

Select MATLAB commands used in Teaching Applied Automatic Controls

G.V. Narayanan

University Of Toledo
Email: nara@utoledo.edu

The teaching of applied automatic controls for students in the engineering technology program is always a challenge in terms of imparting the mathematical knowledge and understanding of the control analysis and design. In this paper, the reasons behind this challenge are identified, and also, the use of select MATLAB commands in teaching applied automatic controls course is emphasized. Of the several useful commands available in MATLAB, only a few important commands are taught to enhance the understanding of the subject. The select MATLAB commands that are taught in the applied controls course are identified in this paper. In addition, this paper gives a sample second order system transfer system that is used to show and to explain the control design concepts in the class. The effect on the students understanding of the subject on such a teaching using these MATLAB commands is also discussed in this paper.

Introduction

In this paper, the subject of MATLAB commands pertaining to the control analysis and design is discussed. Several texts^{1,2,3} exist for teaching controls and use of MATLAB in controls^{2,3}. The teaching of the use of these select MATLAB commands in conjunction with the teaching of the control analysis and design is undertaken in the 'Applied Automatic Controls' course offered to the Engineering Technology (ET) students at the University of Toledo, Toledo. Due to the limitation of paper space, the discussion in this paper will be limited to select MATLAB^{4,5,6,7} commands, and as to how it helps in imparting the mathematical analysis and understanding of the control analysis and design.

There exist several difficulties both for students in terms of understanding the controls subject and for the instructor in imparting the mathematical knowledge required in controls to make students understand the subject within a semester time-frame. The 30 hour time-frame is small for the instructor to make the ET students understand and digest the subject materials in order to make them proficient in this subject. Such time-frame difficulty cannot be avoided even if the course teaching time is extended since this course is analytic-intensive and numerical in terms of the controls analysis and design. With no computational and software aids available, all students will get bogged down with many intermediate calculations needed while generating time-history and frequency domain

plots of the system responses. The MATLAB software and its control commands toolbox rescue the instructor and students in this course with many advantages for both, especially for the ET student. Of course, some special virtual laboratory time needs to be spent by students to get trained in the use of these select MATLAB control commands.

Thus, the ET student is trained in solving the control analysis/design problem without getting distracted with many intermediate calculations, and subsequently, the student's comprehension of the controls subject is much better. The student can then concentrate efforts in the overall design/analysis of a control system, leaving computational crunching to the virtual computer program (MATLAB). This increases the student's confidence and understanding of the controls subject. In addition, the benefits are quite a lot to the student after graduation from this course in the sense that the knowledge of the use of the MATLAB (software) as an assistant will help in the control analysis and design, without extra efforts or training, in his professional career.

Select MATLAB Commands

The select MATLAB commands taught in this control course are classified below according to the five important subject areas, for convenience of teaching. The five areas are chosen specifically to increase the students' ability to remember the commands, and to minimize the teaching difficulties of the control analysis and design.

The five categories of select MATLAB commands are:

- a) Blocks Reduction Commands;
- b) Time-History Commands;
- c) Frequency Domain Plots-Creation Commands;
- d) Second Order Control Design Command;
- e) Other Control Related Commands.

In this paper, to save paper space, the MATLAB output figures are not shown. These can be obtained easily by the reader or the student on a computer.

Blocks Reduction Commands

These commands are:

- 1) Series Command:
- 2) Feedback Command
- 3) Parallel Command

The 'series' command allows two or four blocks in series to be combined into a single block. The 'feedback' command reduces a closed loop of an open block with a feedback block into a single open block. The 'parallel' command allows two or four blocks in parallel to be combined into a single block. Figure 1-A illustrates the use of MATLAB 'series' and 'feedback' commands with an example. The elimination of a loop consisting of two blocks in series and a feedback block, as shown in Figure 1-B, is done using the

script shown in Figure 1-A. A multi-loop elimination can be done by a sequence of single-loop eliminations, similar to a single loop elimination shown in Figure 1-A.

```
% Application of the feedback function with unity feedback
% Elimination of one loop
numg=[1]; deng=[500 0 0]; sys1=tf(numg,deng);
numc=[1 1]; denc=[1 2]; sys2=tf(numc,denc);
sys3=series(sys1,sys2); % combine series blocks
sys=feedback(sys3,[1]) % convert closed into open loop
```

