An Educational Java Applet for Linear Systems

Kent H. Lundberg and Brian F. Williams
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139

***Abstract*— We describe an educational web applet for students studying Linear Systems and Feedback Control. The applet allows students to manipulate a pole-zero diagram and immediately see the corresponding changes that occur in the Bode plot, Nyquist diagram, Nichols chart, and step response. The program is written in Java, so that it runs in any web browser, and it is free software, distributed under the GNU General Public License.**

I. Introduction

A web-based Java applet has been designed for students learning Linear Systems and Feedback Control. The applet helps students reinforce their s-plane intuition with respect to Bode plots, Nyquist diagrams, Nichols charts, and step responses. These plots are created and modified as the student manipulates a small graph area representing the s-plane. Poles and zeros can be added, deleted, and independently moved on the s-plane, with the resulting plots changing dynamically as the state of the pole-zero plot changes. This allows students to visually connect changes in pole or zero locations with the changes in system response.

This applet was inspired by the VisDyCon software [1], [2], which is written in C and runs under UNIX and X Windows. This program is written entirely in Java, allowing students to load it in any web browser with an up-to-date version of Sun Microsystems's free Java Virtual Machine [3]. Of course, Matlab, Mathematica, and Maple all provide more detailed tools to produce these same plots, but this applet can be loaded on any machine for free, has a small learning curve, and encourages student experimentation.

Figure 1 shows the layout of the applet window. Students can add, remove, or relocate system poles and zeros in the pole-zero entry frame. The output plot only displays one response at a time, but the student can switch between responses without affecting the current state of the s-plane. This feature allows students to see the relationships between any one system and all the different plots of frequency and time responses.

The applet treats a complex-conjugate pair of poles or zeros as a single entity, so when one of a pair is added, deleted, or moved, the conjugate pole/zero is also affected. Poles and zeros on the real axis are treated as a single real pole or zero, not as a pair. Multiple poles or zeros can occupy the same coordinate by adding or dragging additional poles to that location. They will appear as a single entity in the pole-zero entry frame, but the text list will show each one individually.

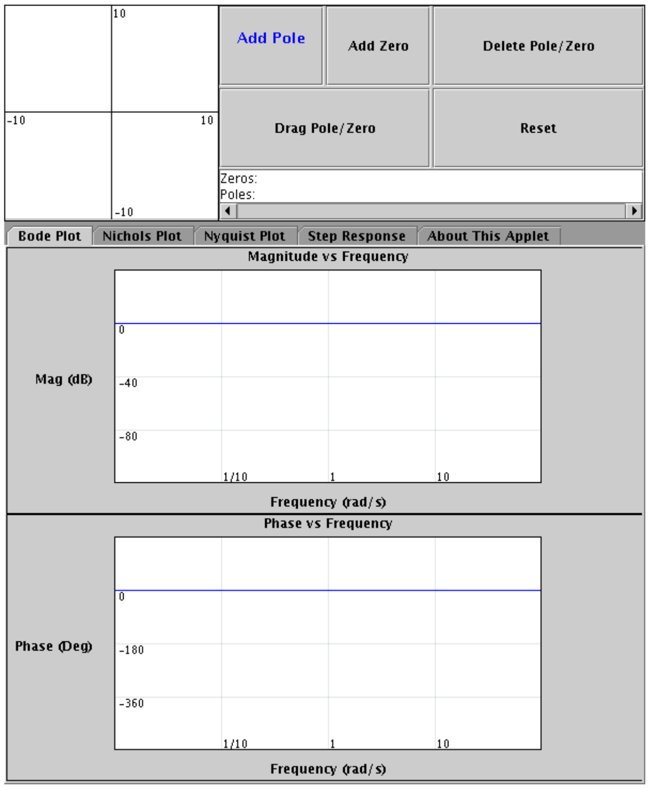


Fig. 1. Applet window layout, showing pole-zero entry frame (upper left), buttons for adding, deleting, and moving the system poles and zeros (upper right), and the current output plot (lower frame). Moving poles and zeros is accomplished by dragging them with the mouse. The text field below the buttons lists the pole and zero locations in $(x \pm yj)$ form.

