2023 Annual Conference & Exposition
Baltimore Convention Center, MD | June 25 - 28, 2023
The Harbor of Engineering Education for 130 Years
ASEE
Paper ID #37645

Employing Live Scripts for Implementing Virtual Laboratories and Activities

Dr. Rick Hill, University of Detroit, Mercy

Dr. Richard Hill is a Professor and Assistant Dean in the College of Engineering & Science at University of Detroit Mercy. Dr. Hill received a B.S. degree in Mechanical Engineering from the University of Southern California in 1998, and an M.S. degree in Mechanical Engineering from the University of California, Berkeley in 2000. He joined the faculty of Detroit Mercy in 2008 after receiving a Ph.D. degree in Mechanical Engineering and an M.S. degree in Applied Mathematics from the University of Michigan, Ann Arbor. His research interests lie in the areas of vehicle control, control and diagnosis of discrete-event systems, modular and hierarchical control, and engineering education. Dr. Hill also has a strong interest in diversifying the STEM pipeline and leads the innovating Detroit's Robotics Agile Workforce (iDRAW) program in partnership with underserved Detroit-area high schools.

Employing live scripts for implementing virtual laboratories and activities

Richard Hill
Department of Mechanical Engineering, University of Detroit Mercy

Abstract

Significant research exists demonstrating the benefits of active and inquiry-based instruction for student engagement, learning, and knowledge retention. The emergence of ever improving software tools provide instructors valuable resources for developing virtual activities to be used within lecture courses, or in place of physical laboratory experiments. This paper describes the use of the MATLAB Live Editor for creating interactive live scripts. These live scripts combine code, formatted text, graphics, and live controls such as numeric sliders, buttons, and drop-down lists, for the creation of lectures and virtual activities that illustrate complex topics through interactive figures and animations. When deployed as in-class activities for students and as interactive homework assignments, students are able to build intuition for course material and construct meaning for themselves. This paper provides techniques and examples for developing and employing live scripts for teaching a range of topics from mathematics, engineering, and science. The author has piloted these techniques in online and in-person courses related to modeling and control over the past two years at University of Detroit Mercy. Student survey data from the winter 2023 semester indicates students found significant value from the live script activities implemented in their course.

Keywords

Active learning, inquiry-based pedagogy, virtual laboratories

Introduction

Learning advanced topics in engineering, mathematics, and science can be challenging for even the best students. Even students who excel academically and can get the “right” answer and repeat back the material they have been presented often lack a deep understanding of the topics they are learning. These challenges arise because many concepts in these fields are abstract; they aren’t physical and can’t be touched or observed directly. Furthermore, students are faced with covering a lot of material over a range of courses, all of which are competing for their time and attention.

Traditionally, courses in these fields have been taught via passive lecture. Lecture has its advantages in that it is quite efficient as a means for disseminating information in a clean, summarized format. The problem with lecture is that students often don’t build their own understanding of the material which limits their retention and ability to apply the information. It is also unlikely a student can maintain their focus for much of a typical 50- to 75-minute lecture. Another drawback of lecture is that a student often doesn’t discover any gaps in understanding

until they go to apply the material, typically while doing their homework, at which point the instructor is no longer available to answer their questions.

Active learning techniques can help address some of these deficiencies as recognized by a range of studies, while ultimately leading to improved student performance [1] [2] [3]. A traditional form of active learning in engineering and science curricula has come in the form of physical experiments [4] [5]. Often these activities take place in a dedicated lab section since the experiments can take significant time to set up and conduct, may require considerable space, and can be messy or dangerous. The use of virtual activities can help to overcome these limitations, in addition to avoiding the cost of the physical equipment, while affording many of the same benefits. There can even be advantages to virtual labs in that students can easily pause and replay an activity, in addition to having more space to reflect since they don't have to focus on procedural skills to the same degree [6]. Virtual labs have been implemented, for example, using a range of numerical simulation tools [7] [8]. A potential challenge of using simulation is that the interface might not be straightforward and it can require additional instructional time to introduce students to the use of the software tools for running the simulations. Work has also been done to create interactive applets and animations for conducting virtual experiments such as PHET [9]. These virtual activities can be very easy for students to use, but their creation may require specialized tools and knowledge which most instructors don't have. In that case, instructors can use materials created by others, but if the virtual activities don't exactly meet their specific needs, they are unable to adapt the resources.

In this work, the use of the *MATLAB Live Editor* for the creation of virtual activities is explained. This tool requires some knowledge of MATLAB, but the learning curve is not too high and many engineering and mathematics faculty are already proficient in the software. The resulting live scripts are very easy for students to use and interact with, even if they don't themselves know how to program in MATLAB. Furthermore, the resulting live scripts are open and can be readily shared with and adapted by other faculty.

In the remainder of this paper, the MATLAB Live Editor is introduced and then a series of implementation examples are described that could be applied to a variety of courses in engineering, mathematics, and the sciences. The paper then concludes with some tips on how to integrate virtual activities into one's courses, including results from the author's experience piloting activities in their own courses at University of Detroit Mercy.