Figure 1-A: Elimination of a single loop example

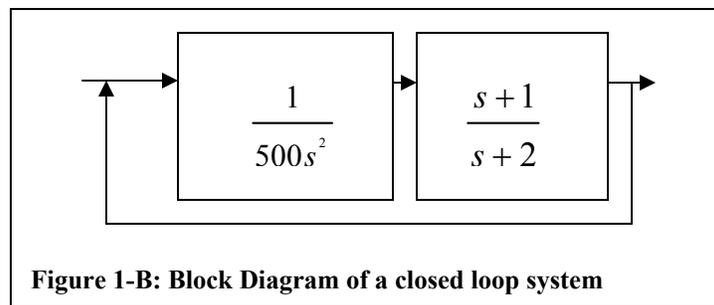


Figure 1-B: Block Diagram of a closed loop system

Time-History Commands

There are three general ways to create the time-history response of any system:

- Use of MATLAB built-in 'step' or 'impulse' command knowing the system transfer function;
- Use of MATLAB 'dsolve' command to solve the differential equation of system;
- Use of MATLAB 'plot' command on the analytical time response function of system.

```
% STEP example
num=[5400]; den=[2 2.5 5402];
sys=tf(num,den);
t=[0:0.005:3];
step(sys,t) %use 'impulse' command instead of 'step'
grid; xlabel('Time (sec)');ylabel('Wheel velocity')
title('STEP example')
```

Figure 2: Use of MATLAB 'step' command²

It is possible to solve the differential equation of any system, either directly or indirectly, for the system variable(s) in the time domain. If the system transfer function is known or derived from the differential equation, then the time response of the system due to step or

impulse input can be obtained directly by the use of MATLAB ‘step’ or ‘impulse’ command, as shown in Figure 2 with an example. However, the use of MATLAB ‘dsolve’ command can be used on any system differential equation to obtain the system variable time function in a symbolic (analytic) form. Such analytic function(s) can be plotted by the use of MATLAB ‘plot’ command. Figure 3-A illustrates the use of ‘dsolve’ command to obtain the system time response in a symbolic (analytic) form. Figure 3-B shows a script to plot the resulting analytic time function by the use of MATLAB ‘plot’ command. Also, in many situations, the MATLAB ‘ilaplace’ command can be used to obtain the analytic form of the system response time function. Such an example is not shown in this paper, but is taught in the class on how to obtain such a result.

```

% DSOLVE example 1
% Use of DSOLVE to get general solution to DE
% with unknown constants of integration (Initial conditions not
used)
dsolve('D2y+10*Dy+100*y=0')
%
% DSOLVE example 2
% Use of DSOLVE to get general solution to DE
% with integration constants computed from Initial conditions
% IC: at time t=0, y(0)=1 and Dy(0)=0
dsolve('D2y+10*Dy+100*y=0', 'y(0)=1', 'Dy(0)=0')
%
% DSOLVE example 3
% Use of DSOLVE to get Total solution to DE
% with integration constants computed from Initial conditions
% IC: at time t=0, y(0)=1 and Dy(0)=0 and
% particular solution for given forcing function f(t)=3sin(2t)
dsolve('D2y+10*Dy+100*y=3*sin(2*t)', 'y(0)=1', 'Dy(0)=0')

```

Figure 3-A: Use of MATLAB ‘dsolve’ command

Frequency Domain Plot-Creation commands

There are three general ways to obtain the frequency response of any system:

- a) Use of MATLAB built-in ‘bode’ command knowing the system transfer function;
- b) Use of MATLAB ‘laplace’ command to solve for the frequency response function;
- c) Use of MATLAB ‘plot’ command for the analytical frequency response function.

