and requirement for mastery seems to build a better foundation than would otherwise be possible if problems are only given in a traditional manner.

Even if a student may have the requisite coding capacity for a course, they may sabotage themselves by taking an inappropriate approach to solving the problem. Aspects of such ill-advised approaches include:

- Not trying to outline the problem approach on paper first which could be in the form of pseudocode or an outline of the solution procedure.
- Making haphazard guesses instead of methodically reevaluating where errors may have occurred.
- Not using the desktop version of MATLAB for debugging. Debugging in the MATLAB Grader environment is much more difficult than the desktop version.
- Using hand calculations to solve the problem whereby only the final number is entered in MATLAB.
- Academic dishonesty by copying solutions instead of doing their own work.

The first four bullet points illustrate shortcuts a student might attempt which usually creates more effort in the long run. Part of the reason these shortcuts are taken is because there is not any recourse for simply entering answers or lines of code to see if MATLAB Grader will award credit. The allowance of infinite opportunities is designed to lower student anxiety and increase the possibility of a student converging on the correct answer, but it can also contribute to a haphazard “try anything” approach. While MATLAB Grader can track the number of attempts, academic dishonesty can be difficult to track since random ID’s are given to the solutions, not student names.

In some cases, it is understandable how student confusion could lead to haphazard approaches. For example, student code can be assessed with a “custom code” option which allows for an assessment of the students work based on any code which the instructor writes to perform the assessment. Feedback seen by the student for this scenario will include a variable they have not yet seen since it was defined by the instructor. Also, there may be different methods of solving the problem which may be equally correct but result in slightly different values. It is possible to account for different answers in MATLAB Grader, but the instructor must have the foresight to account for it when they set up the problem. Similarly, the instructor should be very specific in their problem statement and in their customized feedback. Any vagueness is exacerbated when solutions must be exactly correct for credit. This all means that the development time for MATLAB Grader problems is extensive and that there will likely be unforeseen issues caught by the students. However, with care, patience and flexibility, the development of these problems saves significant grading time.

Technical Difficulties

A common issue with MATLAB Grader is when students use the command “clear”, resulting in an inability for the student’s solution to be tested against the instructor’s reference solution. This can be mitigated by mentioning it in the instructions of the problem, locking a commented code line in the student solution template warning of “clear”, or as an assessment to check to ensure the “clear” command is not within the student solution.

Another technical issue that has arisen is difficulty interfacing with the LMS. This has come in the form of problems not showing up for students and grades not being recorded. The solution thus far has been to ensure that the latest LMS external tool for MATLAB Grader is being referenced in the problem. On occasion students have encountered long solve times when running MATLAB Grader within the LMS, but this has not been observed after using the latest version of the tool.

Recommended Practices

It is important that instructors understand the purpose of each MATLAB Grader problem and how it aligns with the course objectives before creating the problem. The allure of automatic grading is enticing, but hastily created problems can result in confusion amongst the students which translates into a poor user experience and ultimately more difficulty in understanding course content. The frustration from these hastily created problems likely harms more than helps student learning. The appropriate development of MATLAB Grader problems takes time, and it was found that it is better to start small with specific, well-made problems than to do a complete overhaul of the course. Part of such intentionality is the recognition that MATLAB Grader is a tool to support student learning and not the “end-all-be-all” of assignments.

Within the problem statement section for each MATLAB Grader assignment, it is important to clearly state the purpose of the problem and how it identifies with one or more of the course learning objectives. This communicates to the student that the problem is not "work for work's sake" but is designed to foster their growth as an engineer. Following the problem purpose, the instructor needs to be very specific in the problem statement as to what variables are being solved for, and what units those variables are expected to be in. It should also include a description of the approach that students should take when solving the problem. For a typical core curriculum course, this approach might be presented to the students with the following steps:

