2018 Annual American Control Conference (ACC)
June 27–29, 2018. Wisconsin Center, Milwaukee, USA

Automated Grading with a Software-Checking Program in the System Dynamics and Control Curriculum

R. C. Hill and Yasha Parvini

Abstract— **Automated assessment tools offer benefits to instructors by freeing them from the burden of grading, while improving student self-learning by providing immediate feedback without the need for intervention by an instructor. The power of such tools has become all the more important as new technologies enabling the scaling of knowledge dissemination have arisen, such as with Massive Open Online Courses (MOOCs). In this work, the authors provide guidelines for employing software-checking programs for the automated grading of assignments in the System Dynamics and Control curriculum that may not be immediately suited to such tools. Specifically, techniques are discussed for modifying problems that may involve symbolic or graphical solutions, or that may be more open-ended. The approaches proposed in this paper were piloted in two courses at the University of Detroit Mercy during the Fall of 2016 and the Winter of 2017. In these courses, students wrote their solutions as scripts in the MATLAB programming language within the Cody Coursework environment offered by the MathWorks. Overall, the authors found the use of the Cody tool helpful in the instruction of their courses. The student feedback offered via anonymous surveys was also generally positive regarding this approach. For example, in the absence of a person being available to grade their homework, the students appreciated the availability of the Cody tool (avg. score of 3.91/5). As a side benefit, the students also found that this approach to the course helped improve their general programming skills (avg. score = 3.96/5).**

I. Introduction

The cost of education and college tuition has long risen at rates that have outpaced inflation. Over each of the last three decades, specifically, annual increases in 4-year college tuition have grown at a rate of approximately 3-4% greater than inflation in the United States [1]. The technological advancements that have historically improved productivity rates across industries and helped reduce the costs of goods, have typically not extended to the field of education. The advent of the internet and improved content creation and delivery platforms have the promise to reverse this trend, even at the university level. Platforms such as YouTube, iTunesU, the MIT OpenCourseWare project, Khan Academy, and a variety of Massive Open Online Courses (MOOCs), allow for the broad dissemination of knowledge and high-quality lecture material from some of the world's most preeminent scholars [2], [3]. These inexpensive delivery mechanisms greatly reduce the barriers to access to a world-class education that typically arise due to a student's geography and/or socioeconomic status. The ability to implement meaningful assessment of student learning at scale, however, is much more challenging than simply broadcasting information to a large audience [4] and must rely on automated assessment or peer grading [5]. Evaluating one approach for addressing this challenge is the focus of this work.

Some of the learning platforms mentioned include means for automated assessment of student learning through online quizzes and, in the case of programming classes, software-checkers. Many textbook publishers also offer Online Homework Systems that include banks of exam and homework questions. Some of these systems, as well as many Learning Management Systems (such as Blackboard, Moodle, and Canvas), can allow an instructor to create their own online assessments [6]. These assessments can be multiple-choice or open response. At their best, an instructor may be able to create a template for a numerical problem where the specific numbers employed are randomly generated. This helps to prevent individual students from copying each others' answers, and also allows students to gain more practice by repeating a specific problem multiple times with new numbers inserted each time.

In our application, we employ a software-checker to test student-written computer programs. The use of automated software-checkers are quite common in programming courses [7], [8], but less so in general science and engineering courses. By formulating their solutions as a program, the students are in a sense doing the inverse of what they would do in an online homework system since it is the student who is formulating the "template" of the solution. Only in the testing of the solution are actual numbers employed.

The obvious advantage of employing these automated assessment tools is that they can save an instructor a great deal of time since they don't need to grade each individual student's assignment or exam. These time-savings can then free the instructor to spend more of their time interacting with students, answering questions, leading laboratories, and developing other opportunities for active learning [9], [10], [11]. Besides saving the instructors' time, these tools can also make the learning process more efficient by providing the student, and the instructor, more immediate feedback. If a student performs a problem incorrectly, they know immediately and can work to correct any misconceptions they may have. The student doesn't have to wait until they have access to the instructor, or worse, until they receive their graded homework back from the instructor, before they can determine that they misunderstood a concept. Similarly, an instructor can receive feedback about the performance of the class in real-time, and can address it sooner, than if they had

This work was supported in part by the MathWorks.

