

Some uses for Computer-Aided Control System Design Software in Control Education

William S. Levine, *Fellow, IEEE*, and Dimitrios Hristu-Varsakelis, *Senior Member, IEEE*

Abstract—Several ways in which Computer-Aided Control System Design software and rapid prototyping tools have been used to enhance the controls educational program at our university are described. A key to the use in a lecture course has been to provide the students with a computer running the software during every exam. This motivates the students to learn to use the software and facilitates giving exam problems that focus on control design and analysis rather than on mechanical calculations. The use of computer-aided design software and rapid prototyping tools in the lab saves money and allows us to do more complicated experiments and simple projects. It also allows us to emphasize aspects of networked and embedded control systems that are not covered anywhere else within our curriculum.

I. INTRODUCTION

COMPUTER-design control system design (CACSD) software has greatly changed the practice of control system design. Tools such as CONDUCT[1] make it possible for control system designers to synthesize controllers that satisfy very complex and elaborate specifications. However, as evidenced by the available textbooks for undergraduate courses in control, the CACSD software has had relatively little impact on the standard controls curriculum. We have been experimenting for several years with the use of MATLAB/Simulink and several of their control system design adjuncts in our basic undergraduate controls course and in a separate controls laboratory course. This article describes what we have tried, what we think are appropriate uses of the software, and some indications of our successes and failures.

The reason for this article is to share information, start discussion, and, hopefully, to assist in the evolution of a consensus on how these new tools should change the curriculum. We are well aware that others are doing similar and related things and that some of their efforts are much better than ours. Some examples include [2]. We are also aware that some of our experience is unique to our institution and not necessarily applicable elsewhere.

The paper is organized as follows. The next section describes the context of our experiments and the background necessary to understand and evaluate what we have done.

Manuscript received January 19, 2006. This work was supported in part by the National Science Foundation under CRD Grant 0088081.

W. S. Levine is with the Department of Electrical and Computer Engineering, University of Maryland, College Park, MD 20742 USA (phone: 301-405-3654; fax: 301-314-9281; e-mail: wsl@eng.umd.edu).

D. Hristu-Varsakelis is with the Department of Applied Informatics, University of Macedonia, Thessaloniki, Greece

This is followed by a description of what we have done in our undergraduate course. We next describe our controls laboratory. We conclude with an assessment of our results.

II. BACKGROUND

The Department of Electrical and Computer Engineering (ECE) at the University of Maryland offers two undergraduate courses in control. The two courses must be largely independent because of some aspects of the overall undergraduate curriculum that are peculiar to Maryland. Firstly, state law requires that students be able to transfer from their local community colleges to our program at the end of their sophomore year without loss of time or credits. The community colleges teach all of the required courses in the first two years of the Department of ECE curriculum. The key beginning systems course (The title is Signals and Systems) covers linear time-invariant systems at an introductory level and is taught in the third (junior) year. This course is the sole prerequisite for the undergraduate controls courses.

The two courses are entitled, "Control Systems," and "Digital Control Systems," respectively. The control systems course is basically devoted to classical feedback control. It is this course that has been modified to take advantage of modern computer-aided control system design (CACSD) tools. The digital control course has also been the subject of considerable experimentation but not as much in the use of CACSD.

Undergraduate courses are largely linked to textbooks. While it is certainly true that innovative teachers can and do teach entirely from personal notes, this is relatively rare at the undergraduate level. Thus, the most commonly used textbooks largely define the syllabus. Two of the most popular undergraduate controls texts are Dorf and Bishop [2] and Franklin and Powell [3]. They cover pretty much the same material. Both include a fair amount of MATLAB but no serious use of CACSD software. As will be explained shortly, they do not make as much use of even the CACSD tools that are included with MATLAB as we believe is desirable. In fact, we have not found a textbook that really integrates computer-aided design tools into the subject. One reason for this article is to encourage this to change.