The applet treats any pole or zero added or dragged within 0.5 units of the real axis as though it was on the axis. Even though a pole/zero being dragged will snap to the axis when it comes within that tolerance, it is still actively being moved and will un-snap if the mouse moves away from the axis.

The applet can currently be accessed with any web browser with the Java plugin from the applet home page:

```
http://web.mit.edu/6.302/www/pz/
```

Readers are encouraged to play with the program and explore its full capabilities.

II. Bode Plot

When the “Bode Plot” output panel is chosen, the output frame shows the frequency-domain Bode plot. Students can use the applet to explore how the locations of poles and zeros in the s-plane affect the frequency response. Figures 2 and 3 show the applet window before and after the addition of a transmission zero.

III. Nyquist Diagram

When the “Nyquist Plot” output panel is chosen, the output frame shows the frequency-domain Nyquist diagram. Students can explore how the locations of poles and zeros in the s-plane affect the polar plot. Figures 4 and 5 show the applet window before and after the addition of another system pole.

IV. Nichols Chart

When the “Nichols Plot” output panel is chosen, the output frame shows the frequency-domain Nichols diagram. Students can explore how the locations of poles and zeros in the s-plane affect the gain-phase plot. Figures 6 and 7 show the applet window before and after the relocation of a system pole.

V. Step Response

When the “Step Response” output panel is chosen, the output frame shows the time-domain response to a unit-step input. Students can explore how the locations of poles and zeros in the s-plane affect the step response. Figures 8 and 9 show the applet window before and after the addition of another system pole.

VI. Introductory Assignment

Here is a sample introductory assignment [4] to familiarize students with the applet. Once students are familiar with the applet, they are encouraged to use it to develop s-plane intuition.

A. First-order system

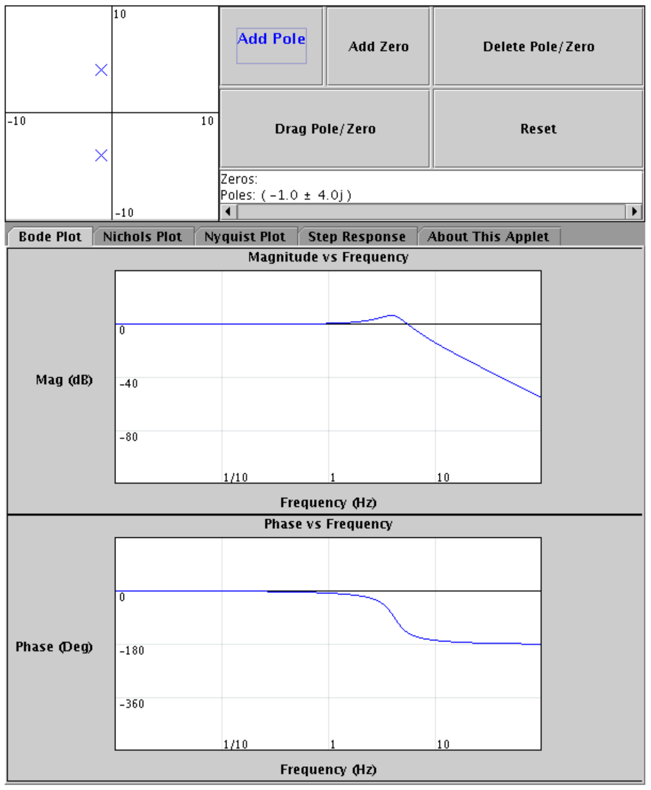
Clear the pole-zero map and add a pole to the negative real axis at $s = -5$ and select the “Bode Plot” tab. What is the corner frequency of this low-pass filter? How does the magnitude response differ from the corresponding asymptotic Bode plot for this system?

B. Second-order system

Reset and now add a pole at $s = -5 + 5j$. Why does the applet automatically add another pole at $s = -5 - 5j$?

