Enhancing Classical Controls Education via Interactive GUI Design

Raymond C. Garcia and Bonnie S. Heck

Classical control design methods regularly use simplifying approximations in the design; for example, root locus concentrates on the position of the dominant second order poles. These approximations are generally good for giving students some insight into the problem and for designing an initial compensator, but the effects of system zeros and higher order poles can make a significant difference in the transient response of the closed loop system. Designers must simulate the system and iterate on their compensator until they get a satisfactory response. Unfortunately, most classical controls textbooks gloss over this iteration procedure, concentrating instead on the initial design methods.

From a student's perspective, the iteration can be very confusing and laborious. On a root locus design, for example, students may try to decrease percent overshoot by either increasing or decreasing the gain in order to improve the damping ratio, but this line of action may not achieve the desired results if a zero is present in the system. Students can move a zero in the compensator, but do they move it to the left or to the right? There is a definite art in the design process. Students can only gain insight by performing many designs in order to gain a feel for how their modifications to the root locus or to the Bode plot affect the transient response. The standard way to perform these iterations is manually; that is, the students complete the design, then simulate the system, then calculate a new compensator, then resimulate, continuing until they get a satisfactory response. This procedure is laborious, so students may lose the insight needed.

Students would benefit greatly from an interactive tool with a graphical user interface (GUI) that automates much of the iteration and provides the appropriate images of the iteration process. One such tool, which has both root locus and Bode plot options available, is described in this article. For the root locus option, the root locus is displayed in one window and the corresponding closed loop step response is displayed in another window. The student can click on a compensator pole or zero on the root locus and drag it to another position. The root locus and the closed loop step response plots are updated in real time as the student is moving the pole or zero. A similar feature is available for Bode plot design where students can move compensator poles and zeros on
the open loop Bode plot and see how the corresponding closed loop step response changes in real time. This provides immediate feedback for students to gain understanding of how compensator poles and zeros can be moved to achieve the desired closed loop step response. This tool has been used at Georgia Tech to enhance the way that classical control is being taught. The classical methods are taught first, then this interactive tool takes the students a step beyond where the textbooks stop.

**Program Description**

This section gives a description of an event driven program LSLNR (short for Linear Systems Learner) that automates much of the iteration used in the design process. This program can be used for either the root locus design or the Bode plot design method. The main procedure is that students complete an initial design using the standard textbook design methods, enter their plant and initial compensator into the LSLNR window, then click on a pole or zero in either the root locus window or the Bode plot window and drag it to a new position. The closed loop step response, displayed in another window, updates automatically during the pole/zero move. The student can also click on different gain values and the closed loop step response updates accordingly. The tool also allows for new poles and zeros to be added to the compensator, thus making it an appropriate tool for the students' initial exposure to the root locus or Bode plots (in addition to its use for design).

Using GUI windows for "interactive learning" is gaining popularity, see for example [1] and [2]. This note describes an interactive learning tool for root locus and Bode design and shows how it can be used to enhance the classical control as it is now taught. The tool is available for free from www.ece.gatech.edu/users/192/education/LSLNR. It can be run using the Student Version of MATLAB 5.0 or MATLAB 5.0 with the Control Systems Toolbox. Other similar tools are available such as the MATRIXx product family (Xmath, SystemBuild, and RealSim to name a few by Integrated Systems) and MATLAB Professional Version Control Systems Toolbox version 4.1. The tool from The Mathworks Inc., called rtool, which serves as the best basis of comparison, shows corresponding Nichols and Nyquist plots but only performs the iteration on the root locus (not on the Bode plot). It does not update the root locus as the pole is being moved. Also, it currently is not available for the Student Version of MATLAB. However, any of the root locus examples described in this article can be performed using any of these tools. All the examples demonstrate how an interactive iterative tool such as these can be used to enhance textbook material.

Because of its GUI window, shown in Fig. 1, the LSLNR tool is intuitive to use. The window allows students to enter the plant and compensator coefficients (or poles and zeros). The Bode design or the root locus design options are selected by clicking on the buttons at the right. Clicking on the root locus button brings up the root locus and closed loop step response windows as shown in Figs. 2 and 3. The students can enter the compensator gain in the root locus menu by using the mouse to click on a root locus position or by entering a numerical value. The closed loop pole is marked with a red "+".

The values of the closed loop poles and the corresponding values of $\zeta$ and $\omega_n$ for a particular gain can be displayed by clicking on the pull-down menu item labeled "Statistics" in the root locus window. The step response window also has a pull-down menu labeled "Performance" where the user can select specific performance variables to display on the step response. The step response in Fig. 3 displays the maximum % overshoot and the 2% set-
Illustrative Examples

Three examples are offered that illustrate the iterative design technique and use of the LSLNR program in enhancing the classical control design methods as they are now commonly taught.