Both authors are with the Department of Mechanical Engineering, University of Detroit Mercy, Detroit, MI 48221-3038, USA hillrc@udmercy.edu and parvini@udmercy.edu.

to wait until all of the students had turned in their assignment and those assignments had been graded. Feedback is widely known to improve student learning [12] and some research, in particular, demonstrates the benefits of automated systems that provide immediate feedback [13], [14].

The challenge that arises with these automated grading tools is that it can be difficult to assess more open-ended type problems, such as those typical to an upper-division course in engineering or science. The work described in this paper, specifically, attempted to implement an automated grading system for the System Dynamics and Control curriculum, though it is believed that the lessons learned here are broadly applicable. Some of the challenge of assessment in modeling and control courses is that the assignments are often design-oriented where there can be many acceptable approaches and solutions, and further, the solutions are often symbolic or graphical in nature, rather than simply numeric.

For this work, the students were required to formulate the solutions to their assigned problems as scripts in the MATLAB programming language. Each script was then run against a series of test cases to determine if the script generated the correct output for a given selection of inputs. A couple of further advantages of formulating the students' assignments in this manner is that it gives the students additional experience in coding and algorithmic thinking in the context of non-programming courses. These skills have become increasingly important as technology continues to change the role and skills needed by the modern engineer [15]. Additionally, this approach to grading gives students some exposure to the typical industrial development process. In industry, an engineer's work typically isn't reviewed line by line, but rather, the design is tested against some set of objective, measurable requirements.

Student assignments were written in and tested using the Cody Coursework environment offered by the MathWorks [16]. The Cody Coursework environment is a software-checker that has several built-in tools that assist students in formulating their solutions and assist the instructor in generating problems, providing student feedback, and grading student solutions. However, the majority of the approaches outlined in this work can be implemented employing alternative software-checking programs or general course management tools. The authors piloted their use of this automated grading system as part of the first two courses of the control systems sequence at the University of Detroit Mercy in the Fall of 2016 and the Winter of 2017.

The remainder of this paper is structured as follows: Section II presents an overview of how the courses were implemented in the context of the Cody Coursework environment, Section III outlines how the software-checking tool was integrated into the System Dynamics and Control curriculum, Section IV presents some results from the implementation of the courses, and Section V draws some conclusions and proposes some directions for future work.

II. Overview of Implementation

The piloting of this approach to automated grading took place in the first two courses of the controls sequence which focus on modeling, system analysis, and controller design. The courses are primarily taken by upper division and first-year graduate students in electrical and mechanical engineering. Historically, these courses have included weekly, hand-graded homework assignments. These assignments have relied heavily on MATLAB functionality for calculations and plotting, but the majority of the work was performed with "paper and pencil." In the two pilot courses, approximately half of the homework problems (about three per week) were replaced with problems that required the students to write a MATLAB script to implement their solution. Each solution script was then automatically graded by checking the outputs of the script against a set of expected outputs for a predefined set of inputs.

A. The Cody Coursework Environment

In this work, the Cody Coursework environment from the MathWorks was employed for the authoring and automated grading of the students' MATLAB scripts. Cody Coursework provides built-in functionality to assist the instructor as well as the students.

Within the Cody environment, an instructor can create a course and individual assignments, each with their own due date. The instructor can then author problems to populate each assignment. Cody Coursework provides tools for problem authoring that allow the instructor to employ different fonts and bullets, to include images and equations (using LaTeX syntax), and to link to external web pages (like MathWorks help pages), in order to make a problem easier to understand for the students. An example problem statement in the Cody environment is displayed in Fig. 1. The instructor can then create a suite of test cases that can be alternately made visible to, or hidden from, the students. There are also several built-in MATLAB functions that can be employed in writing test cases that make checking student solutions simpler, while providing the students useful feedback. A few of these include: `assessVariableEqual`, `assessFunctionAbsence`, and `assessFunctionPresence`.

In the authoring process, the instructor can implement a problem solution that can be used for testing and is available to other instructors. The instructor can also include a template file where a starting point can be provided to the students. The template could provide comments as to which variables are available in the workspace, as well the outline of a solution, or the syntax for a MATLAB function the students are to use. Once a student has completed a solution, they then have the option to **Test** or **Submit** their solution. The Test button runs the student's solution against only the visible test cases, while the Submit button tests the solution against both the visible and the hidden test cases. The instructor can decide how many times the students are able to "submit" their solutions.