The MATLAB Live Editor

An example *live script* created with the MATLAB Live Editor is shown in **Figure 1**. The Live Editor environment is similar to any modern word processing software such as Microsoft Word or Google Docs. The live script can readily include formatted text, headings, hyperlinks, equations, and figures. What distinguishes live scripts is the ability to run MATLAB code and include live controls that allow the user to interact with the document. For example, the user can change the value of a variable or a setting and observe figures and animations react in real time. The live script depicted in **Figure 1** includes a *Numeric Slider* which allows the continuous adjustment of the variable *Kp* (in this case, a proportional control gain). Upon adjustment of the

slider, the associated MATLAB code is set to run automatically and update the figure produced by the code. This interface is very easy for students to interact with and requires almost no additional instructional overhead. In this example, the MATLAB code is visible to the user which could be instructive if a related goal is to teach MATLAB programming. However, if the code obfuscates the learning goal, it can be hidden. Furthermore, the output of the code can be included inline within the document as it is here, or if it is preferred, the output can be shown to the right of the live script document.

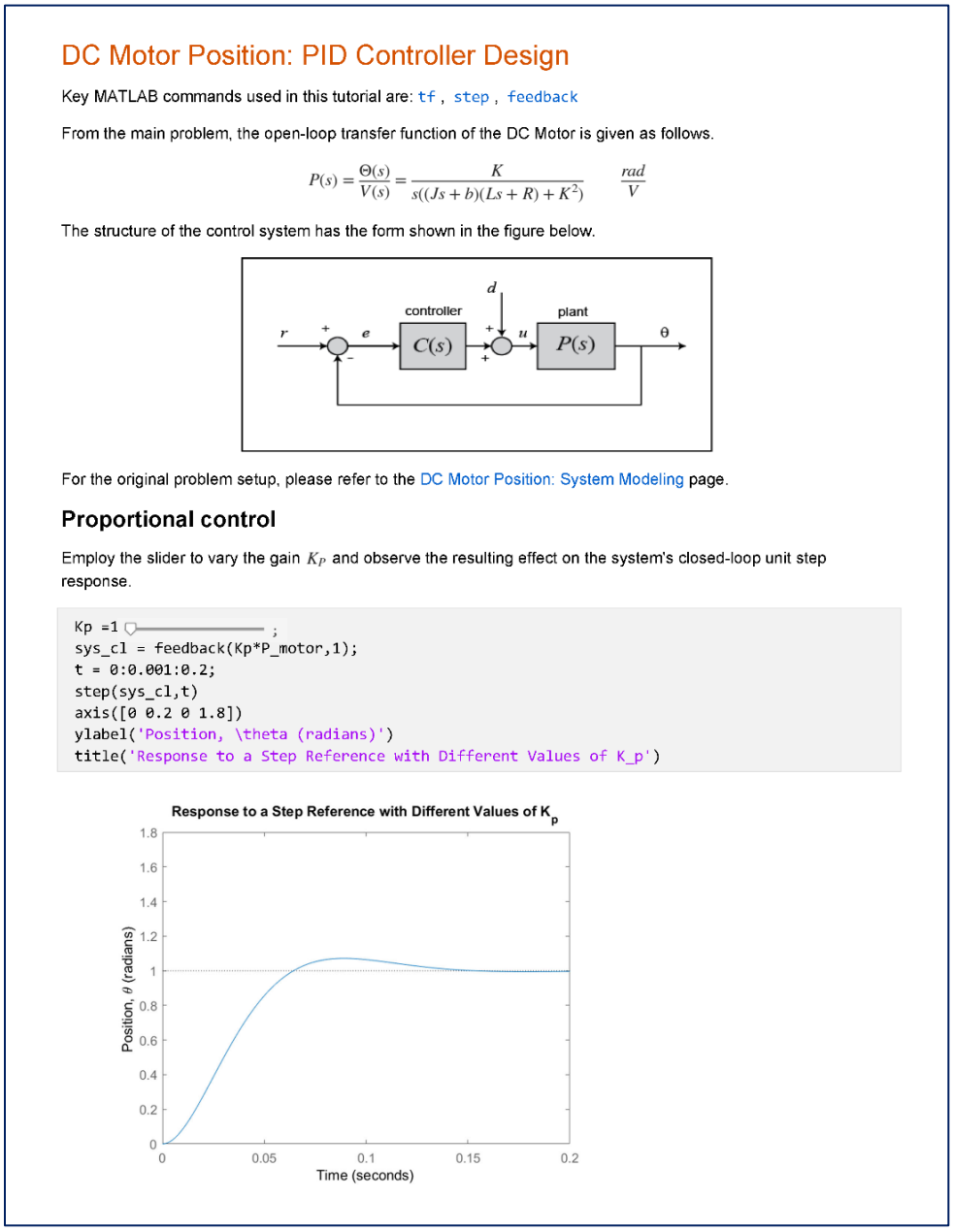

Figure 1 *Example live script created with the MATLAB Live Editor*

Other live controls that allow a variable to be changed "continuously" are the *Numeric Spinner* and *Edit Field* controls. With both the numerical slider and spinner, you specify the range of

values available as well as the step size that can be taken in making a selection between the minimum and maximum values. With the slider, the specific value is chosen by moving the slider, while with the spinner there is an up arrow and a down arrow that allows the user to step between the available values. Alternatively, there is an *Edit Field* control where the user can directly type in the desired variable value, without being limited to the discrete steps available with the slider.

