

# Aerocontrols Matlab Toolbox

**created by: James J. Sorenson**

**sponsored by: Daniel J. Biezd**

Revision 1.2: November 11, 2000

Aero Department of the California Polytechnic State University

# Table of Contents

|                                                                    |    |
|--------------------------------------------------------------------|----|
| <i>How To Use This Manual</i> .....                                | 4  |
| <i>Installation</i> .....                                          | 5  |
| <i>Create a Transfer Function</i> .....                            | 6  |
| ssetup: to create transfer functions.....                          | 6  |
| shf: to create shorthand transfer functions .....                  | 7  |
| maketf: to create transfer functions .....                         | 8  |
| tfmodel: to create transfer functions from Simulink diagrams ..... | 11 |
| ezpade: a time-delay approximation .....                           | 12 |
| <i>Display a Transfer Function</i> .....                           | 14 |
| tf: for polynomial notation.....                                   | 14 |
| zpk: for the Zero-Pole-Gain notation.....                          | 14 |
| sho: for the Shorthand notation .....                              | 15 |
| tc: for the Time-Constant notation.....                            | 15 |
| <i>Bode Plot Functions</i> .....                                   | 16 |
| ezbode: for creating bode plots .....                              | 16 |
| menubode: for creating/manipulating bode plots.....                | 18 |
| Add a plot in "menubode" .....                                     | 19 |
| Display and Label Coordinates in "menubode" .....                  | 21 |
| Zoom in and out of plots from "menubode" .....                     | 24 |
| Gain and Phase Margins within "menubode" .....                     | 25 |
| bodeslope: for Calculating the Slope of a Bode Plot.....           | 29 |
| <i>Root Locus Functions</i> .....                                  | 31 |
| ezrlocus: for creating bode plots .....                            | 31 |
| menurlocus: for creating root locus plots .....                    | 33 |
| Add a plot in "menurlocus" .....                                   | 34 |
| Display and Label Coordinates in "menurlocus" .....                | 36 |
| Zoom in and out of plots from "menurlocus" .....                   | 39 |
| <i>Time Response Functions</i> .....                               | 40 |
| menutimes: for creating time response plots.....                   | 40 |
| Add a plot in "menutime".....                                      | 41 |
| Zoom in and out of plots from "menutime" .....                     | 46 |
| <i>Analysis Functions</i> .....                                    | 47 |
| routh: for calculating stable gain ranges .....                    | 47 |
| ezmargin: for calculating gain and phase margins.....              | 49 |
| survey: for an overall analysis.....                               | 52 |

|                                                        |           |
|--------------------------------------------------------|-----------|
| <i>systemroot: for Solving a Solvent Matrix</i> .....  | 54        |
| <b><i>Exchang Transfer Functions with CC</i></b> ..... | <b>56</b> |
| <b>getcc: to put a CC function into Matlab</b> .....   | <b>56</b> |
| Set up getcc to work for the first time .....          | 56        |
| Use getcc .....                                        | 57        |
| <b>putcc: to put a Matlab function into CC</b> .....   | <b>58</b> |
| Set up putcc to work for the first time .....          | 58        |
| Use putcc.....                                         | 59        |
| <b><i>Miscellaneous Functions</i></b> .....            | <b>60</b> |
| <b>dispmat: for displaying matrices</b> .....          | <b>60</b> |
| <b><i>Resources</i></b> .....                          | <b>61</b> |

## How To Use This Manual

This report gives instructions on how to use the Aerocontrols Matlab Toolbox. In order to simplify things, each section starts with a description of the command, and the proper syntax to use it.

For instance, here is an example for the `tf` function:

syntax: **tf(g)** where `g` is a transfer function.

Basically, "tf" is the command. Everything in the paranthesis (`g` in this example) can be any variable name. So, if you have a transfer function called "example," the command would be "tf(example)."

This manual is also color-coded to clarify things.

Plain black text is the main portion of instructions and descriptions. It is the default text.

**Blue, monospaced text** is used to show text, or results, given by Matlab in the command window. You will usually see menu-text and selections provided by the Matlab programs shown in this color.

**Red, bold, monospaced text** is used to show where you type in commands into the Matlab Command Window.

At the end of each section, this manual shows any commands that are related to the command just described.