1. Before coding, write out your solution process on a sheet of paper. This process should include the following sections:
    a. Given: Describe what is provided and the associated values and units. Use variables that will be used in your MATLAB code. Redraw the problem diagram.
    b. Find: Describe what needs to be solved for. Use variables that the MATLAB Grader solution calls for
    c. Solution: Describe any assumptions that are made. Draw any diagrams appropriate to the solution process such as a free body diagram. Describe each step in your process and follow it with the necessary supporting equations.
    d. Pseudocode: Briefly describe how your solution process on paper will be coded in MATLAB
2. Next, code your solution in a desktop version of MATLAB for easier debugging. Use comments to describe and delineate sections in the solution process corresponding to what was written on paper.
3. Finally, copy and paste your MATLAB code into MATLAB Grader to check your solution process.
4. If one or more assessments do not pass, please review the assessment feedback carefully to see where the error may have occurred either in the solution process or in the implementation of the solution into MATLAB.

If supporting functions are included, it is important to describe what those functions do and the inputs and outputs for those functions. When designing the problem, it is more useful to request arrays for output instead of scalar values to encourage the use of MATLAB functionality and to showcase trends.

After the problem statement has been defined, good coding practices should be employed for the instructor's sake when writing the reference solution so that the reference solution is clear when reviewed at a later date. It is also a good idea to carefully create written problem solutions like those that the students are asked to create. Within the learner template, consider what the objectives of the assignment are. The goal for any assignment is to target learning objectives while minimizing other aspects that may be time-consuming and of little pedagogical benefit. This should influence how much code an instructor may want to provide to the student within the template. An example of such code might be more advanced matrix indexing operations which could greatly simplify the solution process for the student but could be difficult for the student to

figure out how to do. Also, as mentioned prior, consider including a locked statement at the top of the learner template with the reminder that including "clear" anywhere in the solution will result in an error. It may be wise to include redundant instructions reminding students to follow the appropriate process before using MATLAB Grader (e.g., perform solution on paper first, then in the desktop version of MATLAB, then in MATLAB Grader). Following these instructions, an instructor may ask students to write their name as the first commented line within their solution so that the instructor can more easily bypass the random identifiers provided by MATLAB Grader.

The instructor should also be aware of all the different solution approaches that may be used to solve the same problem and if slightly different but equally correct answers could be generated as a result. These approaches should be accounted for in the reference solution and in the tests. Note that the "MATLAB Code" assessment type can account for acceptable tolerance and multiple solutions in the case that multiple solution approaches could be used. In addition to the correct approaches, consider the multitude of errors students could encounter in their solution process and design the tests and feedback accordingly. Note that this will often be an iterative process since students will likely find errors previously unforeseen by the instructor. Remember to be patient when this happens or in the other likely event of a student finding an error in the problem statement or even within the instructor's solution. If students still use "clear" within their solution, it may be worth using an assessment to ensure that "clear" is absent in the student solution.

When considering the allowable number of attempts, keep in mind the balance between minimizing student anxiety and discouraging haphazard approaches. The authors all use an infinite number of attempts but encourage students to reach out if they are still unable to solve for the correct answers after a certain number of attempts (e.g., 10 incorrect submissions). This reminder can also be placed as part of the locked instructions within the learner template.

In addition to the automatic grading provided by MATLAB Grader, it is recommended that instructors collect student's corresponding written work and MATLAB code. While this does not ensure that students will follow the recommended problem-solving order, it does encourage it by requiring all steps in the process to be turned in. Additionally, this enables the instructor to check for proper coding methodologies and for instances of academic dishonesty. To further encourage students to follow the recommended problem-solving order, instructors may need to be careful with the help they give students during office hours. Students often come in after not getting the problem correct in MATLAB Grader and will be quick to show the errors that arise on their screen. However, just like a physical therapist who examines faulty movement patterns to determine the underlying cause of a patient's knee pain, so must the engineering instructor start from the beginning when helping students. First check that their problem-solving approach is correct. If they do not have their problem-solving approach written on paper, insist that they complete that step first. Next, move on to the desktop version of MATLAB to ensure that the code functions as expected. Only if there are no errors at this point should the instructor review what is happening at the MATLAB Grader level.