Since frequency domain plots in any controls analysis are very important, such plots can be done either using built-in MATLAB ‘bode’ command or using commands that perform calculations on and plotting of transfer functions. Figure 4 illustrates an example of the use of ‘bode’ command on a given transfer function example. The steps for direct calculations of the transfer functional values for various frequencies, and hence, creating the bode plot, are outlined in Appendix A. The use of ‘Laplace Transform’ on the differential equation of motion is discussed thoroughly in many mathematics texts,

and hence is not discussed in this paper. The MATLAB ‘laplace’ command can transform a given time function into its corresponding transform function.

```

% Spring-mass-damper forced response
% Forced Response to an zero Initial Condition
% Analytic time function for the response known, derived, or
given
k=2 % spring value
m=1 % mass value
wn=sqrt(k/m) % natural frequency of system
% user can use zeta=0.5 statement directly
%used b = 5, and calculated zeta using b=2*zeta*wn
zeta=1/(2*sqrt(k/m))
% given force function is a step function with 10N
f=10
%static displacement is f/k.
t=[0:0.01:5]; % time points
wd=wn*sqrt(1-zeta^2) % damped natural frequency
% System response analytic function, y(t)
y=(f/k)*(1-(wn/wd)*( exp(-zeta*wn*t).*sin(wd*t+acos(zeta))
));
% Plot forced Response
plot(t,y,'r'),grid % plot action with grids
xlabel('Time (sec)'),ylabel('y(t) (meters)')

```

Figure 3-B: Use of MATLAB ‘plot’ command for response

```

% To define ‘s’ as a symbolic variable, we define in MATLAB as a character symbol, as in:
>>syms s
% G(s)=80.8/(s^2 + 2s + 101) (given Transfer Function, for example)
% Now define TF in MATLAB and assign it in the ‘system’ variable, G
% (or in the feedback variable, H, if it is for feedback), as in this example1:
>>num = [80.8] % numerator of transfer function
>>den = [1 2 101] % denominator of transfer function
>>G=tf(num,den) %Use of MATLAB ‘tf’ function
%Now to create ‘bode’ plot in MATLAB with ‘omega’ limits, by using the ‘bode’ command:
%bode(G,{ $\omega_L$ ,  $\omega_U$ }) where  $\omega_L$  is the lower frequency limit, and  $\omega_U$  is the upper
frequency limit.
>>bode(G,{0.1,100000}) % typed with specific omega values

```

Figure 4: Use of MATLAB ‘bode’ command

Second Order Control-Design Command

Even though there exists several ways to do the control analysis, the unified approach of using the MATLAB ‘sisotool’ is taught to make it easy on students. The SISO tool is a very versatile and general approach for many Single Input Single Output (SISO) systems. The ‘sisotool’ use is illustrated in Appendix B with a second order system example for its control analysis and design.

Other Control -Related Commands²

Figure 5 illustrates the use of 'pzmap', 'zero', and 'pole' commands with an appropriate transfer function example. The 'ode45' command⁵ is also useful to obtain the numerical time response solution without resorting to the analytical system response functions. The use and discussion of 'ode45' command is outside the scope of this paper.

```
% MATLAB 'pzmap' command example for G(s)
numg=[6 0 1]; deng=[1 3 3 1]; % G=num/den
sysg=tf(numg,deng);
z=zero(sysg) % generate zeros of TF
p=pole(sysg) % generate poles of TF
pzmap(sys) % generate pole-zero map of TF
```

Figure 5: Use of MATLAB 'pzmap' command

Classroom Use of select MATLAB commands

The select MATLAB commands have been offered in the MET Controls course as a set of virtual laboratory exercises. There are about seven computer laboratory sessions used to present these select MATLAB commands, and the class room materials are presented according to schedule such that the virtual laboratory work is integrated for student experience of these subject materials.

During the first semester, the classroom materials and the MATLAB laboratory exercises were out of synchronization. This synchronization is now achieved in the subsequent semesters of the offering. Until now, it has been offered for four semesters including the current 2005 spring semester.