As a final note, every command in the Aerocontrols Toolbox has built-in help text. Just type help followed by the command. For instance, to get help on the `tf` command, simply type **help tf**.

If there are any questions that aren't answered in this manual, contact James Sorenson at [james\\_sorenson@mac.com](mailto:james_sorenson@mac.com).

## Installation

Download the file `aerocontrolsv1.exe` (for the PC) or `aerocontrolsv1.hqx` (for the Macintosh). Open the downloaded file by double-clicking it. The file will self-extract into a folder called `aercontrolsv1`. Drag this folder into the Matlab "Toolbox" directory. Start up the Matlab application. Select "Set Path" under the "File" menu and add the "aerocontrolsv1" directory to the end of the Matlab path. Save the path settings and close the window. That completes the installation.

To verify that the toolbox has been installed correctly, type:

`help aerocontrols`

You should be presented with a list of commands provided by the `aerocontrols` toolbox. If Matlab can't find the file, verify that the path settings are correct.

From here on, anytime you wish to use this toolbox, type: `ssetup` to setup Matlab to use the toolbox.

## Create a Transfer Function

There are multiple ways of creating a function using this toolbox. Transfer functions can be created using the symbolic method with `ssetup`, or by using the `maketf` or `shf` function.

### ***ssetup: to create transfer functions***

syntax: `ssetup`

To create transfer functions using symbolic notation usually requires the symbolic toolbox. The command, `ssetup`, gets around this. It creates a series of variables of type "transfer function" called `s2`, `s3`, `s4`, `s5`, etc. These represent  $s^2$ ,  $s^3$ ,  $s^4$ , and  $s^5$  respectively. For instance, to create the following function:

$$\frac{100 (s+5) (s+2)}{(s+3) (s^3+5s^2+s+10)}$$

Type `ssetup` (needed only once per session) and then type:

$$g=(100*(s+5)*(s+2))/((s+3)*(s^3+5*s^2+s+10))$$

The variable `g` now contains the transfer function mentioned above.

Related: `maketf`, `shf`

## ***shf: to create shorthand transfer functions***

syntax: `g=shf('shorthandfunction')`

This creates a transfer functions by using the shorthand notation. This is where the function starts with a system gain, then lists the poles and zeros with (#) while the damping and and natural frequencies are stored in  $[\zeta \omega_n]$ . For example, '(s-3)' would be type in as (-3), while ' $\zeta=.7$  and  $\omega_n=3$ ' would be type in as [.7 3] or [.7,3].

When creating a transfer function, the first value must be a gain, even if it's a 1! Do NOT use any nested loops. This function ignores spaces between groups, so you can insert them for readability if you wish. Just list out the zeros and stability groups in the numerator; put in a '/', then list out the denominator. See the example below:

```
g=shf('25(4) [.7 3]/[.5 12] (6)')
```

This would create a shorthand function in `g` as follows:

$$\frac{25(4) [.7 3]}{[.5 12] (6)}$$

To confirm this, type `sho(g)` to display the function using shorthand notation.

Related: `ssetup`, `maketf`, `shf`

## ***maketf: to create transfer functions***

syntax: **g=maketf** where g is assigned a transfer function.

This function creates transfer functions using one of 3 methods:

1. Zeros, poles, and gain
2. Polynomials
3. Shorthand notation

Let's say you wish to create the following **zero/pole** transfer function into the variable **g**:

$$\frac{100 (s+5) (s-2)}{s^2 (s+3)}$$

Type **g=maketf** . Then type **1** to select the zeros and poles method. Now fill in the following prompts with the gain, zeros, and poles of the function:

What is the gain?: **100**

Type in the zeros within brackets [# # #]: **[-5 2]**

Type in the poles within brackets [# # #]: **[0 0 -3]**

**g** will now contain the following function:

$$\frac{100 (s+5) (s-2)}{s^2 (s+3)}$$

Next, let's say you wish to create the following **polynomial** transfer function into the variable **g**:

$$\frac{100 (s+5) (s^2+2)}{(s+3) (s^3+5s^2+s+10)}$$

Type **g=makezfs** . Then type **2** to select the polynomial method. Now fill in the following prompts with the gain, numerator, and denominator of the function:

What is the gain?: **100**

Enter a numerator polynomial group within brackets [# # #]: **[1 5]**

Enter a numerator polynomial group within brackets [# # #]: **[1 0 2]**

Enter a numerator polynomial group within brackets [# # #]: **<return>**

Enter a denominator polynomial group within brackets [# # #]: **[1 3]**

Enter a denominator polynomial group within brackets [# # #]: **[1 5 1 10]**

Enter a denominator polynomial group within brackets [# # #]: **<return>**

**g** will now contain the following function:

$$\frac{100 (s+5) (s^2+2)}{(s+3) (s^3+5s^2+s+10)}$$

Finally, there is the **shorthand** notation. Suppose you wish to create the following function into the variable **g**:

```
100 (5)
-----
(1) [0.83205, 3.6056] (8)
```

Notice that the brackets are shorthand notation for a [damp, freq] group. Type **g=make\_tf**. Then type **3** to select the shorthand method. Now fill in the following prompts with the gain, numerator, and denominator groups of the function:

```
What is the gain?: 100
Enter a numerator group [# #] or #: 5
Enter a numerator group [# #] or #: <enter>
Enter a denominator group [# #] or #: 1
Enter a denominator group [# #] or #: [0.83205 3.6056]
Enter a denominator group [# #] or #: 8
Enter a denominator group [# #] or #: <enter>
```

**g** will now contain the following function:

```
100 (5)
-----
(1) [0.83205, 3.6056] (8)
```

Related: **ssetup**, **sho**, **shf**

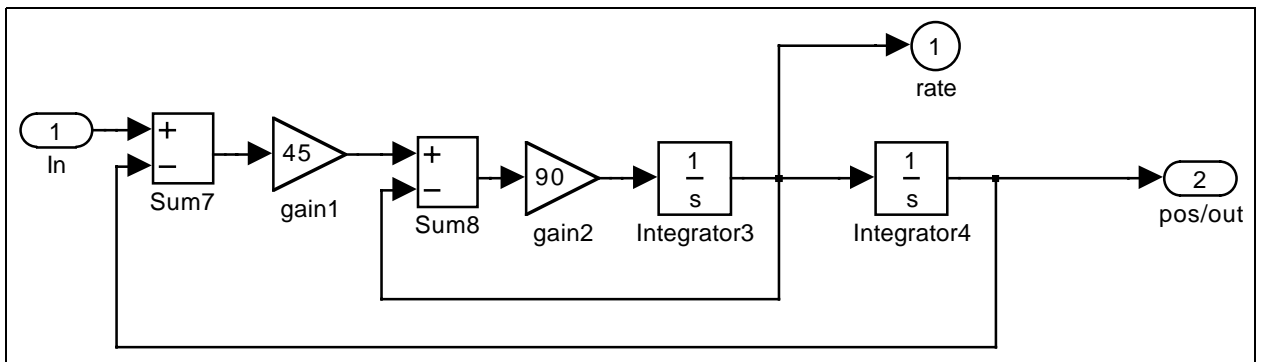
## ***tfmodel: to create transfer functions from Simulink diagrams***

syntax: `g=tfmodel('model')` where:

'model' is a string for the name to the Simulink diagram (without the extension).

`g` is a matrix of transfer functions arranged as: `g(output, input)`.

This creates an array of transfer functions from a Simulink diagram. This permits you to do a frequency response (bode, root locus) on a control system while easily editing the system. Take the Simulink diagram of the actuator as shown in figure 1.



**Figure 1: Simulink diagram called "actuator.mdl" with one input(position) and 2 outputs (rate, position)**

To create transfer functions from this diagram, type the following:

```
>>g=tfmodel('actuator')
```

This creates the following in `g`:

Transfer function from input to output...

$$\text{\#1: } \frac{4050 \text{ s}}{s^2 + 90 \text{ s} + 4050}$$

$$\text{\#2: } \frac{4050}{s^2 + 90 \text{ s} + 4050}$$

The variable, `g`, is a matrix of transfer functions arranged as (output, input). So, `g(1,1)` gives the transfer function from input 1 to output 1 (rate), while `g(2,1)` gives the transfer function from input 1 to output 2 (position). In order to make changes: edit the Simulink Diagram, save your changes, then run the `tfmodel` command again.