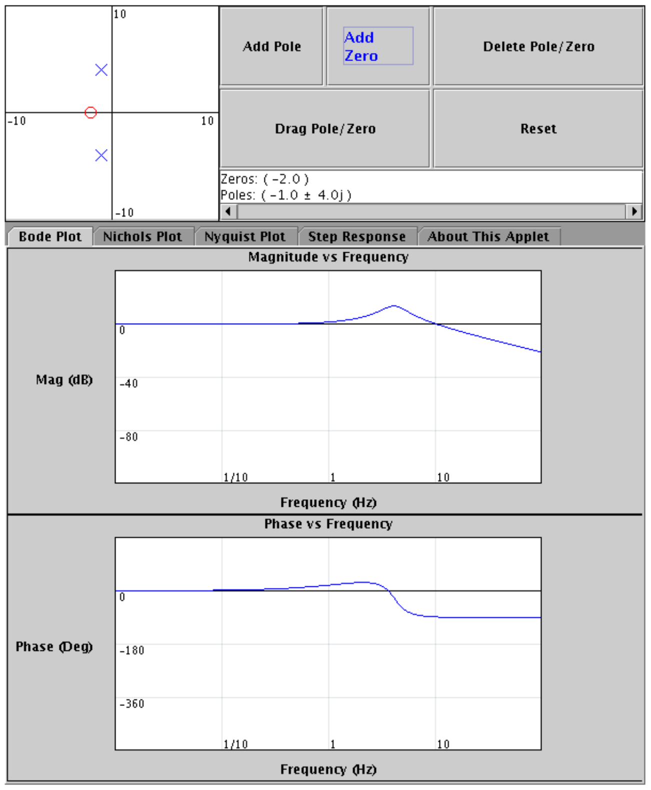
Drag the conjugate poles horizontally while keeping the x coordinate negative and the y coordinate approximately equal to 5. Does the corner frequency of the system change? What do you notice in the transition of the phase as $x \to 0$? How is this mirrored in the magnitude response?

Now look at the step response and put one pole at $s = -5$. Drag that pole vertically while maintaining the x coordinate. What happens to the step response?



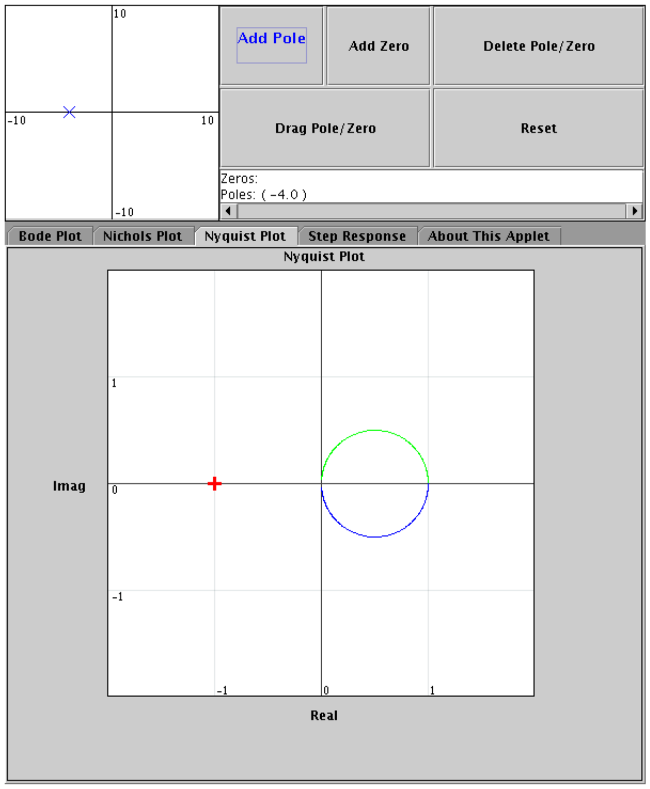
Zeros:
Poles: (-1.0 ± 4.0j)

Fig. 2. Bode plot output panel showing pole-zero plot and Bode plot of lightly damped second-order system.