Some MATLAB Grader assignments are used to demonstrate conceptual topics that can be missed unless the student's attention is explicitly called to it. For this reason, it may be pertinent to implement LMS quizzes following the submission of MATLAB Grader assignments. For example, the quiz might ask how increasing the length of a rod influences the expected buckling load if the preceding MATLAB Grader problem provided such graphical output.

For ease of student access, it's advised to implement MATLAB Grader within the university's LMS. This is a convenient approach because it automatically integrates MATLAB Grader scores within the gradebook. Prior to integrating the problems, ensure that the most up-to-date external tool for MATLAB Grader is installed (V1.3 at the time of writing). Once selected, this will bring up the MATLAB Grader environment where the instructor may select a previously created problem or develop one from scratch. Note that when copying a course on the LMS to use as a template for the same course in a new term, the MATLAB Grader assignments will be present, but the instructor will need to reselect the appropriate MATLAB Grader problems. For this reason, it is recommended to use the MATLAB Grader web environment when creating MATLAB Grader problems and store them within a collection that can be easily navigated.

It is important to recognize that MATLAB Grader problems fit most squarely within the "Application" portion of Bloom's taxonomy. Therefore, these problems would most likely be too difficult for students still struggling in the lower levels of remembering and understanding concepts. To address this issue, instructors could assign concept quizzes or assign problem components such as drawing a free body diagram. Likewise, MATLAB Grader requires a convergent solution and may not easily address the more open-ended levels of Bloom's taxonomy (evaluation and creation) like a design project could.

There are some interesting problems that MATLAB Grader could facilitate that may suit a portion of the class and be overwhelming for the rest. For example, a problem including projectile motion with air resistance requires an iterative approach that would appropriately challenge students trying to expand their understanding of particle dynamics but would overwhelm others. Unfortunately, the latter group may spend an excessive amount of time trying to solve the problem when their time could be better spent on problems addressing more basic concepts. For this reason, it may be beneficial to apply a "Specifications Grading" approach where certain levels of problem difficulty are assigned according to the grade a student aspires to achieve in the course [16].

Specific Guidelines to an Introductory Programming Course

MATLAB Grader is most especially useful in an introductory programming course because it allows for frequent iteration that is not otherwise easily achieved via standard grading approaches. Also, requiring correct solutions ensures that the students practice until a correct

answer is achieved which is in accordance with mastery learning principles [16], [17]. Since this course may be the student's first exposure to coding, it is important to appropriately scaffold assignments within a set to facilitate their learning. An example problem set corresponding to learning arrays, for loops, and plotting may look like this:

- Problem 1: Provide an array and request the use of indexing to pull values from the array.
- Problem 2: Provide an equation in the problem statement (e.g., $f(x) = x^3 - 2x^2 + x$) and a scaffolded "for" loop within the template. Here the student must enter the equation with appropriate indexing. Code for plotting would be provided in the student template.
- Problem 3: Provide a different equation without the scaffolded "for" loop in the student template. The student must create the entire "for" loop with appropriate indexing. Code for plotting can be provided in the template.
- Problem 4: Provide multiple arrays and request student to plot arrays without the use of scaffolded code.[1]
- Problem 5: Provide an equation with input arrays. Ask the student to calculate the output array via a "for" loop and plot the results.
- Problem 6+: Assign word problems where the student must develop their own equations, calculate the output array via a "for" loop and plot the results.

Following these problems, the instructor could consider incorporating nested "for" loops and 3D surface or line plots depending on the objectives of the course. At this point it is extremely crucial to enforce good coding habits. Therefore, the pseudocode should be collected for at least some of the problems in addition to the actual MATLAB code and graded. Additionally, some problems should be assigned outside of MATLAB Grader. It is especially useful if the instructor requires that these problems be checked off in person prior to submission so that the student's solving strategy and code format may be monitored.