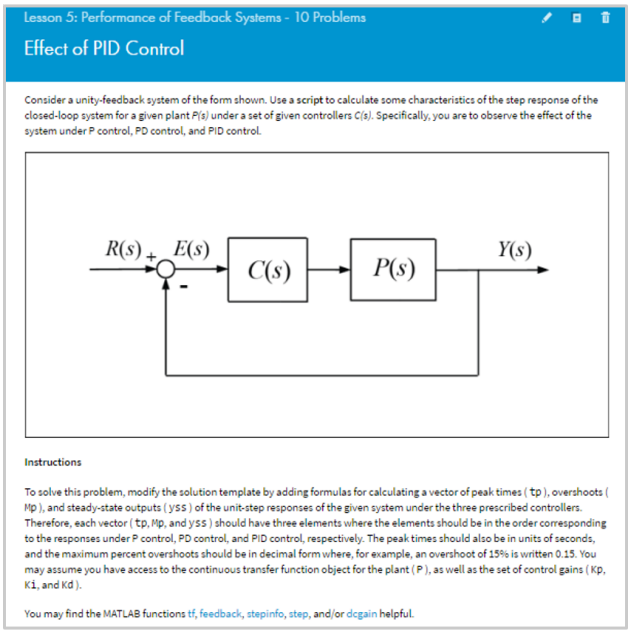

Fig. 1. Example problem statement in Cody Coursework

Cody Coursework further provides the instructor tools for grading and tracking class progress. The informatics tool available within the Cody environment is illustrated by the example visualization of student solutions for one assigned problem in Fig. 2. This tool displays each solution by its size in the order of its arrival. It also displays whether a submission is correct or not. This tool allows an instructor to quickly assess the success of the class in attempting a given problem, in real-time. Further, the tool can help identify solutions that are very similar (possible incidences of cheating) or that are outliers (possibly done in an unintended manner). Further, the instructor can inspect an individual solution by clicking on the corresponding marker in the tool.

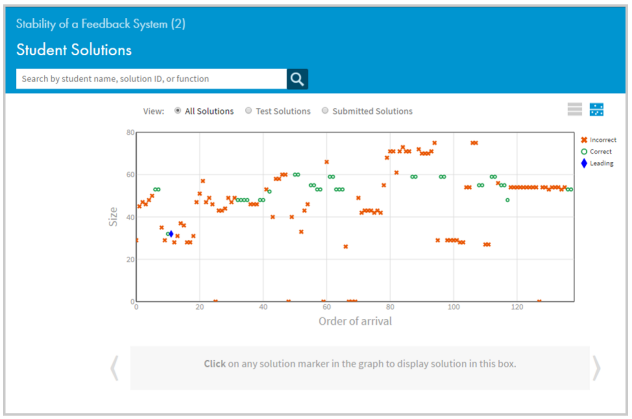

Fig. 2. Example of the Cody Coursework informatics tool

An instructor can also generate reports for a given problem which can assist with grading. A report can identify when a solution was submitted, how large it is, whether it is correct or not, and how many of the test cases were passed. An instructor can then assign credit based on whether the solution is correct, or can assign partial credit based on how many test cases were passed or based on the size of the submitted solution.

B. Possible Implementation Challenges

As with any new approach to pedagogy, there may be some resistance from the students. The student attitudes from the two courses in this pilot were mostly positive and are explored in detail in Section IV. One way to break down barriers with students is to spend a little time explaining the motivation for employing the approach. The use of new tools also may require some additional instruction. In our courses, the necessary instruction was minimal since the students were already familiar with MATLAB and because they found the Cody environment relatively easy to use.

Implementing assignments in this manner also presented some minor challenges that required a little thought to address. For one, since the grading is performed only based on the output of the students' scripts, students may not employ the instructed approach. One best practice is to always employ at least one hidden test case. This prevents students from hard-coding answers to the visible test cases in their scripts. Use of the MATLAB functions `assessFunctionAbsence` and `assessFunctionPresence` can also help to ensure that students don't use a prohibited MATLAB function, or alternatively, that they do use a particular MATLAB function. A final check is to use the Cody informatics tool to identify outlier submissions whose size indicates that the approach employed may be significantly different than the rest of the class.