Example 1:
PI/Lag Design Using Root Locus

One of the standard design methods for PI or lag compensators is to choose the compensator pole and zero both near the origin to get the additional gain without shifting the uncompensated root locus to the right significantly, see [3]-[7]. The compensated root locus has a branch near the origin. This branch is generally ignored in the design because the corresponding pole has a small residue (due to the close proximity of the pole and zero). The designer picks the gain based on the "dominant pole(s)," which is actually further from the imaginary axis. Most textbooks stop the design at this point. However, the effect of the branch near the origin can be very significant when computing settling time; this effect is not seen until the time response is plotted. The designer can choose to change the gain or move the compensator zero in an iterative fashion, plotting the closed loop time response for each iteration.

Consider Example 7.17 from [7]: The plant is given by

$$G(s) = \frac{0.25}{s + 0.1}$$

Fig. 6. Root Locus for Example 1.

Fig. 7. Step Response for Example 1.

Fig. 8. Step response after iteration on Example 1.
The specifications are time constant no more than 2.5 sec and zero steady-state error to a step input. Using classical design methods based on dominant pole approximations, the PI compensator is chosen to be

\[ G_c(s) = \frac{1.231(s + 0.01)}{s} \]

The resulting closed loop poles are at \(-0.4006\) and \(-0.0076926\). The example in Phillips and Harbor ends at this point; however, simulating the response shows that the results are not very good and that iteration is necessary.

To use the LSLNR program for the design iteration, the plant and compensator of the initial design are entered into the Linear System Command Center window using the GUI shown in Fig. 5. Clicking on the root locus button brings up the root locus window and the closed loop step response window as shown in Figs. 6 and 7. The gain is entered in the root locus window. As seen from the step response shown in Fig. 7, the settling time of 291 seconds is very long due to the effect of the neglected branch near the origin.

A designer has the option of increasing the gain or moving the compensator zero. Using this GUI, the student can select the “Move” option from the root locus window (by clicking on the “Move” button). Once the button is clicked, the compensator zero can be dragged to a new location. The root locus and the closed loop step response are updated automatically during the move operation so that students get immediate feedback on their actions. For example, moving the compensator zero to \(s = -0.1\) (to cancel the plant pole) gives an excellent response as shown in Fig. 8. Note that the settling time has been reduced to 9.16 sec after the iteration. In addition, students can go back to the main LSLNR window and click on the Bode plot option if they would like to check the gain and phase margin or simply to correlate their root locus design with the resulting open loop Bode plot.

The reader is invited to try this program when designing PI or lag compensators for more complicated systems. For example, try a system with plant poles at \(s = -1\) and \(s = -2\) and no zeros. An initial lag design might place a zero at \(s = -0.1\) and a pole at \(s = -0.01\). The branch near the origin will usually present problems for the settling time, so that an iterative design method as demonstrated here is warranted. Most classical controls textbooks give very little attention to this problem with PI or lag compensators. Thus, this GUI program provides an excellent means of enhancing the standard “textbook” design approach for PI and lag compensators. The images and the immediate feedback enable students to gain insight into the design very quickly.

**Example 2: Lead/PD Compensation Using Root Locus**

In lead or PD compensation, zeros are added to the system in order to move the root locus to the left to speed up the response. An unwanted side effect is that zeros tend to increase the percent overshoot; thus, the system may exhibit overshoot even if the gain is chosen so that the closed loop poles have \(\zeta \geq 1\) as demonstrated in Fig. 2. One common procedure that mitigates this effect is to choose the zero to cancel a real plant pole (if stable). However, the general role of compensator zeros on the closed loop transient response
should be investigated by students as part of their design.

Consider Example 10.3 from [4]: The open loop transfer function is

\[ GH = \frac{K}{s^2} \]

With specifications of settling time \( \leq 4 \text{ sec} \) and percent overshoot (P.O.) \( \leq 35\% \), the compensator is chosen as

\[ G_c(s) = \frac{s + 1}{s + 3.6}, \quad K = 83, \]

so that the closed loop "dominant" poles are located at \(-1 \pm j2\). The resulting damping ratio of \( \zeta = 0.45 \) would seem to yield an acceptable percent overshoot. In fact, using the common approximation for percent overshoot given by

\[ P.O. = 100e^{-\frac{\pi}{\sqrt{1-\zeta^2}}} \]

yields a value of 21% for the present example. The plant and compensator are entered into the linear system command window and the gain is entered in the root locus window menu with the resulting root locus displayed in Fig. 9.