Related: `ssetup`, `sho`, `shf`

## ***ezpade: a time-delay approximation***

syntax: **g=ezpade(T,N)** where **g** is the resulting **N**th-order approximate transfer function for  $e^{(-T*s)}$ .

This creates a time-delay transfer function of  $e^{(-T*s)}$  to multiply to an existing transfer function. Matlab does not allow 's' to be in exponents, so an approximation is needed. Matlab has this function built in, but it does not give the results as a transfer function. With **ezpade**, it will give you a function you can multiply to an existing function, or you can use **ezpade** within a transfer function.

For example, let's make the following function, using a 3<sup>rd</sup>-order approximation:

$$\frac{10s}{s^2 + 3s + 7} * e^{-4s}$$

One way is to create two separate functions, then multiply together:

```
>>g1=(10*s)/(s2+3*s+7)
```

```
Transfer function:
```

```
10 s  
-----  
s^2 + 3 s + 7
```

```
>>g2=ezpade(4,3)
```

```
Transfer function:
```

```
-s^3 + 3 s^2 - 3.75 s + 1.875  
-----  
s^3 + 3 s^2 + 3.75 s + 1.875
```

```
>>g=g1*g2
```

```
Transfer function:
```

```
-10 s^4 + 30 s^3 - 37.5 s^2 + 18.75 s  
-----  
s^5 + 6 s^4 + 19.75 s^3 + 34.12 s^2 + 31.88 s + 13.12
```

The other way is to include the **ezpade** within the function:

```
>>g = ((10*s) / (s^2+3*s+7)) *ezpade(4,3)
```

Transfer function:

$$\frac{-10 s^4 + 30 s^3 - 37.5 s^2 + 18.75 s}{s^5 + 6 s^4 + 19.75 s^3 + 34.12 s^2 + 31.88 s + 13.12}$$

Either way, you now have an approximate time-delayed function for analysis.

Related: [ssetup](#)

## Display a Transfer Function

A transfer function can be displayed using multiple formats: polynomial, zero-pole, time-constant, and shorthand. This section describes the command for each notation.

### ***tf: for polynomial notation.***

syntax: `tf(g)` where `g` is a transfer function.

This is the default display for matlab and is rarely needed. Simply typing `g` (or whatever is the name of the transfer function) is usually sufficient to get a polynomial display.

```
>>tf(g)
```

```
Transfer function:
      100 s + 500
-----
s^4 + 15 s^3 + 75 s^2 + 165 s + 104
```

Related: `sho`, `tc`, `zpk`

### ***zpk: for the Zero-Pole-Gain notation***

syntax: `zpk(g)` where `g` is a transfer function

This Matlab built-in function displays the transfer function factored down. This makes the zeros, poles, and gain easily discernable.

```
>>zpk(g)
```

```
Zero/pole/gain:
      100 (s+0.05)
-----
(s+8) (s+1) (s^2 + 6s + 13)
```

Related: `sho`, `tc`, `tf`

### ***sho: for the Shorthand notation***

syntax: **sho** (g) where g is a transfer function

This function displays a transfer function using the shorthand method. System gain is listed first. The negative of the poles and zeros are shown in paranthesis (#), while the dampening  $\zeta$  and natural frequency  $\omega_n$  are shown in brackets [ $\zeta$ ,  $\omega_n$ ]. This format is most helpful for estimating the stability and root locus of a transfer function.

**>>sho** (g)

```
100 (0.05)
-----
(1) [0.83205, 3.6056] (8)
```

Related: **tc**, **tf**, **zpk**

### ***tc: for the Time-Constant notation***

syntax: **tc** (g) where g is a transfer function

This function displays a transfer function using the time-constant method. Type gain is listed first. The rest of the transfer function is similar to the pole-zero format, except that each factor is formatted to show the time-constant at the front of each factor-group. Time-constant formatting is used to estimate the response of a transfer function.

**>>tc** (g)

```
0.048077 (20s + 1)
-----
(1s + 1) (0.076923s^2 + 0.46154s + 1) (0.125s + 1)
```

Related: **tc**, **tf**, **zpk**

# Bode Plot Functions

## *ezbode: for creating bode plots*

syntax: `ezbode(g, range, 'style')`

where:

`g` is the transfer function you wish to plot (required).