Another challenge in writing problems for a software checker is capturing equivalent student solutions. For example, do the provided test cases allow for sufficient differences in tolerance and data type when testing outputs? The MATLAB function `assessVariableEqual` helps in this regard in that it allows tolerance to be set as an option, and further, it provides error messages to the user indicating when the two variables being compared are of different data type or size. Other examples of capturing equivalent solutions include angles and symbolic solutions. For example, the angles -90 degrees and 270 degrees are equivalent, but your test case may not identify them as such. One approach in that case is to compare the outputs of a trigonometric function, rather than the angles directly. Another example is comparing two transfer functions where one transfer function may have an extra common factor in the numerator and the denominator. In that case, you may need employ the `minreal` function to capture a pole-zero cancellation, or if the common factor is just a constant, you may just compare the transfer functions' poles, zeros, and DC gains, rather than comparing the transfer functions directly. Something that can put an instructor at ease is to remember that your solution and test cases don't have to work for all cases (positive/negative inputs, real/complex inputs, stable/unstable

systems, minimum/non-minimum phase systems), they only have to work for the cases you specify.

In the following section, examples of how to address some of these challenges, as well as how to provide useful feedback to the students, are detailed. More generally, the conversion of "traditional" modeling and controls homework problems to a form that can be automatically graded is discussed.

III. Integrating into the System Dynamics and Control Curriculum

Coding assignments are well-suited to grading by a software-checking program and have seen wide application of such tools. Solutions to some typical homework problems in the System Dynamics and Control curriculum also are straightforward to implement as MATLAB scripts; for example, problems that can be solved algebraically, or that simply involve calling MATLAB functions. Other types of modeling and control problems, however, are symbolic, or graphical, or more design-oriented, and require some re-framing in order for the students to implement their solutions as MATLAB scripts.

A. Algebraic Problems

Consider an example problem where the student is to calculate the control gains K_1 and K_2 for the feedback system shown in Fig. 3 in order to achieve a specific peak time t_p and maximum percent overshoot M_p for the system's step response.

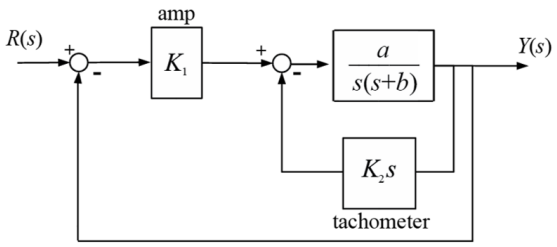

Fig. 3. Feedback system for an example homework problem

In this case, the closed-loop transfer function has the form of a canonical second-order system and the gains can be calculated as an algebraic function of the given plant parameters (a,b) and the given step response requirements (t_p,M_p). In this manner, the solution isn't much different than what a student would do with paper and pencil, they just need to express their calculations as a MATLAB script, such as the following:

```
wd = pi/tp;
zeta = sqrt(log(Mp)^2/(pi^2 + log(Mp)^2));
wn = wd/(sqrt(1-zeta^2));
K2 = (2*zeta*wn - b)/a;
K1 = wn^2/a;
```

Even with this simple example, the use of a software-checking tool can add value to the students by demonstrating how their solution is verified against the performance produced by the controller they designed, rather than by checking the controller itself. Furthermore, the visible test case provided to the students can also introduce them to alternative MATLAB commands. An example Test Case for this problem is shown in Fig. 4.