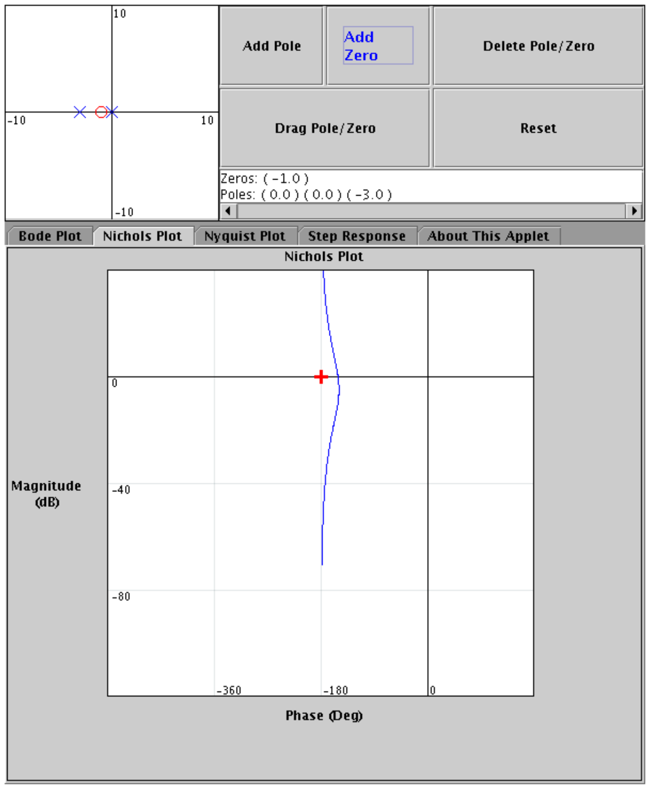
Zeros: (-2.0)
Poles: (-1.0 ± 4.0j)

Fig. 3. Bode plot output panel showing pole-zero plot and Bode plot of lightly damped second-order system with transmission zero.



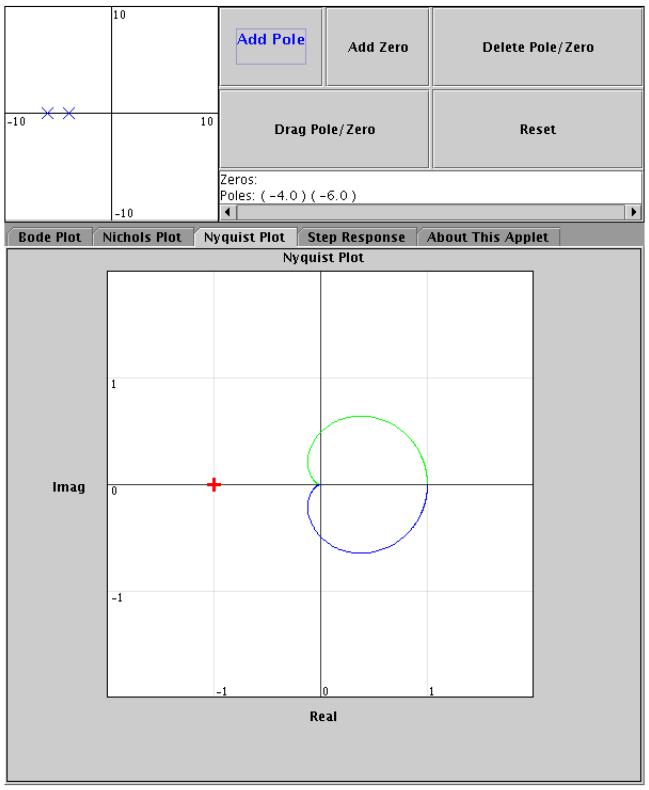
Zeros:
Poles: (-4.0)

Fig. 4. Nyquist plot output panel showing pole-zero plot and Nyquist diagram of system with one real-axis pole.