`range` is the frequency range to plot in a cell array (optional). Ex: `ezbode(g, {1e-1, 1e5})`

`'style'` is the style of the line you wish to draw (in single quotes) (optional).

Styles are as follows:

|   |         |   |                  |    |         |
|---|---------|---|------------------|----|---------|
| y | yellow  | . | point            | -  | solid   |
| m | magenta | o | circle           | :  | dotted  |
| c | cyan    | x | x-mark           | -. | dashdot |
| r | red     | + | plus             | -- | dashed  |
| g | green   | * | star             |    |         |
| b | blue    | s | square           |    |         |
| w | white   | d | diamond          |    |         |
| k | black   | v | triangle (down)  |    |         |
|   |         | ^ | triangle (up)    |    |         |
|   |         | < | triangle (left)  |    |         |
|   |         | > | triangle (right) |    |         |
|   |         | p | pentagram        |    |         |
|   |         | h | hexagram         |    |         |

This command displays a bode plot for a transfer function. The main advantage of this function over the built-in, Matlab function: `bode`, is that it formats the plot using dB for magnitude, shows the 0 magnitude and 180 phase line, and wraps the phase line to stay within 0 to -360 degrees. It also allows you to use `menubode` to manipulate it. If you had the following transfer function for `g`:

```
g=
Transfer function:
          1000 s + 50
-----
s^4 + 15 s^3 + 75 s^2 + 165 s + 104
```

The command:

```
»ezbode(g, {1e-2, 1e2}, 'rs--')
```

will create a bode plot for  $\omega=0.01\text{rad/sec}$  to  $\omega=100\text{rad/sec}$  using a red dashed line with squares as shown in figure 2. Using the command again will add additional plots to the window. Use the `figure` command to start a separate plot window.