Since a major objective of this course is for students to learn appropriate debugging techniques, it is appropriate to allocate class time for demonstration of debugging within MATLAB and to provide assignments geared towards debugging. It is critical to emphasize that all debugging should be done within the desktop version of MATLAB since it is much more difficult to debug within the MATLAB Grader environment.

Finally, it is most important to allocate a significant portion of class time for students to work on problems (MATLAB Grader or otherwise). Likewise, it is useful to hold office hours in a public setting that can accommodate many students. This design allows students to correct their mistakes more quickly by increased interaction with each other and the instructor.

Specific Guidelines for Upper-Level Computational and Elective Courses

These courses perhaps provide the most engaging application of computational problems and therefore present an exciting use case of MATLAB Grader. Here more than anywhere else, it

[1] For a method of assessing plots, refer to GraderPlus, especially the function "mg_isCurveInPlot" [18].

is recommended to slowly build the complexity of the MATLAB Grader problems throughout the term. It also may make more sense to present some concepts via MATLAB Live Scripts because it can easily integrate instructional content with MATLAB code to demonstrate trends in the form of graphics or animations.

One limitation of the MATLAB Grader platform is that it is not well-suited for more complex programming assignments where students may need to develop multiple function files as part of the solution process. The platform is rigid in that the problem must be able to be solved as a single script file or a single function file. For instance, the solution of the pressure distribution on a supersonic airfoil may require separate function files to determine the properties across an oblique shock and expansion fan. Such a problem is challenging to administer in MATLAB Grader because a student cannot develop these function files in MATLAB Grader and then have MATLAB Grader utilize these functions. Instead, the instructor has to provide these function files as part of the problems for students to call on, which may not fully satisfy the learning outcome that the instructor is aiming for.

It should be emphasized that the course learning objectives should be carefully reviewed to determine if the amount of effort spent on MATLAB Grader design will reap sufficient benefit. Remember that what an instructor considers “fun” may easily be overwhelming for most students!

Conclusion

MATLAB Grader was employed in undergraduate courses throughout a mechanical engineering curriculum at Milwaukee School of Engineering, a PUI. This versatile tool was used to aid in the teaching of programming fundamentals, key concepts in core courses, and assign more complex design-type problems in upper-level computational and elective courses. The tool affords students the opportunity to investigate a topic in-depth while at the same time having the solution procedure scaffolded by receiving instantaneous feedback on intermediate stages. Integrating MATLAB Grader through an LMS is straightforward and offers the benefit of automatically updating the student’s grade based on the automatically graded assessments. It was found that it is time consuming to create and deploy a well-designed problem, but it is essential for creating the best possible learning experience with MATLAB Grader, minimize student frustration, and ultimately save time for the instructor. Through the implementation of many MATLAB Grader problems in many different courses, lessons were learned which informed the development of a rough set of guidelines presented in this paper. This includes being deliberate about designing the MATLAB Grader problem according to specific learning outcomes and using these outcomes to inform the type of problem (either script-based or function-based) and what should be assessed. The problem description should be written as clear as possible to ensure a higher probability of successful completion. Good coding practice should not be ignored and an emphasis on writing pseudo-code, a flowchart, or problem solution procedure for example, should be encouraged if not a direct component of the assignment’s grade. Care should be taken when choosing which assessment types are relevant, and testing a simply array variable may be

more useful as compared to the “custom code” option as the later shows assessment variables to students confusing them in the debugging process. Also be aware of the limitations of the platform with the “clear” command and inability to use multiple function files. While these drawbacks of MATLAB Grader limit the scope of what types of problems can be assigned, the potential benefits outweigh the difficulties, and a well-designed problem is worth the effort.