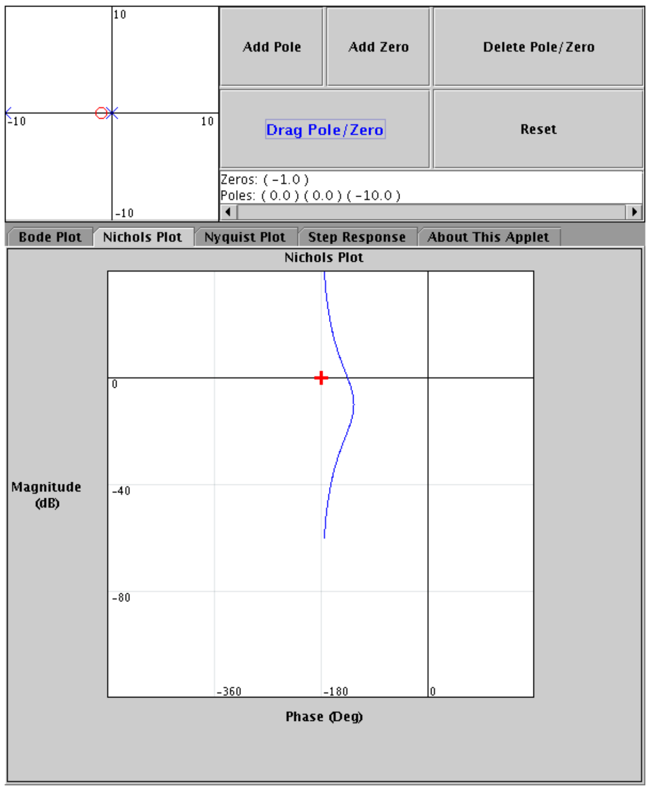
Zeros: (-1.0)
Poles: (0.0) (0.0) (-3.0)

Fig. 6. Nichols plot output panel showing pole-zero plot and Nichols plot of double-integrator system with lead compensator.



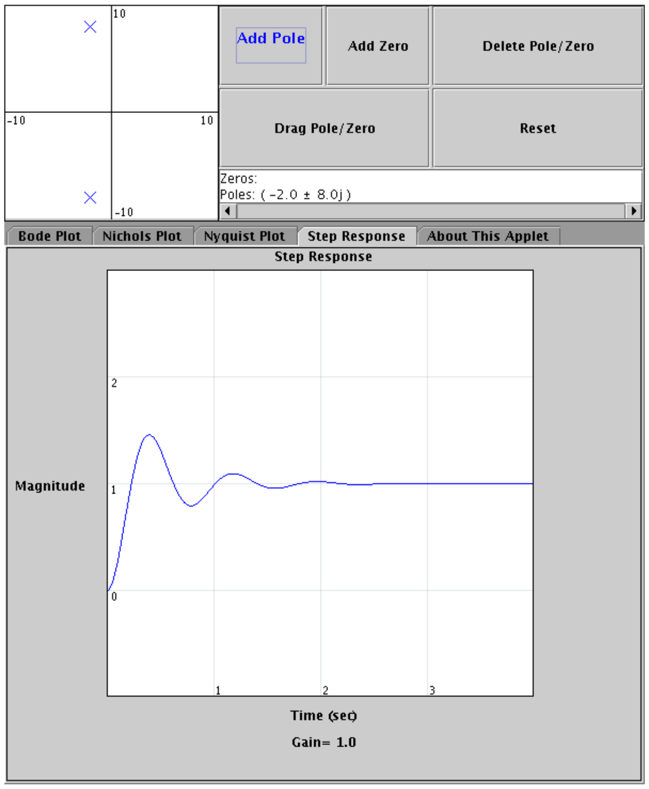
Zeros:
Poles: (-4.0) (-6.0)

Fig. 5. Nyquist plot output panel showing pole-zero plot and Nyquist diagram of system with two real-axis poles.