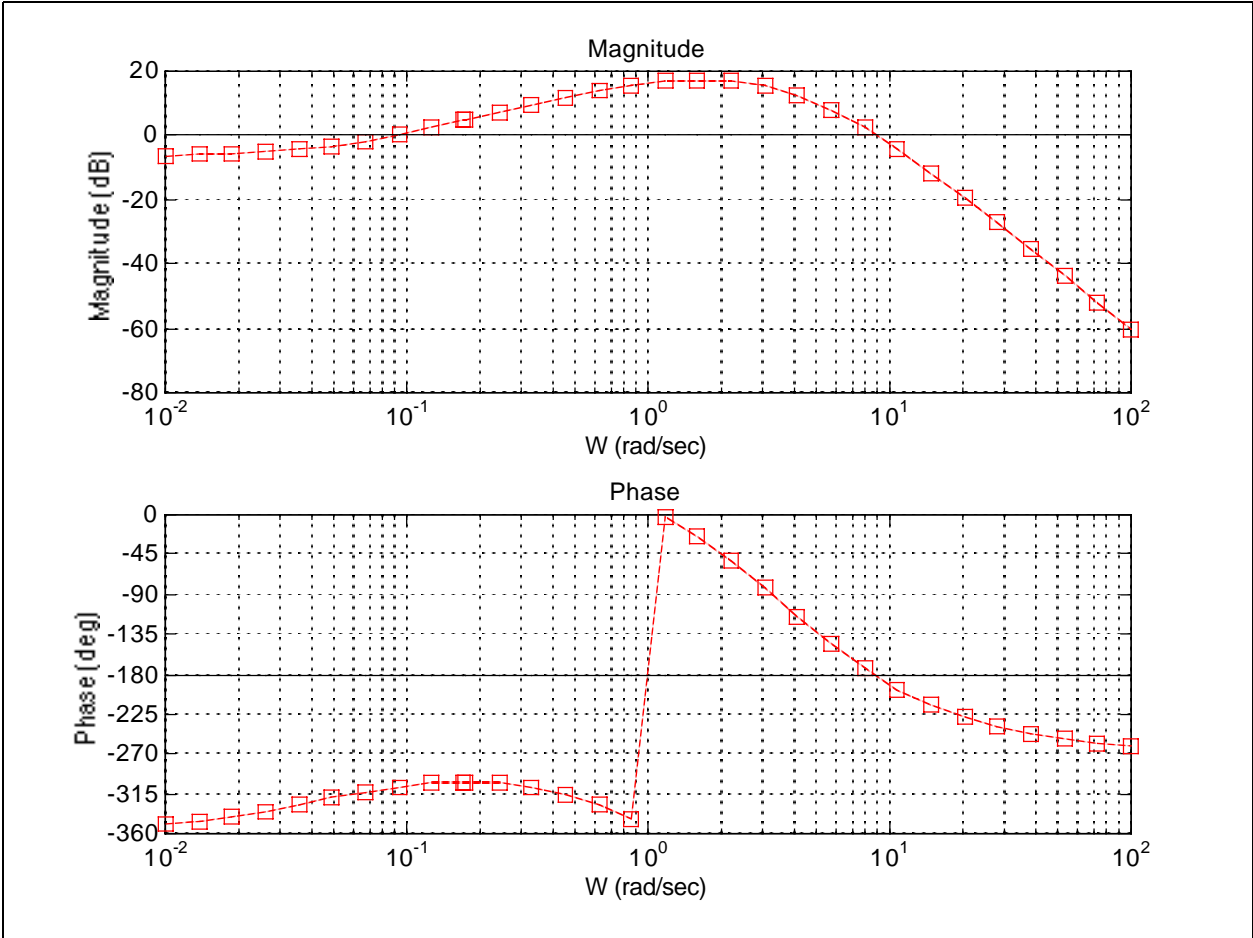


Figure 2: Bode plot of g

Related: `bodeslope`, `menubode`, `menurlocus`, `menutime`

## ***menubode: for creating/manipulating bode plots***

syntax: **menubode**

This script provides a menu interface for creating and manipulating bode plots. It provides an interactive menu for plotting different style/color plots, calculating margins, zooming, and displaying coordinates. It also has the ability to plot the asymptotic approximation of a transfer function as well.

**»menubode**

Select from the following

- A) Add a bode plot to the chart
- C) Display coordinates of mouse click
- M) List and plot the gain and phase Margins
- Z) Zoom in on the chart
- G) Toggle the grid on/off
- S) Add the asymptote bode approximation to the plot
- N) New Figure
- Q) Quit this menu

Type in your choice:

Related: **bodeslope, ezbode, menurlocus, menutime**

## Add a plot in "menubode"

To add a bode plot, type in **menubode** and select "A) Add a bode plot to the chart."

Type in your choice: **A**

Then type in the name of the function ("g" in this case).

Type in the function name to add: **g**

It will then display the function and provide a menu for plotting color. In this example, red is chosen.

Transfer function:

$$\frac{1000 s + 50}{s^4 + 15 s^3 + 75 s^2 + 165 s + 104}$$

(y)ellow, (m)agenta, (c)yan, (r)ed, (g)reen, (b)lue, (w)hite, blac(k)  
What color? (single letter, Enter for (b)lue): **r**