A fundamentally different live control is the *Drop Down* control. This control configures a menu to be specified that allows the user to choose between discrete options or settings. For example, a user could choose between fundamentally different mathematical functions or models that aren't simple scalar values. The final two live controls are Boolean in nature, the *Button* and *Check Box*. These two options essentially control whether a piece of code runs or not. For example, a button launches a specified section of code to run when clicked, while the check box can be configured to either cause a piece of code to be run when checked, or to alternatively inhibit a piece of code from running. Examples of how these controls might be used include to show or hide the solution from a student, or to refresh a figure.

The Live Editor also includes *Live Tasks* which are comprehensive, interactive applications for conducting a variety of analysis and processing tasks. **Figure 2** shows a depiction of the MATLAB Live Editor environment with the live tasks drop-down menu open.

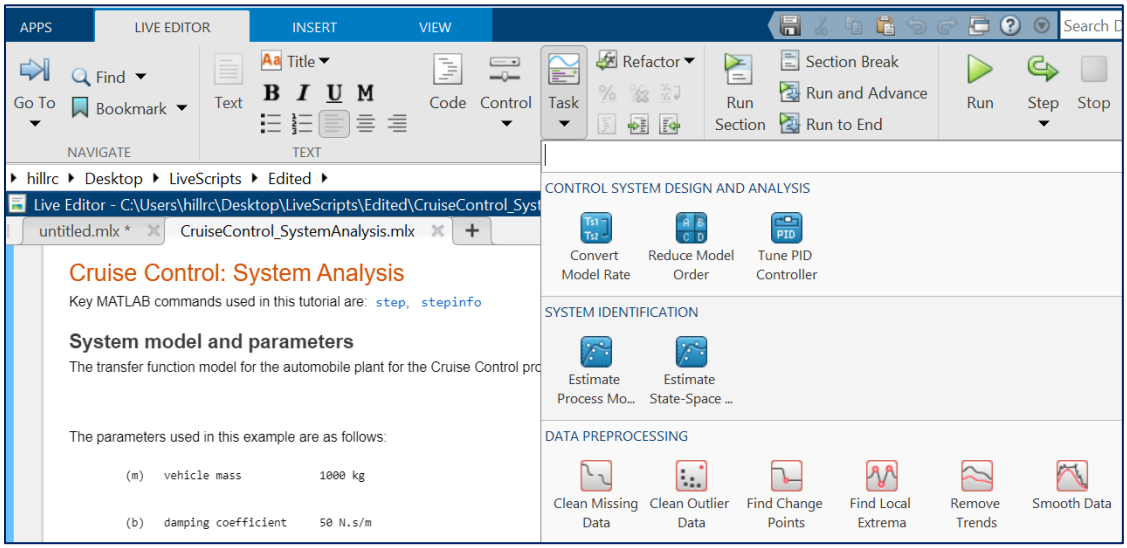

Figure 2 *Depiction of the MATLAB Live Editor environment*

Implementation Examples

In this section, several implementation examples are described to illustrate the functionality of the MATLAB Live Editor while providing examples of how it can be used across a range of science, engineering, and mathematics courses. Many of the examples that will be presented here can be accessed from the website: *https://www.mathworks.com/campaigns/products/control-tutorials.html*

Connecting Math to the Physical World

Many students have learned mathematics by memorizing abstract procedures and facts. This approach doesn't promote deep understanding and can limit the students' ability to retain the information as well as their ability to apply the concepts in different contexts. Visualizing mathematical equations in terms of their graphical representations can help students to build understanding. The emergence of graphing calculators and software such as the base MATLAB, Mathematica, Desmos, and GeoGebra have proven to be powerful tools for helping students to visualize mathematical concepts. However, sometimes students can be challenged in setting up an investigation in terms of how to input the mathematical relationships, how to formulate the graphical output, and how to make the appropriate changes to highlight the concept an instructor may wish for them to recognize. By building a custom live script, the instructor can remove much of the mental overhead required by the student to set up the investigation so that they can focus on interpreting the result. For example, the instructor can set up the mathematical structure with specific elements constrained, while leaving other elements free, and can format the scales, colors, and labels of the outputs for the students.

Some simple examples of how to employ the Live Editor for developing mathematical understanding is to employ numeric sliders to modify the parameters of an equation and observe the impact on the graph of the resulting function. For example, to see the effect on the amplitude, frequency, and phase of a sinusoidal function, or the effect of coefficients on the shape of a surface in three dimensions. The live script shown in **Figure 3** includes sliders for changing the real and imaginary part of complex conjugate poles of an underdamped second-order system (same as changing the roots of the characteristic equation of a second-order differential equation). The generated figures are shown to the right of the live script (as opposed to inline) and include one plot showing the locations of the poles in the complex plane and a second plot showing the output response of the system for a step input. The annotations on the two figures are added by the custom functions `displayPoles` and `displayCharacteristics` which are defined at the end of the live script (not shown).