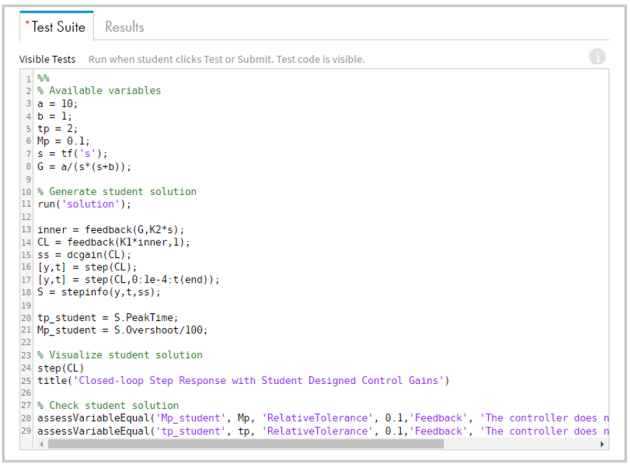

Fig. 4. Example visible test case in Cody Coursework

B. Symbolic Problems

Another class of problems common to modeling and controls courses require the generation of a symbolic solution. For example, the students may be asked to find the function that is the solution of a given differential equation, or to find the function representing the output of a dynamic system represented as a transfer function. One approach to these problems that would work within Cody Coursework would be to allow the students to use the Symbolic Math Toolbox for MATLAB. Many instructors, however, desire that the students learn (and demonstrate) how to generate the solutions themselves. In this case, one approach would be to provide the students the structure of the expected solution within the problem statement, and then have the students generate a script for calculating the parameters of the provided solution.

For example, consider that the the students are asked to symbolically solve the first-order differential equation

$$a\dot{x}(t) + bx(t) = \sin(\omega t) \tag{1}$$

for the given initial condition $x(0) = x_0$. Within the problem statement, the students could be provided the structure of the solution

$$x(t) = Ae^{rt} + B\sin(\omega t) + C\cos(\omega t) \tag{2}$$

and could be asked to write a script to calculate the parameters of the solution (r, A, B, C) based on specific coefficients of the given differential equation (a, b, ω). Again, the students will perform their solution primarily with

paper and pencil, and will only write the script to represent their final symbolic solution, such as in the following. This is consistent with the recommendation commonly made to students that they should perform their work symbolically, and should only substitute the numbers at the end of their solution process.

```
r = -b/a;
M = [b, -a*w; a*w, b];
result = inv(M)*[1 0]';
B = result(1);
C = result(2);
A = x0 - C;
```

Another common type of homework problem requires students to generate a symbolic mathematical model, for example, the governing equation(s) or transfer function representation of a physical system. In this case, it may not be necessary to provide the structure of the solution since we know the structure of a transfer function (or a linear differential equation). Rather, we can check the characteristics of the resulting model (coefficients, poles, roots of characteristic equation, etc.) for a given set of numerical parameters (mass, spring constant, capacitance, etc.).

For example, consider a problem where students are asked to determine the transfer function model of the system in Fig. 5 for an input of force f and an output of position y_1.

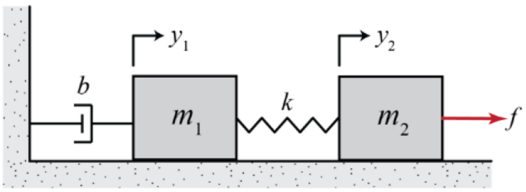

Fig. 5. Mass-spring-damper system for an example modeling problem

The MATLAB script solution submitted by the student would then just be a representation of the transfer function in terms of the available system variables, such as shown in the following. The script would then be tested against the poles, zeros, and DC gain of the "correct" transfer function model for a given set of numerical parameters.

```
num = k;
den = [m1*m2 m2*b (m1+m2)*k k*b 0];
G = tf(num,den);
```

C. Graphical Problems

Much of the analysis and design performed in control system courses requires the generation of graphs such as time response plots, frequency response plots, and the root locus. While a software-checking program can't exactly evaluate a graph, it can check the computation of individual points on a graph or specific aspects of a graph. Cody Coursework then allows the generation of plots corresponding to the calculations being evaluated as a visual means of providing feedback to the students.