The controls laboratory course is available to seniors and graduate students in the electrical and computer, mechanical, and aeronautical engineering departments. It has also been

taken by students in the math and computer science departments. The course is jointly operated, supported and listed by the ECE and Mechanical Engineering (ME) Departments. This has several important advantages but it also introduces some complications. The main pedagogical advantage results from the ME students having considerably more laboratory experience than those from ECE while the ECE students generally have better theoretical training. By pairing an ME with an ECE we get the students to teach each other. There is also a substantial cost savings; more than twice as many students are taught for nearly the same cost. The major difficulty, not entirely due to the joint nature of the course, is that we cannot make a controls course a prerequisite for the lab. The other difficulties are minor and mostly administrative.

III. THE UNDERGRADUATE CONTROLS COURSE

As was explained earlier, the course is primarily devoted to classical feedback control. Thus, we have used both of the textbooks mentioned earlier. We have also experimented with a very modern approach to this subject based on the textbook by Goodwin, Graebe, and Salgado [4]. The main novelty in the course is that all the exams are given in a computer classroom—every student has a computer on his or her desk throughout the exam. The student has a choice of using the computer or pencil and paper to solve each exam problem. This arrangement was made in response to student complaints about an earlier version of the course in which the students were taught and encouraged to use MATLAB® in doing their homework but the exams were conventional pencil and paper tests. Their feedback was that they wanted the same tools available on the exams as for their homework.

The availability of software to plot root loci, Bode and Nyquist plots, and Nichols charts changes, at a minimum, the questions that one can sensibly ask on the exams. Of course, the real point is that this also changes the material that one should teach. Furthermore, using the single-input-single-output (SISO) control design GUI **sisotool** (We indicate MATLAB commands by bold-face type), and its analysis menu, the students can also watch, for example, as the step response of the closed-loop system changes in response to their changes in the controller parameters. This alters the role of the various design rules of thumb found in most textbooks.

The course is taught in a classroom that has a computer and a display setup so the instructor can demonstrate the use of the various tools. The students are encouraged to experiment on the computer on their own. On several occasions this had meant that the students introduced the professor to the latest MATLAB tool.

So far we have primarily described only the tools available to the students during exams. These same tools are also available when they do their homework. This does not require them to have their own computer. There are many computers available on campus for students to use when the

need arises. How does the easy availability of these tools change the syllabus?

There are two major changes. First, the amount of time spent on teaching students to plot root loci, Bode and Nyquist plots, and Nichols charts can be drastically reduced. Notice that we did not say omitted. One subject that is omitted is the Routh-Hurwitz criterion. Our students can simply use MATLAB to compute all the roots of any polynomial with real coefficients. Second, the time spent on control system design, and on deeper aspects of control, can be significantly increased. We still spend the first part of one lecture on the simplest rules for plotting root loci. We believe that this helps the students understand the meaning of the plots. Despite our best efforts, too many of them remain confused for a little while over open loop versus closed loop in interpreting the plots. This generally corrects itself after the first exam. We also give examples of incorrect plots including a case where the input data is correct. Specifically, we show them that MATLAB does not factor $(s + 1)^{15}$ correctly. The important question is how best to convey a deep understanding of the root locus. We do not believe that either plotting root loci by hand or knowing how to do so enhances understanding as much as knowing the limitations of the algorithms and seeing many root locus plots. We also ask our students to create “interesting” root locus plots. This gets almost no response.

We introduce the frequency response plots by briefly describing the basics of the Bode plot. Many of the students have seen Bode plots before in the signals and systems course but this apparently makes very little impression on them. We do discuss the plots for one and two-dimensional systems in normalized form. We believe this helps the students understand the idea and it previews the development of lead and lag compensators and notch filters.

We do not spend any time on drawing Nyquist plots. We do teach them the Nyquist stability criterion. We then use the MATLAB Nyquist plots to illustrate the perils of accepting automatically scaled plots as the truth. We show them, and assign homework that emphasizes this, that the automatic scaling frequently hides crucial features of the plot resulting in incorrect assessments of stability. The lesson is somewhat undercut by the ease with which students can check stability in other ways, specifically using the root locus. This could be fixed with some help from the Mathworks. What is needed is a way for the instructor to hide the system that produced the frequency response from the students while still giving them all the data contained in the plots. This would emulate the practical control design situations where the designer has an experimental frequency response but no analytical model.