The testing of understanding of the control materials with the introduction of the laboratory exercises showed an improvement among students' understanding of the subject materials. There is definitely a success in the use of above materials in the MET Controls course. But a rigorous research was not done to assess the percent of success with the introduction of above select MATLAB commands virtual laboratory exercises.

Conclusions

The paper has been confined to the use of select MATLAB commands. The MATLAB help facility can provide additional information on the use of these individual commands. The paper has concentrated on the application of these select commands as is taught in the controls analysis and design course. The elimination of the distraction of intermediate calculations in the controls analysis/design helps all ET students concentrate and understand the important subject matter during the course. An additional advantage of learning the MATLAB control commands is with regard to his increased ability to design any control system in his professional career, without getting bogged down with

many numerical calculations. Of course, the use of MATLAB commands did boost his ability to perform numerical calculations needed in other areas of his engineering study.

References

- 1) Bateson, R. N., "Introduction to Control System Technology", Second Edition, Prentice Hall, 2002
- 2) Dorf, R. C., "Modern Control Systems", Ninth edition, Prentice Hall, 2001
- 3) Kuo, B. C. and Golnaraghi, F., "Automatic Control Systems", Eighth Edition, Wiley, 2003
- 4) Chapman, S. J., "MATLAB Programming for Engineers", Second Edition, Brooks/Cole, 2002
- 5) Magrab, E. B., Azarm, S., Balakrishnan, B., Duncan, J., Herold, K., and Walsh, G., "An Engineer's Guide to MATLAB", Prentice Hall, 2000
- 6) Etter, D. M., Kuncicky, D. C., and Hull, D., "Introduction to Matlab 6", Prentice Hall Engineering Source, 2002
- 7) Kuncicky, D. C., "MATLAB Programming", Prentice Hall Engineering Source, 2004

Dr. G.V. Narayanan teaches at the University of Toledo, Toledo, Ohio. He can be contacted by email at 'nara@utoledo.edu'.

Appendix A: How to Create Bode Plots

Given Transfer Function (TF) in the rational polynomial form, denoted by $G(s)$. For example, the function $G(s)$ is given by:

$$G(s) = \frac{80.8}{s^2 + 2s + 101}$$

Steps to create Bode plot are: (Do not forget to insert the semi-colon at the end of the MATLAB command when you compute thousands of values as a vector)

- 1) Set the complex variable, $s = i\omega$ in $G(s)$; now $G(s) = \frac{80.8}{(i\omega)^2 + 2(i\omega) + 101}$
- 2) Open MATLAB
- 3) Define ω values in the range $\{0.1, 100000\}$; In MATLAB, type
`>>omega = [0.1:1:100000];`
- 4) Compute $G(i\omega)$ for these values; In MATLAB, type
`>>G = 80.8 ./ ((i*omega).^2 + 2*(i*omega) + 101);`
- 5) Compute Magnitude values of these G (complex) values; In MATLAB, type
`>>mag = abs(G);`
The MATLAB 'abs' command is used to compute magnitude values instead of `sqrt(real(G).^2+imag(G).^2)`
- 6) Compute DB equivalents of Magnitude values; In MATLAB, type
`>>MDB = 20*log10(mag);`

- 7) Create a Semi-Log plot of MDB vs. omega, with omega along log x-axis;
`>>semilogx(omega, MDB)`
 The plot of the magnitude of G(s) in DB format is obtained.
- 8) Since omega values are defined for same frequency values, these same frequency values are used for Phase values computation. To compute phase values, type in MATLAB:
`>>phase = angle(G)*180/pi;`
 The MATLAB 'angle' command is used to compute atan(imag(G)/real(G)).
- 9) Create a Semi-Log plot of Phase vs. omega;
`>>semilogx(omega, phase)`
 The plot of the phase of G(s) is obtained.

Note: Both plots can be shown in one figure, and such plots are not shown here in this paper.