For example, consider a problem that requires the students to generate the Bode magnitude plot for a given transfer function. In the test suite, you could provide as inputs the transfer function as well as a vector of frequencies at which the magnitude of the system is to be calculated. The output of the student-generated script would then be tested against the expected vector of magnitudes. An example solution is shown in the following.

```
for k = 1:length(w)
     top = polyval(num,i*w(k));
     bottom = polyval(den,i*w(k));
     mag(k) = abs(top/bottom);
end

semilogx(w,20*log10(mag),'*')
title('Bode Magnitude Plot')
xlabel('frequency (rad/sec)')
ylabel('magnitude (dB)')
```

In this test suite, numerous built-in MATLAB functions were disallowed, for example, functions such as `bode`, `freqresp`, `evalfr`, `frd`, and `idfrd`. Depending on the goals of the problem, an instructor could provide some starter elements within the solution template. For example, an instructor could provide the structure of the `for`-loop, or could provide the syntax for generating the Bode plot using the data calculated by the student's script. The Bode plot that is produced isn't evaluated by the software-checker, but it can provide the student insight. The output of the provided student solution is shown in Fig. 6.

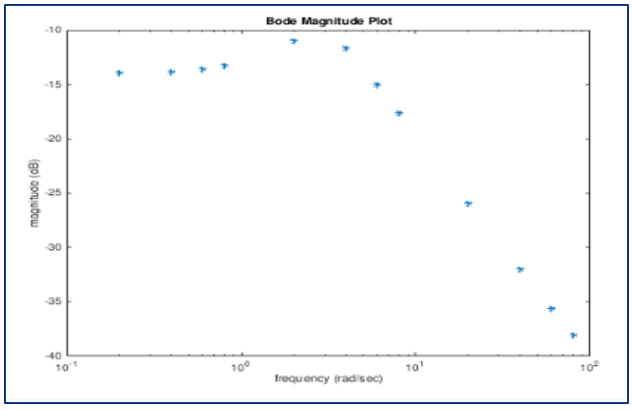

Fig. 6. Example output for a numerical graphical problem

An alternative to generating a plot numerically, point by point, is to have the students calculate important characteristics of a graph. In the case of a step response plot, this could include the overshoot, peak time, or settling time. In the case of a Bode plot, the students could be required to calculate a break frequency, crossover frequency, or resonant magnitude. For a root locus plot, the characteristics might be angles of arrival/departure, break-away/break-in points, or crossings of the imaginary axis.

Rather than having students generate a graph, it is also common to ask students to interpret a given graph. Within a

software-checking environment, this type of problem could be implemented by either providing as inputs specific characteristics of a graph (maximum overshoot, peak time, phase margin, etc.), or by providing vectors of data corresponding to a graph. In the case that the inputs are vectors of data, the students could be required to perform interpolation in order to identify the specific characteristics of interest. Alternatively, the students could be allowed to employ built-in MATLAB functions to determine important characteristics. Some such functions include `max`, `min`, `stepinfo`, `lsiminfo`, and `margin`.

The following script employs the `stepinfo` command to assist in identifying a canonical, underdamped, second-order system based on a given set of step response data (time (`t`) versus output (`y`)).

```
S = stepinfo(y,t,yfinal);

Mp = S.Overshoot/100;
tp = S.PeakTime;

wd = pi/tp;
z = sqrt((log(Mp))^2/(pi^2+(log(Mp))^2));
wn = wd/sqrt(1-z^2);
K = yfinal*wn^2;
```

D. Design Problems

Design of controllers, and systems in general, is another common type of problem found in the modeling and controls curriculum. If the system of interest does not have a simple, standard form, then these problems often involve some iteration where a parameter is tuned until the design requirements are met. Requiring students to generate a script that formalizes the rules for how this iteration is performed can actually be a benefit. Sometimes students solve these design problems with an element of guess and check, without truly specifying the details of their thought process.

In these design problems, it may be desirable to provide the students the structure of the logic for performing the required iteration. The following template provides the outline of a solution for designing a lead compensator, `C`, for a given plant model, `P`, in order to achieve a specific phase margin and level of steady-state error. An element that is added here is a condition for exiting the iteration if the solution does not seem to be converging.

```
i = 0; % loop counter

while((abs(Pm_goal-Pm)>0.1) && (i<100))
  C = ;
  [Gm,Pm,Wgm,Wpm] = margin(C*P);
  i = i+1;
end

if i == 100
  msg='Not solved within 100 iterations';
  error(msg)
end
```

E. Conceptual Problems

A type of problem that a software-checker is not very good for evaluating are more open-response problems. One option is to employ a multiple-choice format, but a software-checker isn't really needed for evaluating such problems. Another alternative is to have the students go through some process, then just draw their attention to the conclusion they are supposed to come to. For example, consider a problem that asks the students to generate a series of graphs as a parameter is varied and then notice a trend. In the test suite employed within Cody, the solution could generate a figure with a title that highlights the trend the student is supposed to notice. For example, the graphs shown in Fig. 7 demonstrate how the peak magnitude in a Bode plot decreases as the amount of damping in the system is increased.