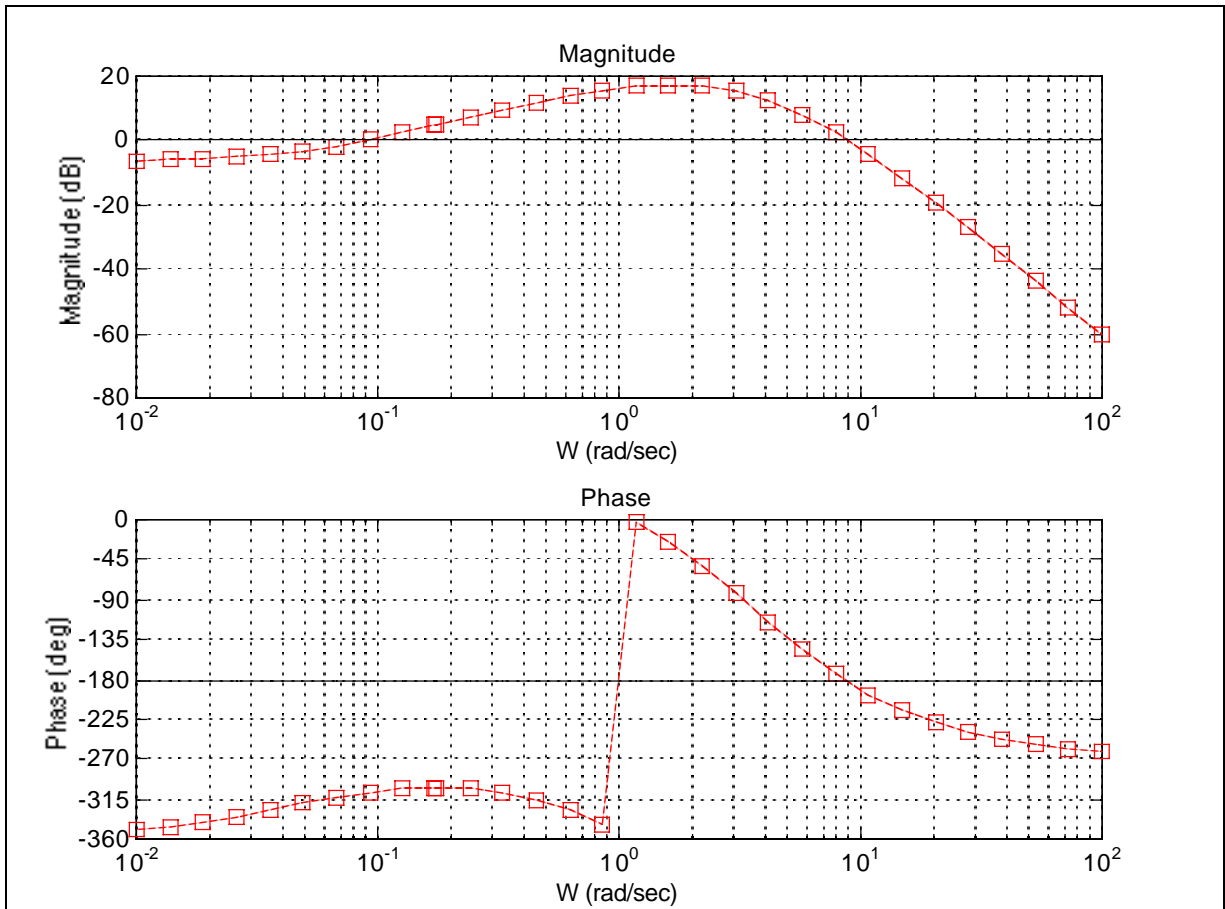
Next, a menu of symbols is provided. These symbols will be displayed along the line. In this example, squares are chosen.

(.) (o) (x) (+) (\*) (s)quare (d)iamond (p)entagram (h)exagram  
Triangles in respective directions: (v) (^) (<) (>)  
Select a shape for marks along the line (Enter for none): **s**

Next, a menu of line-styles is displayed. In this example, we choose a dashed line for plotting.

(-)solid, (:)dotted, (-.)dashdot, (--)dashed  
Style of the line? Use the characters in (), Enter for (-)solid: **--**

Figure 3 shows the resulting bode plot of transfer function **g**. Magnitude is shown on top in decibels. Phase is below with the frequency axis in sync with both plots. Notice that the line is red, dashed, and is labeled with squares. For convenience, the 0dB magnitude line and 180deg phase line are clearly drawn. Selecting **Edit=>Copy** will copy the plot to the clipboard for pasting into another application. The command window brings you back to the menu where you can add additional plots to the same window for comparison.



**Figure 3: Resulting bode plot**

Related: [ezbode](#), [menubode](#), [menurlocus](#), [menutime](#)

## Display and Label Coordinates in "menubode"

Sometimes you need to find the coordinates of a specific point on a bode plot. Under **menubode**, the menu item, "Display coordinates of mouse click," will show the magnitude, phase, frequency, and magnitude-slope of any point clicked on the chart. Let's say we have a plot of the following transfer function **g**:

$$\frac{1000 s + 50}{s^4 + 15 s^3 + 75 s^2 + 165 s + 104}$$

Within the **menubode** menu, select "(C) Display coordinates of mouse click."

Type in your choice:**c**

You will then be prompted for the transfer function to get coordinate information from. In this example, the transfer function is called **g**.