Appendix B: How to use MATLAB 'sisotool' in a control design

Problem³: Consider the second order model of the aircraft attitude control system. The forward path Transfer Function (TF) of the system is given by:

$$G(s) = \frac{4500K}{s(s + 361.2)}, \text{ with } K = 181.17$$

Determine the optimum values of the parameters, K_p, K_D , of a PD controller connected in series to the above forward path TF in a Unit Feedback closed loop system that will meet the following performance specifications:

In Time Domain:

- a) Maximum Percent Overshoot $\leq 5\%$
- b) Rise Time $\leq 0.005\text{sec}$
- c) Settling Time $\leq 0.005\text{sec}$

In Frequency Domain:

- d) Phase Margin $\geq 80^\circ$
- e) Resonant Peak ≤ 1.05
- f) Frequency Bandwidth $\leq 2000\text{rad / sec}$

Solution:

A properly designed PD controller can affect the performance of a control system in the following ways:

- 1) Improving damping and reducing maximum overshoot (PO);
- 2) Reducing rise time (t_r) and settling time (t_s);

- 3) Increasing Frequency Bandwidth (BW);
- 4) Improving Gain Margin (GM), Phase Margin (PM), and Magnitude at resonant frequency (M_r);
- 5) Possibly accentuating noise at higher frequencies;
- 6) Possibly requiring a relatively larger capacitor in circuit implementation.

The ACSYS Frequency Tool³ is used to solve this design problem. ACSYS helps in the easy input of transfer function. Of course, ACSYS uses the MATLAB 'sisotool' and passes transfer function to this sisotool.

There are three major steps in the control system analysis and design:

Step1: Determine the specification quantities for the uncompensated given system.

For uncompensated given system: $K_p = 1$ and $K_D = 0$

The closed loop TF is written as:

$$G(s) = \frac{815,265}{s^2 + 361.2s + 815,265}$$

If the analytical formulas are used, one first finds the damping value of this given system. Damping ratio, ζ , is given by the expression: $2\zeta\omega_n = 361.2$,

where $\omega_n = \sqrt{815,265} = 902.92 \text{ rad/sec}$. Hence, $\zeta = \frac{361.2}{2(902.92)} = 0.2$.

Percent Overshoot for this closed loop without controller is given by:

$$PO = 100e^{\frac{-\pi\zeta}{\sqrt{1-\zeta^2}}} = 100e^{\frac{-\pi(0.2)}{\sqrt{1-(0.2)^2}}} = 52.66$$

Settling time is given by: $t_s = \frac{2}{\zeta\omega_n} = \frac{2}{(0.2)(902.92)} = 0.0111 \text{ sec}$ (approximate estimate)

All these values can easily be obtained by using ACSYS. (The following computing process of using ACSYS is repeated as many times, as is required, for various values of K_p and K_D in the control system analysis.

A short summary of 'How to Use ACSYS for determining the specification quantities that we are interested in, namely GM, PM, Gain CO, BW, M_r , t_r , t_s , etc.' is given below: (The several mini-steps of using the frequency Response tool of ACSYS are discussed below.)

- 1) Open ACSYS³;
- 2) Click on the Frequency Response Tool;

- 3) Enter the TF for $G = \frac{4500}{s^2 + 361.2s}$ and $G_c = 181.17$ and click 'Calculate' button
- 4) Click on 'Bode' button, and wait for a few seconds for MATLAB to open the 'sisotool'. A 'bode' plot is obtained for closed loop TF of the given system.
- 5) Fill in specification values obtained for various K_D values in a table format as shown in Table 1 below;
- 6) First record specification values for the parameter $K_D = 0$, values shown on Bode plot for PM, GM, and Gain Cross Over (CO).
- 7) To get PO, rise time and settling time, click the 'Response to Step Command' option under the 'Analysis' menu. A time history plot window is obtained now. Right click inside the window and choose the characteristic options, namely, 'peak response', 'rise time' and 'settling time'. Three Blue dots can be seen on the blue curve. Move cursor over these dots to read the values, and record on the table for PO, rise time, and settling time under $K_D = 0$.
- 8) Now, to get M_r and BW values, click the 'Closed Loop Bode' option under the 'Analysis' menu of the SISO window. A Bode plot for the closed loop is drawn in a separate window or together with the previous time history plot. Right click inside the closed loop bode plot and choose the characteristic option, 'peak magnitude'. A Blue dot is seen on the Blue curve of the magnitude plot. Move cursor over the dot to read the peak magnitude in decibel (DB). Convert the decibel magnitude to absolute value with the relation $10^{\left(\frac{\text{peakDB}}{20}\right)}$, and record this M_r computed value.