Zeros: (-1.0)
Poles: (0.0) (0.0) (-10.0)

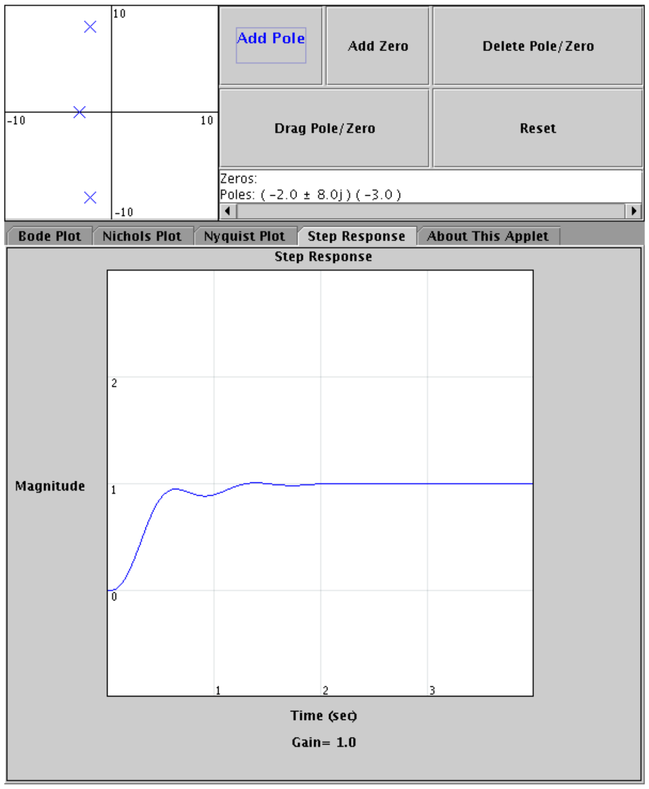
Fig. 7. Nichols plot output panel showing pole-zero plot and Nichols plot of double-integrator system with extended lead compensator.



Zeros:
Poles: (-2.0 ± 8.0j)

Gain= 1.0

Fig. 8. Step response output panel showing pole-zero plot and step response of lightly damped second-order system.



Zeros:
Poles: (-2.0 ± 8.0j) (-3.0)

Gain= 1.0

Fig. 9. Step response output panel showing pole-zero plot and step response of lightly damped second-order system with additional pole.

C. Poles and zeros

Create a system with one pole and one zero. Drag each along the real-axis and note how each singularity affects the step response.

For each combination you should first predict the initial and final values using the Initial-Value Theorem and the Final-Value Theorem of the Laplace transform, and then test your solution with the applet.

$$G_1(s) = \frac{0.2s+1}{0.125s+1}$$

$$G_2(s) = \frac{0.125s+1}{0.2s+1}$$

$$G_3(s) = \frac{0.2s+1}{1-0.2s}$$

$$G_4(s) = \frac{1-0.2s}{0.2s+1}$$

D. Lots of poles and zeros

Many times throughout feedback design you will encounter a system with singularities at undesirable locations. It is often tempting to negate these singularities by canceling them out. That may not always be a wise decision.

Create many poles along the negative-real axis in a linear fashion. You should have about 8-10 poles spread evenly in the end. Now, suppose you really wanted a system to respond like a system with the transfer function

$$G(s) = \frac{1}{0.1s+1},$$

and decide to cancel all the poles with a zero with the exception of the pole at $s = -10$. Use this method but don't take measures to line up all the cancellations exactly. What features do you notice in the step response and Bode plot that differ from the desired response?

VII. Further Assignments

Once students are familiar with the applet, assignments can be written that encourage them to investigate the interaction of Bode plots, Nyquist diagrams, Nichols charts, and step responses. In addition to straight-forward homework problems, assignments in the form of "Scavenger Hunts" can also be given, which encourage students to creatively explore various system responses:

1) Find a transfer function whose step response undershoots zero.
2) Find a transfer function, using poles only, that has a step response with greater than 100% peak overshoot.
3) Find a transfer function whose Nyquist plot encircles the -1 point once.
4) Find a transfer function whose Nyquist plot encircles the -1 point twice.
5) Find a transfer function whose Nyquist plot includes a loop-de-loop off of the real axis.