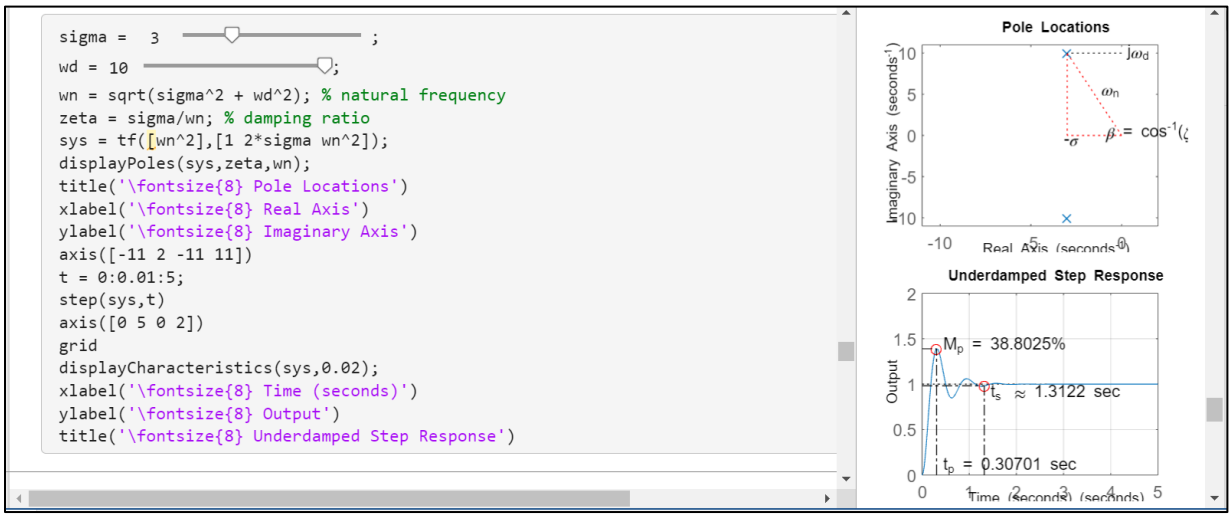

Figure 3 *Live script with sliders for visualizing mathematics*

A related live script that employs drop down controls is shown in **Figure 4**. In this case, the first drop down menu allows the user to select between distinct linear systems such as a 1st Order System, an Underdamped 2nd Order System, etc. (with some initial conditions included). In this case, the systems are configured within the drop-down control in terms of the matrices of their associated state space representation (A, B, C, D) and the initial value of the state vector (x0). The second drop-down menu then allows the user to select between different forcing inputs to the previously chosen system model. Plots of the resulting free and forced response of the chosen system and forcing function are then displayed to the right.

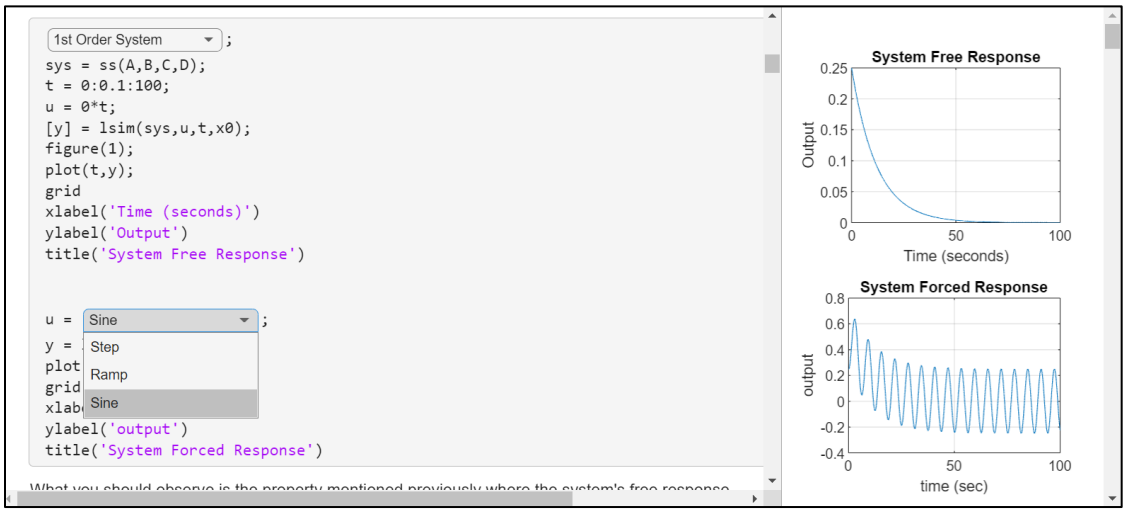

Figure 4 *Live script with drop-down menus for visualizing mathematics*

The live script from which the images of **Figure 3** and **Figure 4** were taken can be downloaded from the web site introduced at the beginning of this section[1].

Virtual Experiments

The live scripts presented so far allow students to "experiment" with the parameters of different mathematical relationships. These mathematical relationships can represent physical systems and hence can allow instructors to implement virtual laboratories allowing students to construct meaning for themselves. The advantages of using live scripts in this capacity is that it eliminates the expense of physical equipment, while reducing the required space and set up time so that the experiments can be conducted in a regular classroom in the middle of lecture, during a synchronous online class meeting, or even by students as part of their homework outside of class time. Numerical simulation tools have been used to implement virtual lab activities in this manner for quite some time. The advantage of live scripts is again that they reduce the mental overhead for the students. With live scripts, the students don't have to go into a model to change parameters or structure, they don't have to run distinct iterations of the simulation, or store and format the output data in a way to highlight the concept being targeted. With a live script, the controls are easy to interface with and the change in the output can be seen immediately in exactly the manner the live script author intended.