The Nichols chart is an extremely useful design tool that is underemphasized in the usual undergraduate course because of the difficulties of drawing the plots. This need not be true any longer. We devote a lecture to explaining the Nichols chart and its use in determining the closed-loop

frequency response from the open-loop frequency response. Again, forcing students to work from frequency response data without access to an exact analytical model would enhance the lessons and the course. The reason this is necessary is that the students can exploit the analysis menu in **sisotool** to watch the changes in the closed-loop frequency response as they vary the parameters of the open-loop system if they have an analytical model. Allowing the instructor to create a plant that is unavailable to the students would facilitate homework and exam problems dealing with robustness, noise rejection, and modeling.

Notice that we have shifted the emphasis in the course away from how to create the needed plots. Instead, we devote considerable time to how to design SISO control systems. The CACSD tools allow us to emphasize three nonlinear aspects of controller design, all resulting from the simplest and most common nonlinearity, actuator saturation. These are integrator windup, conditional stability, and performance degradation. By checking the closed-loop transfer function between overall input and the input to the actuator, the students can see directly the consequences of placing the poles and zeros of their compensators too far apart. They can also easily check the different sensitivity functions. All of these are obtained from the analysis menu in **sisotool**.

We have also experimented with teaching the students how to do pole placement for single-input systems when the state is available. This gives a brief introduction to state space methods. The difficulty of finding a controller that does not require impractical gains tends to be discouraging to the students.

It should be apparent that we have taken a very conservative approach. We have not completely eliminated discussion of how to create the plots by hand. A case can be made for doing that. However, we do not believe there is much value in having students do even one plot by hand.

The emphasis on design in our course and the availability of the CACSD tools makes almost all of the problems in the common textbooks very simple for our students, one might say too simple. We create homework problems in which they are given a system in transfer function form and a set of specifications the controller must satisfy. We then ask them to create a satisfactory design. We also give “design” problems of the same sort on the second exam and the final. Grading is a bit of a nuisance because the answers are not unique. We check their answers by inputting them into **sisotool** and determining whether their controller meets the given specs, or not. Typically, many students overly complicate these design problems and produce controllers that have more poles and zeros than is necessary.

We have not performed a scientific assessment of the effect of these changes on student’s learning. Anecdotally, they are pleased with the sense of really doing engineering rather than solving academic problems. We emphasize for them that the course does not deal with many aspects of real

controller design, for example, with the choice of hardware.

IV. THE LABORATORY COURSE

The laboratory course begins with a sequence of standard experiments and ends with a small project. The students work in teams of two except when we have an odd number of students in the section. Then one team consists of three. We deliberately avoid larger teams because we have found that larger groups tend to include passive drones—students who do no work. Because large teams are a requirement for the accreditation of capstone design courses, the controls lab is not a capstone design course.

There is a one hour common lecture and three hours of lab each week. We try hard to avoid having students devote extremely large amounts of time to this course. We do not think it is good for them to sacrifice their other courses in the interest of completing their projects.

The standard experiments are intended to achieve 5 goals, to teach the students that the real system is not the same as its theoretical model and the most common and important differences, to teach the students how to function in a laboratory (i.e., to observe, to debug, to record, and to document), to familiarize the students with the available equipment, to show the students how to find and buy the appropriate hardware for their projects, and to introduce the students to networked and embedded control.

The four experiments are not at all novel. They were purchased from Quanser but could have been obtained from several other vendors. They are, in the order in which they are performed, the coupled water tanks, the heat flow, a simple servomechanism, and the rotary inverted pendulum.

All of the controllers are digital. In fact, they are implemented on a pair of PCs. The two PCs at each station share a keyboard and a display. One of the PCs is equipped with a Quanser MultiQ-PCI card for data acquisition and output. All of the computers in the lab are linked via a wireless communication network.

The controller normally is coded using MATLAB/Simulink on the PC that does not have a DAC card and then the code is compiled and downloaded onto the other PC using the MATLAB/Simulink Real Time Workshop (RTW). However, the code can be written and run on the DAC-equipped machine alone. In fact, the code can be downloaded to any of the DAC-equipped computers in the lab and that computer can run the control. We have had the students do this to emphasize the network aspects of the course and, occasionally because of a failed machine.