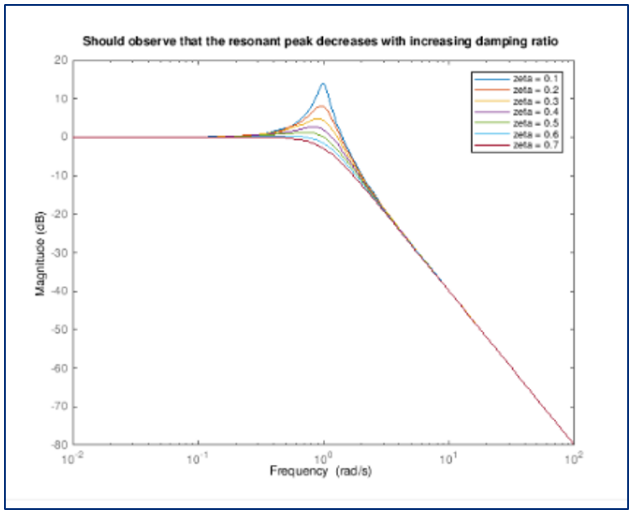

Fig. 7. Example output for a conceptual graphical problem

IV. Results from Pilot Courses

The application of this automated grading system was piloted in ELEE 5700: Controls II in the fall of 2017, as well as in ENGR 4220: Control Systems in the winter of 2018. ENGR 4220 is the introductory course on modeling and control and is primarily taken by undergraduate electrical, mechanical, and robotics engineering students in their junior or senior year. ELEE 5700 is the follow-on controls course and is primarily taken by first-year graduate students or senior-level undergraduate students as a technical elective.

Post-course surveys were obtained from 25 of the 31 students that were enrolled in these two courses. Only 3 of the obtained surveys were from students in ELEE 5700, with the remaining 22 coming from students in ENGR 4220. The background information obtained from the students indicated that, overall, they had pretty good programming background, but that they had minimal experience with automated software-checking programs. Specifically, all but one student had prior computer programming experience and all but two students had prior experience with MATLAB. Only 31% of the students had taken a programming course

where their assignments had been automatically graded by a software-checking program, while approximately 54% of the students had experience with an online homework system.

The students' attitudes towards the approach taken in the two courses were evaluated based on their responses to survey questions employing a Likert-type scale, as well as their opinions expressed via some open-response questions. Overall, it seemed that the students mostly found the approach beneficial, especially in the case that there wasn't a grader available to evaluate their paper and pencil solutions. In the following, the students responded employing a scale of 1 (strongly disagree) through 5 (strongly agree).

- The immediate feedback generated when submitting (incorrect) test solutions was helpful in debugging solution errors. (avg = 3.91)
- If I had a class where weekly homework assignments were not graded, I would appreciate the use of Cody-type problems. (avg = 3.91)

It also seemed that the students found the approach provided a side benefit of improving their programming skills.

- These problems helped me improve my understanding and knowledge of the MATLAB programming language. (avg = 4.05)
- These problems helped me improve/refresh my computer programming skills in general. (avg = 3.96)

It is not as clear that the students prefer this approach to more traditional "paper and pencil" type problems.

- I would like to see Cody-type problems employed in my other classes. (avg = 3.27)
- The solution of a Cody problem is more challenging than solving an equivalent "paper and pencil" problem. (avg = 3.68)

The free response questions primarily indicated that the students liked the overall approach and appreciated the immediate feedback. Some students, however, struggled with translating their paper-and-pencil solutions into a MATLAB script and were frustrated in debugging their solutions when they didn't pass the hidden use cases. It is the authors' opinion that some of these challenges could be mitigated as the problem statements and feedback provided to the students is improved.