[1] *https://www.mathworks.com/campaigns/products/control-tutorials.html#system-analysis*

In addition to collecting and visualizing data, physical experiments can help students build intuition through observation of motion, sound, and light, which can be quite visceral and can connect with students in a manner that a trace on an x-y plot can't. Such physical observations can be emulated in virtual experiments by the inclusion of animation. **Figure 5** depicts a live script with an animation of a pendulum. In this case, the initial angle of the pendulum can be varied and the resulting accuracy of a linear approximation of the nonlinear pendulum model can be assessed by comparing the resulting outputs. The first time an animation is run for a new parameter, it may run a bit slowly because of the associated computation. Live scripts, however, include a play back function (shown at the bottom of the live script) that allows the animation to run very quickly, even faster than real time, during subsequent replay.

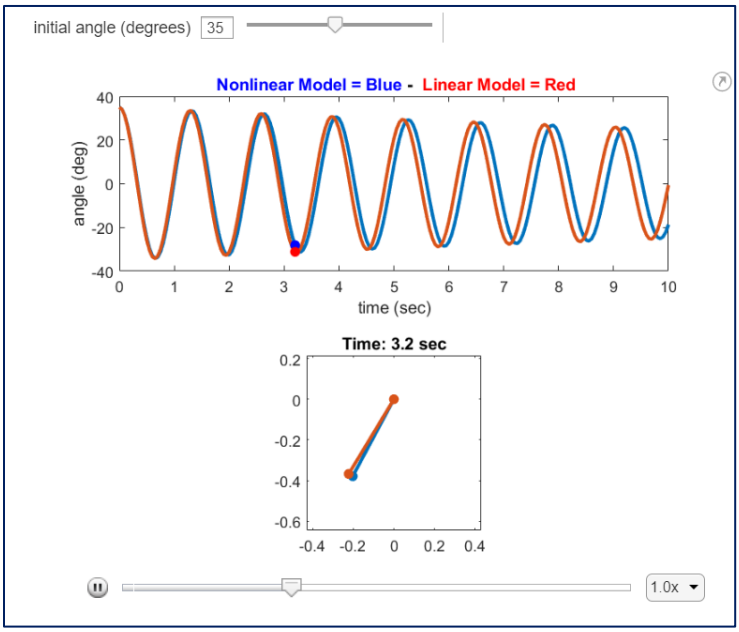

Figure 5 *Example live script with pendulum animation*

One could envision developing a related live script where different aspects of the pendulum could be altered and the resulting behavior observed. For example, the length of the pendulum, the mass of the end bob, different amounts and types of friction, etc. One of the attractive aspects of live scripts is that they are open and easy to adapt for instructors with even basic knowledge of MATLAB. The live script presented in **Figure 5** was adapted from a live script for a double pendulum available on the MathWorks website[2]. In general, the MATLAB Central File Exchange provides a good forum for instructors and general users to share materials.

In addition to demonstrating theoretical concepts, another purpose of laboratories can be to teach about implementation and dealing with real-world challenges. Even though virtual laboratories are idealized, some aspects of implementation can be emulated. For example, a virtual experiment with a motor can be built to include nonlinear friction, while noise can be added to the data and the data can be sampled and quantized to emulate the measurement process. **Figure 6** shows a portion of a live script for a virtual motor speed control laboratory. The virtual

[2] *https://www.mathworks.com/matlabcentral/fileexchange/35251-matlab-plot-gallery-animation*

experiment emulates calculating motor speed based on integer counts of an encoder with noise added and a finite sampling period. Earlier in the live script the user can experiment with the sampling frequency, but in **Figure 6**, the user can change the time constant of a first-order low pass filter to observe the tradeoff between amount of smoothness that is achieved in comparison to how much lag the filter adds.