This approach makes it seem superficially to the students that their controller is a Simulink block diagram. This has the advantage that coding the controller is simple, intuitive, and in a language the students already know. This keeps the focus on the controller and off the coding. The disadvantage, eventually corrected by experience, is that students sometimes forget that they are dealing with computer code. We emphasize that none of this is

particularly novel, nor is it unique to our lab. Anyone can do this relatively easily and others have.

The coupled water tanks experiment is very well liked by the students. They do a detailed calibration of the sensor and actuator, identify the parameters of a nonlinear model of the system dynamics, design and implement a P and a PI controller, and implement several other linear controllers near a nominal operating point. This experiment is an especially good example of PI control with integrator windup. For many students this is the first time they have designed and implemented a system that they can see working.

The students do not like the heat flow experiment. The dynamics are very slow so it takes a long time to collect data and calibrate. We are not able to measure the actual temperature inside the tube so the control is to a voltage, not a temperature. The best part of the experiment is the measurement of the delay and the determination of a model.

The servomechanism experiment is also well-liked. It is the first one where we really emphasize digital control. We do this in three ways. First, we ask them to implement a safety shut off. If the input to the motor is not set to zero before the computer stops control, the rod on the motor shaft usually swings wildly when the computer does stop controlling the motor. Second, we have them do a system identification using MATLAB's **ident** tool. This is not great for the motors but it is an excellent introduction to system identification for the students. The resulting model is an ARX discrete time system. Lastly, we ask them to implement a discrete-time controller.

The rotary inverted pendulum is everybody's favorite. It is done in two parts. The students are given the linear quadratic regulator (LQR) solution which they implement using a crude estimator for the pendulum angular velocity. All the other state variables are measured. They are then given an intuitive idea of how to invert the pendulum and asked to design by themselves a controller that will invert the pendulum and then switch to the LQR when the pendulum is close enough to vertical. This is a switching controller so, very decidedly a hybrid controller. They work very hard at this. They need to rely on their experimental and analytical skills. It has taken some of the groups a long time to succeed. They have all been very highly motivated to do so and, so far, every group has eventually built a working controller.

Our university has an open house every year about the time the students complete the pendulum experiment. We invite them to demonstrate their controller. Many do and they clearly take great pride in what they have done. They also provide very nice explanations of how their controller works.

Student projects are a very important aspect of the course. The projects have two primary goals, to teach the students that a major part of control system design is to choose the sensors and actuators and to teach the students to properly

document their work. The very short time period available to complete the projects limits their complexity and heavily influences the course schedule.

Preliminary written proposals for the projects are due two weeks after the course begins. These are short, less than a page. They basically describe the system to be designed and built. We ask the students to implement their controller in the lab's PCs. This saves time, money, and simplifies the design. Because most of the students have no previous experience with the lab hardware it is very difficult for many of them to produce this preliminary proposal on time. We allow two additional weeks but we strongly encourage promptness. The student response has improved considerably over the years. Apparently, the grapevine warns them and they respond by arriving in the first class with some ideas. We have also noticed that the recent students are more likely to have found a project idea on the web than in the past. We encourage this.

Most of the preliminary proposals are too ambitious. It is notable that most of the students have no concept of how difficult and time-consuming it is to design and build a system. We discuss each proposal privately with the students and mutually agree on a project. We insist on systems projects unless at least one of the team has experience designing and building electronic or mechanical devices. That is, we ask the students to buy all the components of their project and assemble the system. Of course, they also have to create the controller on the computer and get it to work.

A detailed proposal is then due four weeks later. This proposal is expected to contain a detailed project plan and a parts list, with prices and suppliers. We need this early because of the lead time needed to purchase items through the university. Often, students use their own money to buy components for the lab in order to save time although we do not encourage this. Although they have all had a first year course that is supposed to teach them how to find parts and devices that can be bought for projects, we frequently have to demonstrate this.

The detailed proposals are reviewed immediately upon receipt. Occasionally changes are suggested. Components are ordered once the proposal is approved. The next step is to complete the design and build the system. This is not the last step. A final report is required. The instructions for the final report are to produce a document that would allow a student, similar to yourself, to duplicate your project exactly, including any experimental results.