6) Starting with a double integrator $L(s) = 1/s^2$, find a simple transfer function with a Nichols plot that dodges the -1 point to the left.
7) Starting with a double integrator $L(s) = 1/s^2$, find a simple transfer function with a Nichols plot that dodges the -1 point to the right. What happens on the Nyquist plot in this case?

VIII. Conclusions

These applets allow students to manipulate pole-zero diagrams and immediately see the corresponding changes that occur in the Bode plot, Nyquist diagram, Nichols chart, and step response. By adding, removing, or relocating system poles and zeros in the s-plane, students reinforce their s-plane intuition.

Appendix I
Software Algorithms

The following appendices document the algorithms used for finding the step response. While these results are well known, neither we nor our colleagues were able to find a single reference that included complete instructions for numerically calculating the step response of an arbitrary linear transfer function. We document our collection of findings here for completeness, and we hope others will find them useful.

Appendix II
Some Matrix Math

To numerically calculate the step response, the system transfer function with a monic denominator

$$\frac{X}{U}(s) = \frac{b_1 s^3 + b_2 s^2 + b_3 s + b_4}{s^3 + a_2 s^2 + a_3 s + a_4}$$

is translated to a state-space description with the appropriate initial conditions. The state-space description is found from the locations of the transfer-function poles (the a_n in the denominator) and the dc gain of the numerator (b_4) as

$$\dot{\mathbf{x}} = \mathbf{A} \cdot \mathbf{x} + \mathbf{B} \tag{1}$$

where the states are

$$\mathbf{x} = \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix}$$

and the $\mathbf{A}$ and $\mathbf{B}$ matrices are

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -a_4 & -a_3 & -a_2 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ b_4 \end{bmatrix}.$$

The initial conditions are found from the locations of the zeros in the numerator. The feed-forward terms of the *observability canonical form* can be found from the first $n+1$ “Markov parameters” [5]:

$$\mathbf{H} = \mathbf{F}^{-1} \cdot \mathbf{G}$$

$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ a_2 & 1 & 0 & 0 \\ a_3 & a_2 & 1 & 0 \\ a_4 & a_3 & a_2 & 1 \end{bmatrix}^{-1} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}.$$

The Markov parameters can also be found from

$$\frac{b_1 s^3 + b_2 s^2 + b_3 s + b_4}{s^3 + a_2 s^2 + a_3 s + a_4} = H(s) = \sum_i^{\infty} h_i s^{-i}$$

For the step response, the initial state of the $\mathbf{x}$ vector is identical to these feed-forward terms

$$\mathbf{x}_0 = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}. \tag{2}$$

This differential equation (1) and initial condition (2) can be easily dumped into a fourth-order Runge-Kutta solver.

Appendix III
Fourth-Order Runge-Kutta

Runge-Kutta Four is a well-known numerical method [6] for solving differential equations of the from

$$\dot{\mathbf{x}} = \mathbf{A} \cdot \mathbf{x} + \mathbf{B}$$

by

$$\begin{aligned} \mathbf{x}_{n+1} &= \mathbf{x}_n + (\mathbf{K}_1 + 2\mathbf{K}_2 + 2\mathbf{K}_3 + \mathbf{K}_4)/6 \\ t_{n+1} &= t_n + h \end{aligned}$$

where h is the time-step size and the column vectors $\mathbf{K}$ are

$$\begin{aligned} \mathbf{K}_1 &= h(\mathbf{A} \cdot \mathbf{x}_n + \mathbf{B}) \\ \mathbf{K}_2 &= h\big(\mathbf{A} \cdot (\mathbf{x}_n + \mathbf{K}_1/2) + \mathbf{B}\big) \\ \mathbf{K}_3 &= h\big(\mathbf{A} \cdot (\mathbf{x}_n + \mathbf{K}_2/2) + \mathbf{B}\big) \\ \mathbf{K}_4 &= h\big(\mathbf{A} \cdot (\mathbf{x}_n + \mathbf{K}_3) + \mathbf{B}\big). \end{aligned}$$

These equations are iterated at each time step t_n.