V. Conclusion and Future Work

In this paper, motivation and guidelines for employing a software-checking program for the automated grading of assignments in the System Dynamics and Control curriculum were presented. In particular, techniques for converting typical homework problems that may be symbolic, graphical, and/or design-oriented in nature were discussed in the context of the Cody Coursework environment offered by the MathWorks.

The paper also presented results from the piloting of this approach in two course offerings at the University of Detroit Mercy in the Fall of 2016 and the Winter of 2017. Overall, anonymous surveys of student attitudes indicated that the students appreciated the use of the Cody Coursework tool and found it helpful to their learning. Future directions of this work include formally assessing the effect of using this automated grading tool on student learning outcomes.

During the course of this project, 90 homework problems were created in Cody Coursework for use in courses on the modeling, analysis, and control of dynamic systems. A copy of the current instantiation of this course on Feedback Control can be accessed from: `https://coursework.mathworks.com/courses/4431-copy-of-feedback-control`. The overall Cody Coursework Catalog available from the MathWorks also can be found at: `https://coursework.mathworks.com/catalog`. An updated version of this course on Feedback Control will eventually be added to the catalog.

References

[1] CollegeBoard, "Average rates of growth of published charges by decade," https://trends.collegeboard.org/college-pricing/figures-tables/average-rates-growth-published-charges-decade, accessed: 9-11-2017.

[2] J. Kay, P. Reimann, E. Diebold, and B. Kummerfeld, "MOOCs: So many learners, so much potential..." *IEEE Intelligent Systems*, vol. 28, no. 3, pp. 70–77, 2013.

[3] S. Iqbal, X. Zang, Y. Zhu, Y. Y. Chen, and J. Zhao, "On the impact of MOOCs on engineering education," in *MOOC, Innovation and Technology in Education (MITE), 2014 IEEE International Conference on*. IEEE, 2014, pp. 101–104.

[4] A. Chauhan, "Massive open online courses (MOOCs): Emerging trends in assessment and accreditation," *Digital Education Review*, no. 25, pp. 7–17, 2014.

[5] C. Piech, J. Huang, Z. Chen, C. Do, A. Ng, and D. Koller, "Tuned models of peer assessment in MOOCs," in *Educational Data Mining 2013*, 2013.

[6] L. C. Chirwa, "A case study on the impact of automated assessment in engineering mathematics," *engineering education*, vol. 3, no. 1, pp. 13–20, 2008.

[7] P. Ihantola, T. Ahoniemi, V. Karavirta, and O. Seppälä, "Review of recent systems for automatic assessment of programming assignments," in *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*. ACM, 2010, pp. 86–93.

[8] T. Staubitz, H. Klement, J. Renz, R. Teusner, and C. Meinel, "Towards practical programming exercises and automated assessment in Massive Open Online Courses," in *Teaching, Assessment, and Learning for Engineering (TALE), 2015 IEEE International Conference on*. IEEE, 2015, pp. 23–30.

[9] J. A. Rossiter, S. Dormido, L. Vlacic, B. L. Jones, and R. Murray, "Opportunities and good practice in control education: a survey," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 10 568–10 573, 2014.

[10] R. Hill, "Hardware-based activities for flipping the system dynamics and control curriculum," in *American Control Conference (ACC)*, 2015, pp. 2777–2782.

[11] R. M. Reck, "Applying a common framework to develop undergraduate control systems laboratory kits," in *Frontiers in Education Conference (FIE)*. IEEE, 2017, pp. 1–8.

[12] S. Brown and P. Knight, *Assessing learners in higher education*. Psychology Press, 1994.

[13] M. Richards-Babb, J. Drelick, Z. Henry, and J. Robertson-Honecker, "Online homework, help or hindrance? What students think and how they perform," *Journal of College Science Teaching*, vol. 40, no. 4, p. 81, 2011.

[14] M. L. Arora, Y. J. Rho, and C. Masson, "Longitudinal study of online statics homework as a method to improve learning," *Journal of STEM Education: Innovations and Research*, vol. 14, no. 1, p. 36, 2013.

[15] J. A. Cook and T. Samad, "Controls curriculum survey: A CSS outreach task force report," November, 5 2009.

[16] MathWorks, "MATLAB Cody Coursework," https://www.mathworks.com/academia/cody-coursework.html, accessed: 09-11-2017.