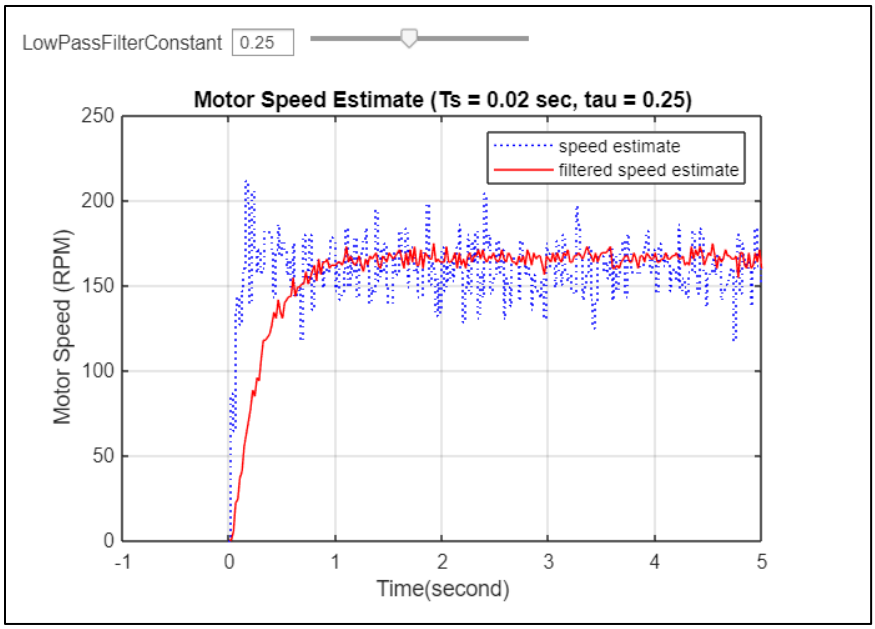

Figure 6 *Live script of a virtual motor experiment*

Building Intuition for Graphical Tools

Many fields of science and engineering employ graphical tools to assist with analysis and design. This is particularly true for the modeling and control of dynamic systems, which is the field of engineering in which the author has so far piloted the use of live scripts. For example, control engineering relies on graphical tools such as the root locus, Bode plot, Nyquist plot, and Nichols chart. The author's experience has been that the step-by-step construction of these plots is informative for the students' understanding of what they represent and how they can be used, but often students get bogged down in the process to the point that they lose their grasp of why the plots are being created in the first place. The use of live scripts can leverage the built-in functionality of MATLAB to create plots, while illuminating how the plots are generated and what they represent.

For example, the live script shown in **Figure 7** helps to illustrate the concept of frequency response and how that information is represented by the Bode plot[3]. In this live script, the user can select an input frequency via the slider. The plot on the left includes the sinusoidal input to the given system and the corresponding steady-state output (once the transient response has died

[3] *https://www.mathworks.com/campaigns/products/control-tutorials.html#frequency-domain-methods-for-controller-design*

out). This figure illustrates how the amplitude of the output sinusoid is scaled and shifted with respect to the input. The Bode magnitude and phase plots for the given system are then shown in the figure on the right. The black boxes on the plots correspond to the scaling of the output (top plot) and phase shift (bottom plot) for that specific frequency of input. As the user adjusts the frequency of the sinusoid input, the graphs on the left are updated and the boxes on the plots on the right move in correspondence to the new frequency and resulting scaling and phase shift.

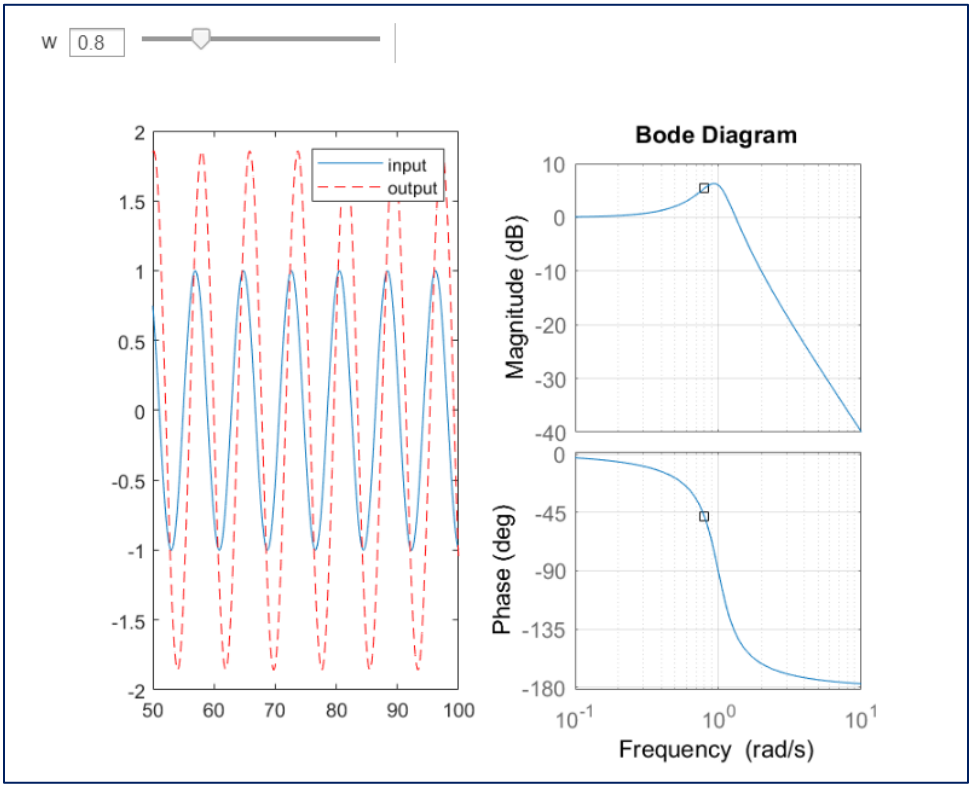

***Figure** 7 Live script illustrating the Bode plot*

Developing Design Skills

The ability to design is an important skill for many engineers. Since the design process is often iterative, it can be challenging for students in a course to have sufficient time to go through the entire analysis and design process multiple times as they would as a professional engineer. Leveraging software tools, such as MATLAB, can increase the speed with which students can repeat multiple cycles. The use of live scripts can also help students with this process. The example shown in **Figure 8** illustrates a live script where a student can tune the three terms of a PID controller and observe the impact on the system's resulting response and hopefully achieve any provided specifications[4]. In this case, annotations have been added to the response graph indicating that it is required that the Settling Time (t_s) be less than 0.04 seconds and the Maximum Percent Overshoot (M_p) be less than 16%. In a traditional controls course, often

[4] *https://www.mathworks.com/campaigns/products/control-tutorials.html#dc-motor-position-pid-controller-design*

students perform their controller design for simple, linear systems for which they have a model and can directly calculate the control gains to satisfy a given set of requirements. In practice, an engineer will have a model with uncertainty (or no model at all) and the system will have nonlinearities and complexity that prevents the theoretical design of a controller that guarantees a given set of requirements will be met. Therefore, a professional controls engineer will often need to tune their controller experimentally following the calculation of their initial design. Despite this, many students don't get practice tuning a controller under such conditions.