Appendix IV
Matlab Code

The Matlab code shown in Figure 10 implements the above algorithms in Appendices II and III and demonstrates the simplicity of the implementation. This program served as a benchmark during testing of the Java applet.

Appendix V
Java Source Code

This work was inspired by the work of Will Durfee [1] and Wilson Rugh [7]. However, in contrast to these excellent projects, the source code for this applet is freely available, and we encourage other students and educators to download our code for any and all purposes. See the GNU General Public License [8] for more details.

The Java source code for this applet can be downloaded free of charge from the applet home page
`http://web.mit.edu/6.302/www/pz/`
The program includes code from Alexander Anderson’s complex number package [9] and the MathWorks and the National Institute of Standards and Technology’s matrix package JAMA [10]. Special thanks go to them.

```
clear
zlist = [1.5 2.5 ]';
plist = [-2-j -2+j -3 -4];
gain  = abs(prod(plist));
[num,den] = zp2tf(zlist,plist,gain);

N = length(den) - 1;
A = [ zeros(N-1,1) eye(N-1) ; ...
      -den(N+1:-1:2) ];
B = [ zeros(N-1,1) ; num(N+1) ];
for n = 1:N+1,
   F(n,:) = [den(n:-1:1) zeros(1,N+1-n)];
end
H = inv(F)*num';
X(:,1) = H(1:N);

% RK4: Runge-Kutta Fourth-Order Method

h = 0.01;
for n = 1:(4/h),
   K1 = h*(A*X(:,n)+B);
   K2 = h*(A*(X(:,n)+K1/2)+B);
   K3 = h*(A*(X(:,n)+K2/2)+B);
   K4 = h*(A*(X(:,n)+K3)+B);
   X(:,n+1) = X(:,n)+(K1+2*K2+2*K3+K4)/6;
end  % Shampoo, rinse, repeat.

t = 0:h:4;
plot(t,X(1,:))
grid
```

Fig. 10. Matlab code for calculating the step response. Starting with lists of the system poles and zeros, the transfer function is generated. From this transfer function, the state-space description (1) and the initial conditions (2) are found. These results are used by the Runge-Kutta fourth-order solver to numerically calculate the step response. The Java applet described in this paper uses this algorithm.

References

[1] W. K. Durfee, "VisDyCon: Visual dynamics and control," in *Proceedings of the 1991 IFAC Conference on Advances in Control Education*, Boston, 1991, pp. 128–131.

[2] M. B. Wall, "Interactive dynamics and control," Massachusetts Institute of Technology. [Online]. Available: http://web.mit.edu/visdycon/www/

[3] "Download Java technology," Sun Microsystems. [Online]. Available: http://www.java.com

[4] I. J. Dancy, "Educational hardware for feedback systems," Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, Sept. 2004.

[5] T. Kailath, *Linear Systems*. Englewood Cliffs: Prentice-Hall, 1980.

[6] R. K. Nagle and E. B. Saff, *Fundamentals of Differential Equations and Boundary Value Problems*. Reading: Addison-Wesley, 1986.

[7] W. J. Rugh, "Signals, systems, and control demonstrations," Johns Hopkins University. [Online]. Available: http://www.jhu.edu/~signals/

[8] "GNU General Public License," Free Software Foundation. [Online]. Available: http://www.gnu.org/copyleft/gpl.html

[9] A. Anderson, "JavaComplex.tgz," a complex number class for Java. [Online]. Available: http://www.netlib.org/java/

[10] "JAMA: a Java matrix package," The MathWorks and the National Institute of Standards and Technology. [Online]. Available: http://math.nist.gov/javanumerics/jama/