Future work will focus on expanding the repository of problems for each course as well as developing problems for other courses in the curriculum. In addition, efforts are also focused on including more detailed feedback in the existing problems, and in the new problems that are being developed. Mechanisms to help students reflect on analysis results are also being expanded and developed so that they can better retain concepts and trends conveyed by the MATLAB Grader problem instead of stopping once they have satisfied all of the milestones. Finally, qualitative and quantitative techniques to evaluate the impact of MATLAB Grader problems on student learning are also being explored to further validate the utility of this tool. This includes the possibility of a formal survey of student feedback to garner recommendations and guide future refinements.

References

[1] J. Hollingsworth, "Automatic graders for programming classes," vol. 3, no. 10, pp. 528-529, 1960.

[2] S. Song, M. Antonelli, T. W. Fung, B. D. Armstrong, A. Chong, A. Lo and B. E. Shi, "Developing and Assessing MATLAB Exercises for Active Concept Learning," *IEEE Transactions on Education,* vol. 62, no. 1, pp. 2-10, February 2018.

[3] R. L. Armacost and J. Pet-Armacost, "Using Mastery-Based Grading to Facilitate Learning," in *33rd Annual Frontiers in Education*, 2003.

[4] J. Lorkowski, V. Kreinovish and O. Kosheleva, "In Engineering Classes, How to Assign Partial Credit: From Current Subjective Practice to Exact Formulas (Based on Computational Intelligence Ideas)," in *2015 IEEE Symposium Series on Computational Intelligence*, 2015.

[5] J. E. Toney, "Application of Mastery Learniner in an Online MATLAB Programming Course," in *2023 ASEE Annual Conference & Exposition*, Baltimore, MD, 2023.

[6] Y. B. a. A. Vignoni, "Automated code evaluation of computer programming sessions with MATLAB Grader," in *2021 World Engineering Education Forum/Global Engineering Deans Council (WEEF/GEDC)*, Madrid, Spain, 2021.

[7] M. Li, "Developing Active Learning of Linear Algebra in Engineering by Incorporating MATLAB and Autograder," in *2023 ASEE Annual Conference & Exposition*, Baltimore, MD, 2023.

[8] L. N. a. K. Hekman, "Improving Student Learning Experience with MATLAB Grader and Live Scripts," in *2022 ASEE Annual Conference & Exposition*, Minneapolis, MN, 2022.

[9] N. Smith, "Integration of Instructional Technology Tools including Matlab Grader to Enhance Learning in a Hybrid Vibrations Course," in *2020 ASEE Virtual Annual Conference Content Access*, 2020.

[10] M. C. Sevier and V. Prantil, "Bridging FEA Theory and Practice with MATLAB Grader - Work in Progress," in *2022 ASEE Annual Conference & Exposition*, Minneapolis, 2022.

[11] M. Holmgren, "X steam for matlab," 21 October 2006. [Online]. Available: www.x-eng.com.

[12] T. L. Bergman, A. S. Lavine, F. P. Incropera and D. P. DeWitt, Introduction to Heat Transfer, Wiley, 2011.

[13] P. Bishay, "Teaching the finitie element method fundamentals to undergraduate students through truss builder and truss analyzer computational tools and student-generated assignments mini-projects," *Computer Applications in Engineering Education,* vol. 28, pp. 1007-1027, 2020.

[14] P. Kosasih, "Learning Finite Element Methods by Building Applications," *International Journal of Mechanical Engineering Education,* vol. 32, no. 2, pp. 167-184, 2010.

[15] P. Felten and L. M. Lambert, Relationship Rich Education, Baltimore: Johns Hopkins University Press, 2020.

[16] L. B. Nilson, Specifications grading: Restoring rigor, motivating students, and saving faculty time, Routledge, 2015.

[17] J. M. a. J. Ranalli, "A mastery learning approach to engineering homework assignments," in *2015 ASEE Annual Conference & Exposition*, Seattle, WA, 2015.

[18] D. Kosf, "GraderPlus - A Library for Writing Test Code in MATLAB Grader," [Online]. Available: https://github.com/DavidKosf/GraderPlus. [Accessed February 2024].