An important aspect of the projects is that the students are given permission to fail. That is, their project does not have to result in a working system in order to be regarded as a success. This has important implications for the proposals and for the final reports. A proposal does not have to demonstrate that the planned project will succeed. We tell the students, and we expect, that only about 1/2 to 2/3 of the projects will result in a working system. As long as the

final report properly documents what went wrong and explains why the failure occurred the grade can be, and usually is, as high as if the project succeeded. Of course, a project that fails because of lack of effort by the students receives a poor grade. We have on several occasions used a failed project from the previous year as a project, building upon the prior experience.

What sort of projects can the students do in the available time? Almost every year a proposed project is to control a ball on a planar plate by controlling the corners of the plate. We suggest doing a ball and beam instead. Even this is hard for them to do in the allotted time. In fact, no one has successfully completed this project so far. However, each of the groups that have attempted this project has worked extremely hard, done creative and original mechanical design, found and bought unusual hardware, and, in my opinion, learned a great deal. This illustrates the benefits of allowing students to receive full marks for ambitious projects that do not fully succeed.

Another class of projects that are frequently proposed is some sort of robot car that tracks a given path. We suggest that they buy the car, install a sensor, and implement a computer control using the lab computers. Obstacle avoidance versions of this project almost always are successful. Path tracking ones are often successful but have a higher failure rate.

Despite the unhappiness with the heat flow experiment, many students propose some sort of temperature control project. For example, one group proposed to design and build a controller for the oil temperature in a french fry maker. We suggested that maintaining tight control over the temperature of water would be just as interesting and educational and a lot less messy and dangerous. Most of these projects succeed to some extent. Sensing and actuation tend to limit the achievable control.

One pair of students proposed a pH control using the apparatus for the coupled tanks experiment. The pH sensor proved to be a problem, especially after someone dropped and broke it. We plan to try this one again.

Several students have entered the class with very good experimental backgrounds. One, who was already working part time as an engineer designed and built a tiny automated walking bug. Although the project was interesting I don't think we taught him very much.

V. CONCLUSION

With regard to the lecture course, providing the students with a computer that has MATLAB during the exams has been an unqualified success. Because of this the students really commit to learning to use MATLAB. The focus of the course has shifted to control design and analysis and away from generating plots. The door has opened to much more sophisticated use of the computer and CACSD tools in the course.

With regard to the lab, the use of computer-aided rapid

prototyping tools has allowed us to do more complicated experiments and projects while keeping the cost of the lab comparatively low. We could have saved more money by building a lot of the experiments ourselves but the cost in terms of faculty time and effort would have been prohibitive.

As we have emphasized, we do not claim to have made any important innovation. Others have done much more exciting things in both the classroom and the laboratory. If you are one of them we would very much like to hear from you. We believe it would be very useful to exchange ideas and experiences relative to using modern CACSD tools in education. We also believe it is very important to do this.

To learn more about our controls lab, please visit www.isr.umd.edu/ISR/FacultyBios/Levine_bio.html and click on the undergraduate controls laboratory.

ACKNOWLEDGMENT

The laboratory course is the result of the combined vision and efforts of Gregory C. Walsh, Dimitris Hristu-Varsakelis and William S. Levine.

REFERENCES

- [1] M.B. Tischler, J.D. Colbourne, M.R. Morel, D.J. Biezad, K.K. Cheung, W.S. Levine, and V. Moldoveanu, "A multidisciplinary flight control development environment and its application to helicopter," *IEEE Control Systems Magazine*, vol. 19 (4), pp. 22–33, Aug. 1999.
- [2] <http://www.control.lth.se/education/processes/>
- [3] R.C. Dorf and R.H. Bishop, *Modern Control Systems (Tenth Edition)*. Upper Saddle River, NJ, Pearson Prentice Hall, 2005.
- [4] G.F. Franklin, J.D. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems (Fourth Edition)*. Upper Saddle River, NJ, Prentice Hall, 2002.
- [5] G.C. Goodwin, S.F. Graebe, and M.E. Salgado, *Control System Design*. Upper Saddle River, NJ, Prentice Hall, 2001.