Step2: Determine the specification quantities for the forward path transfer function of the PD compensated system

The forward path transfer function of PD compensated system (meaning that PD control mode controller block is in series with the given system block) is given by:

$$G(s) = \frac{4500K}{s(s + 361.2)}(K_p + K_Ds), \text{ with } K=181.17.$$

(Note that PD controller is in series, so the transfer function get multiplied)

The uncompensated system is for parameters with $K_p = 1$ and $K_D = 0$. First, find the 'limiting' K_D value of the compensated system (with $K_p = 1$) below which the system will become unstable. This can be obtained by looking at the damping ratio of the compensated system.

If the closed loop transfer function of the compensated system is written with K_p and K_D symbol values, the following expression is obtained:

$$G_c = \frac{815,265(K_p + K_D s)}{s^2 + (361.2 + 815,265K_D) + 815,265K_p}$$

Please note that the symbol G_c is used to represent TF of the compensated system. Hence, the damping ratio of the compensated system is given by:

$2\zeta\omega_n = 361.2 + 815265K_D$. We have, $\omega_n = \sqrt{815,265K_p}$ rad / sec and so, we get

$$\zeta = \frac{0.2 + 451.46K_D}{\sqrt{K_p}}$$

For system to be stable, we need ζ to be greater than zero.

So, for $K_p = 1$, the numerator $(0.2 + 451.46K_D) \geq 0$, so $K_D \geq -0.000443$ for the compensated system to be stable. Hence, for system stability, this damping expression suggests K_D and K_p must be greater than zero. We will assume $K_p = 1$, and determine the value of K_D that meets all specifications.

As a first guess, design the compensated system such that the damping ratio will have a critical value of 1. For this value, the K_D value can be computed from

expression $\zeta = 1 = 0.2 + 451.56K_D$. This gives $K_D = 0.00177$.

Now, one chooses one value above and another below this computed K_D value. Thus, say, three K_D values are chosen, namely, 0.0005, 0.00177 and 0.0025, and see if any of these will meet all our specifications.

For each of these values, the ACSYS frequency tool is used to determine the specification quantities as shown in Table 1 below.

Step 3: Final Design values of K_p and K_D

One can see from the tabulated values of the specified quantities, the K_D value of 0.00177 meets all the specifications.

With this value of $K_p = 1$ and $K_D = 0.00177$, the operational amplifier circuit for the PD controller can be designed. A circuit with resistors and capacitors is designed to get these same values of K 's that meets specification values. Such a discussion is outside the scope of this paper, and the details can be found in Ref. [1,3].

The following final design plots (in addition to the other plots) are needed:

- 1) Time Domain plots for Uncompensated and Compensated design in one plot, with PO indicated.
- 2) Bode plots for Uncompensated and Compensated design in one plot with phase margin indicated.

Table 1: Specification Quantities Table for $K_p = 1$ and various K_D values³

K_D	GM (db)	PM (deg)	gainCO rad/sec	BW rad/sec	M_r	t_r (sec)	t_s (sec)	PO (%)
0	∞	22.6	868	1370	2.522	0.00134	0.0217	52.7
0.0005	∞	46.1	913	1326	1.381	0.00144	0.0088	25.7
0.00177	∞	82.9	1502	1669	1.025	0.00119	0.0049	4.2
0.0025	∞	88.9	2046	2083	1.000	0.00103	0.0017	0.7

The MATLAB ‘bode’ and ‘time’ plots for various K_D are not included with this paper to save paper space. However, it is easy to obtain these plots on a computer by the student or the reader.