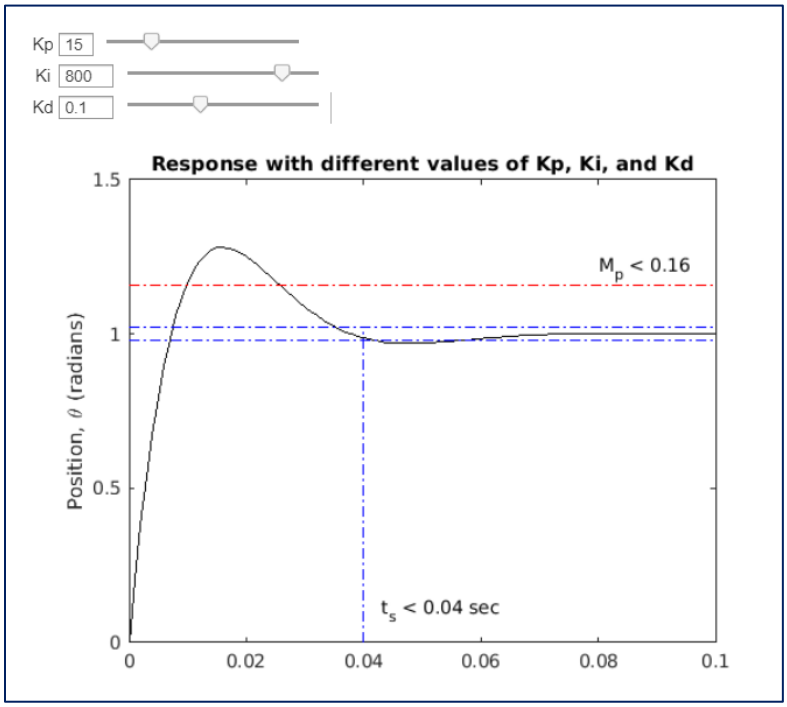

Figure 8 *Controller design live script*

The use of live scripts for iterative design can also assist with the gamification of assignments where students can play against classmates or themselves to get the best score or to beat some objective.

Integrating into Instruction and Student Feedback

Live scripts, such as the examples described so far in this paper, can be employed in a course in a number of ways to help improve student engagement and learning. In this section, some suggestions are provided including tips based on the author's piloting of activities in courses on modeling and control over the past two years at University of Detroit Mercy. The author has developed his use of live scripts based on observations and informal feedback from the students. During his most recent winter 2023 semester course offering, the author conducted an anonymous survey of the students. The course was an upper-division required undergraduate course on modeling and control system design. There were 12 total respondents, all having

senior standing except for one junior. In terms of major, 10 were mechanical engineering students and two were electrical engineering.

Instructors can employ live scripts in their lectures to demonstrate concepts; instead of explaining a topic only in words, or by speaking to a static picture, instructors can "show" an idea through an interactive graph or animation. A live script can replace an instructor's lecture slides where you can imagine a document being open within MATLAB with the instructor scrolling through the live script as their lecture proceeds. Alternatively, an instructor can jump back and forth between live scripts and their PowerPoint slides or work on a white board. If the live scripts are made available, the students can also follow along with the demonstrations on their own devices.

Instructors can also employ live scripts during class meetings as activities to engage students. These activities can break up lecture to re-engage the attention of the students and can provide opportunities for students to build a deeper understanding of the material. This requires at least some of the students to have access to a computer device during class and the live scripts can be distributed via an online course management system, or even via email or text. One option is to conduct a course in a computer lab, though in the author's experience sufficient numbers of students bring laptops or tablets to class that working in a lab isn't necessary if students team up for activities. Students can also use their phones by launching the live scripts in the MATLAB Mobile environment.

When integrating the virtual activities into class time, it is important to structure the class so that the activities help the students achieve the desired outcomes. For example, the author has found it helpful to provide the students a very direct question that they can try to answer as a result of the activity, rather than asking something vague, such as, "what did you notice?". The author often polls the students with a multiple-choice question that is specific and provides immediate feedback to the instructor about the level of the classes' understanding. In an in-person class the polling can be done by a show of hands, but can also use clickers or phone-based apps such as from Top Hat or Poll Everywhere. Polling is also built into most online course delivery services such as Zoom or Blackboard Collaborate. The polls help to promote all students engaging with the activity, which can be especially challenging with online courses. These questions can then be a jumping off point for class discussion or further clarifying instruction. For example, if there is a split in the answers from the class, the instructor can ask the students to turn to their neighbor and explain why they chose the answer they did. In online courses, students can be put into breakout rooms for discussion or the activities themselves. A *Peer Instruction* approach to teaching such as this has been shown to be quite effective for improving student learning [10] [11].