The actual step response, shown in Fig. 10, shows a P.O. of 47% (well above the 21% predicted by the dominant pole approximation). Again, the discrepancy stems from the presence of a third pole and a system zero (as stated by Dorf). The example in Dorf does not iterate further on the design, but suggests adding a prefilter to eliminate the effect of the zero.

However, simply iterating on the compensator pole and/or zero positions improves the response dramatically. (Adjusting the gain alone will not achieve the desired results.) In particular, a student can move the compensator zero to the right or the compensator pole to the left to improve the response. Again, the student simply clicks on the "Move" button in the root locus window and then clicks on the zero (pole) and drags it to another location. The root locus and the step response plots are updated automatically giving students immediate feedback on their actions. Figs. 11 and 12 show the result after moving the zero to \(-0.555\). Both overshoot and settling time specifications are met.

The reader is invited to try other PD or lead designs using this program; for example, consider the effect of the compensator zero position on the time response for the system shown in Figs. 2 and 3.

**Example 3:**

Lag Design Using Bode Plot

Consider a system given by

\[ G(s) = \frac{1}{(s + 1)(s + 2)} \]

The specifications are for steady state error to a step response to be less than 2% and for phase margin to be at least 50°. An initial design is to pick the compensator gain to be \( K = 98 \) to achieve the steady state error criterion, then pick the compensator zero to be at one decade below the frequency at which \( \angle G(j\omega) = -180 + 55° \). The pole is selected to be at the frequency at which the new Bode plot intersects the plot of KG with K=98. Using straight-line approximations in drawing the Bode plot gives a lag compensator as:

\[ G_c(s) = \frac{98(s/0.4 + 1)}{s/0.07 + 1} \]

This compensator and plant are entered into the LSLNR window with the resulting plots shown in Figs. 13 and 14. For reference sake, the uncompensated as well as the compensated open loop Bode plots are displayed. The actual phase margin is 37°, which leads to a percent overshoot of 30%. (Note there is a common approximation between phase margin and damping ratio of \( \phi_m = 100\zeta \).) The phase margin can be increased by moving the compensator zero and/or pole to the left, which is accomplished by clicking on the "Move" button and then dragging the pole or zero located on the bar under the frequency axis to a new frequency. For example, moving the zero to \( \omega = 0.2 \text{ rad/sec} \) and then the pole to \( \omega = 0.022 \text{ rad/sec} \) gives a phase margin of 50° with the corre-
sponding Bode plot and step response shown in Figs. 15 and 16.

It should be mentioned that the entire process could have been streamlined by skipping the straight-line approximation step (but retaining the same basic design procedure). In that case, the plant is entered as above but with the gain of 98. The initial compensator is chosen as 1 (1 in the numerator and denominator). A zero is added at \( \omega = 0.3 \text{ rad/sec} \) (following the guideline given above). The pole has to be selected in the range needed (say about a decade below the zero) and then moved until the specification is met. This procedure resulted in the pole at \( \omega = 0.03 \text{ rad/sec} \) with the same crossover frequency and phase margin as obtained above.

Summary

Iteration is a natural part of a control design procedure, yet it is not stressed in most undergraduate controls textbooks. In root locus design, students design a compensator using a variety of second-order dominant pole approximations for percent overshoot, settling time, etc. Since zeros and higher order poles in the system can have a significant effect on the system response, students should simulate the closed loop system and modify their compensator as needed to get the desired closed loop characteristics. However, most textbooks gloss over the iteration step, often stopping their numerical examples after the initial design. In Bode design, there are two major approximations; one is the use of straight lines for the Bode plot itself and the other is that there is only a loose connection between the Bode plot and the transient response.

This column presents a graphical and somewhat automated tool for performing the iteration when using classical control design methods. The graphical user interface is an easy way for students to see the effect of the gain and the compensator pole/zero locations on the closed loop step response. Clicking on poles and zeros and dragging them to new locations while seeing the closed loop step response update automatically provides an immediate feedback for students that allows them to gain insight into the iterative design process. Thus, the tool presented in this article extends the capabilities of classical control design methods as they are currently presented in textbooks. Three examples are given that demonstrate the use and effectiveness of this tool. Furthermore, the Bode plot and the root locus windows are tied together in such a way that students can see how changes to the root locus affect the open loop Bode plot and vice versa. Students who have used this tool in controls classes at Georgia Tech have given extremely positive feedback on it. The tool is available for free at www.ece.gatech.edu/users/92/education/LSLNR.

Bibliography