In the author's winter 2023 course offering, the students overall had a positive attitude toward the use of live scripts in class. In response to the statement, *"The live script activities were helpful for my understanding,"* all 12 students *Agreed* or *Strongly Agreed*. Employing a five-point Likert scale ranging from 1 for *Strongly Disagree* to 5 for *Strongly Agree*, the average response was 4.33. While for the statement, *"The live script activities break up the monotony of lecture,"* the average student response was 3.92. The open-ended comments regarding the live scripts were also positive. One student wrote, *"I liked being able to view and modify the live-plots in class. It really helped connect the math behind the plot to its various components."*

Regarding the Peer Instruction approach to live script use, another student commented, *"Particularly, I liked how we were given the opportunity to guess what would happen first. Then the professor explained why the guesses were wrong or right after the system was shown in the live script."*

One challenge with implementing activities during class is making time for the activities. Two approaches to address this include to cut down on the amount of material covered in a course or to displace some instruction to outside of class time via videos or pre-class readings. The author has attempted to do both, though the process hasn't been without its challenges. Specifically, in their courses the author has removed some more advanced topics (such as a lecture on linearization), has removed examples for less common systems (such as examples with non-minimum phase systems), and has reduced some of the time spent on detailed mathematical examples (fewer examples of partial fraction expansion, fewer detailed requirements in drawing the root locus, etc.). The author has also replaced some lecture content with videos to be watched outside of class. In one online graduate course, the author has completely flipped the course and replaced all lecture with videos thereby allowing class time to be used for activities and discussion. This is a large ask for the students, but it has worked well in this course because the graduate students tend to be mature, working professionals who are taking the course because they have a strong interest. In the author's undergraduate winter 2023 course, the course wasn't flipped, but students were required to watch between 15 and 30 minutes of video per week to free up a little more class time. For the statement, *"It is worth watching videos outside of class to make space for in-class activities and discussion,"* the average student response was 3.42 (between *Neutral* and *Agree*). Two of the 12 students *Disagreed* with the statement, while five students *Agreed* or *Strongly Agreed*. When asked how about the amount of time spent in class employing live scripts, all 12 students responded that the time was *About Right* or they would like *A Bit More*. Assigning the answer *About Right* a score of 3 and *A Bit More* a score of 4, the average response was 3.58.

Live scripts can also be used for work outside of the classroom. For example, they can be used for self-instruction and pre-class reading, like a living textbook. In this manner, live scripts can include narrative like a text book, but can also have embedded activities and can provide templates for students to input their own work. One approach is to include elements that the students can choose to hide or show/run when they are ready to check their work. Setting up a live script as an unfilled template can also allow live scripts to be used as homework assignments.

Conclusion

This work has described the use of the MATLAB Live Editor for the creation of live scripts for use as virtual activities and laboratories to improve student engagement and learning. Several examples were provided and the author's experience in piloting activities over the past two years has been shared. The author plans to continue developing interactive materials for use in their courses with a further goal to formally assess their impacts on student learning. In addition to the website provided by the author sharing the live script activities they have developed, the MathWorks website includes significant content explaining the functionality and implementation

of live scripts[5]. Furthermore, the MATLAB Central File Exchange[6] is a valuable resource for discovering live scripts that other users have created that can be readily adapted.

References

[1] M. Prince, "Does active learning work? A review of the research.," *Journal of engineering education,* vol. 93, no. 3, pp. 223-231, 2004.

[2] S. Freeman, S. L. Eddy, M. McDonough, M. K. Smith, N. Okoroafor, H. Jordt and M. P. Wenderoth, "Active learning increases student performance in science, engineering, and mathematics.," *Proceedings of the national academy of sciences,* vol. 111, no. 23, pp. 8410-8415, 2014.

[3] P. C. Wankat and F. S. Oreovicz, Teaching engineering., West Lafayette: Purdue University Press, 2015.

[4] L. D. Feisel and A. J. Rosa, "The role of the laboratory in undergraduate engineering education," *Journal of engineering Education,* vol. 94, no. 1, pp. 121-130, 2005.

[5] R. C. Hill, "Hardware-based activities for flipping the system dynamics and control curriculum," in *American Control Conference (ACC)*, Chicago, IL, 2015.

[6] J. Ma and J. V. Nickerson, "Hands-on, simulated, and remote laboratories: A comparative literature review," *ACM Computing Surveys (CSUR),* vol. 38, no. 3, pp. 1-24, 2006.

[7] N. V. K. Jasti, S. Kota and P. Venkataraman, "An impact of simulation labs on engineering students' academic performance: a critical Investigation," *Journal of Engineering, Design and Technology,* vol. 19, no. 1, pp. 103-126, 2021.

[8] R. C. Hill, "The Effective Use of Simulation in the Introductory Controls Curriculum," in *American Society for Engineering Education North Central Section Conference*, Saginaw, MI, 2009.

[9] C. E. Wieman, W. K. Adams and K. K. Perkins., "PhET: Simulations that enhance learning," *Science,* vol. 322, no. 5902, pp. 682-683, 2008.

[10] C. H. Crouch and M. Eric, "Peer instruction: Ten years of experience and results," *American journal of physics,* vol. 69, no. 9, pp. 970-977, 2001.

[11] R. C. Hill and K. Plantenberg, "Assessing a conceptual approach to undergraduate dynamics instruction," in *Proceedings of the 2014 ASEE North Central Conference*, 2014.

[5] https://www.mathworks.com/products/matlab/live-editor.html
[6] https://www.mathworks.com/matlabcentral/fileexchange/