Type in the function name to get coordinates from:**g**

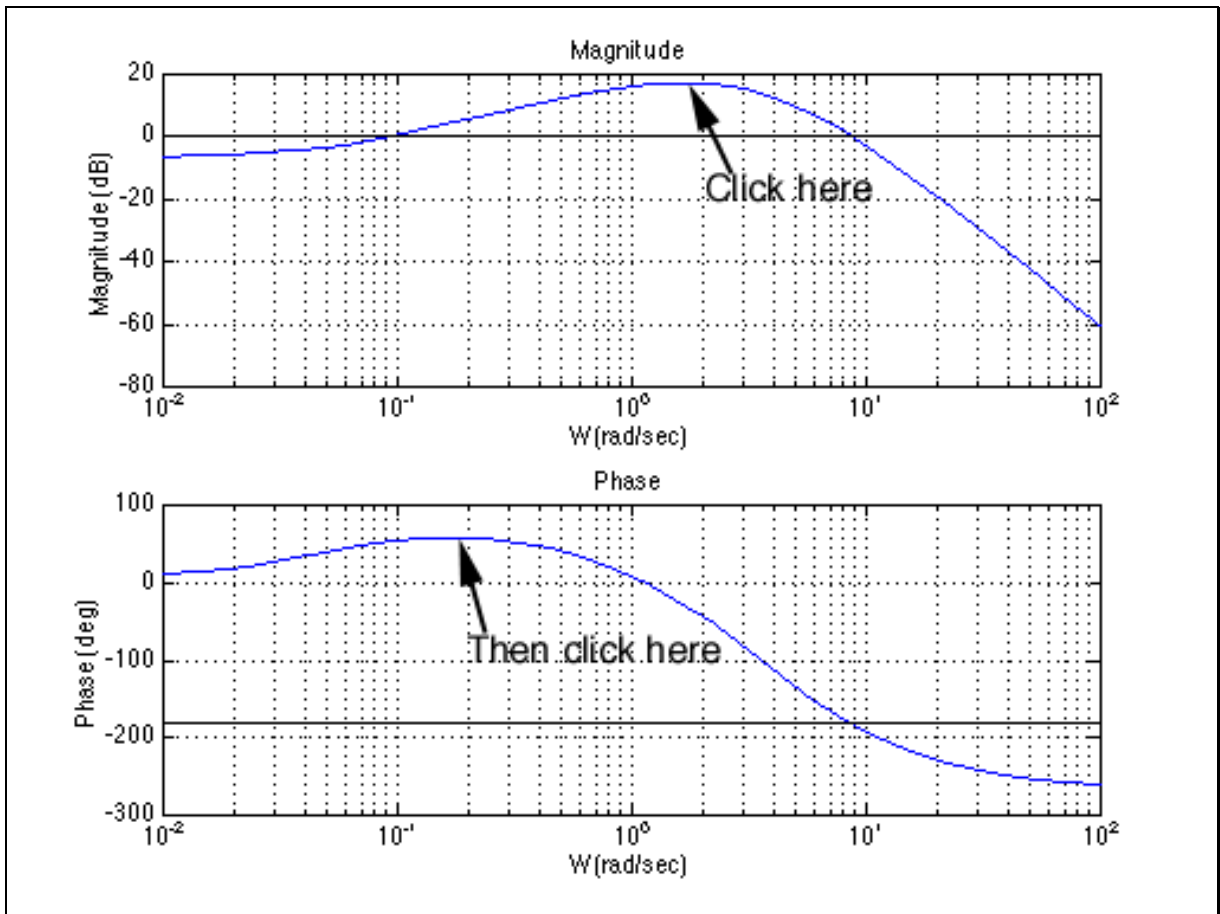
It will then display the transfer function, give the following instructions, and switch you to the plot.

Transfer function:

$$\frac{1000 s + 50}{s^4 + 15 s^3 + 75 s^2 + 165 s + 104}$$

Left-Click on a point to display coordinates in the command window.  
Right-Click to label the point on the plot itself.  
Press ENTER while on the chart to Quit.

To elaborate: let's see you want information for the maximum magnitude and phase angle. Left-click on the two peaks and information will be shown in the command window. See Figure 4.



**Figure 4: Clicking for information**

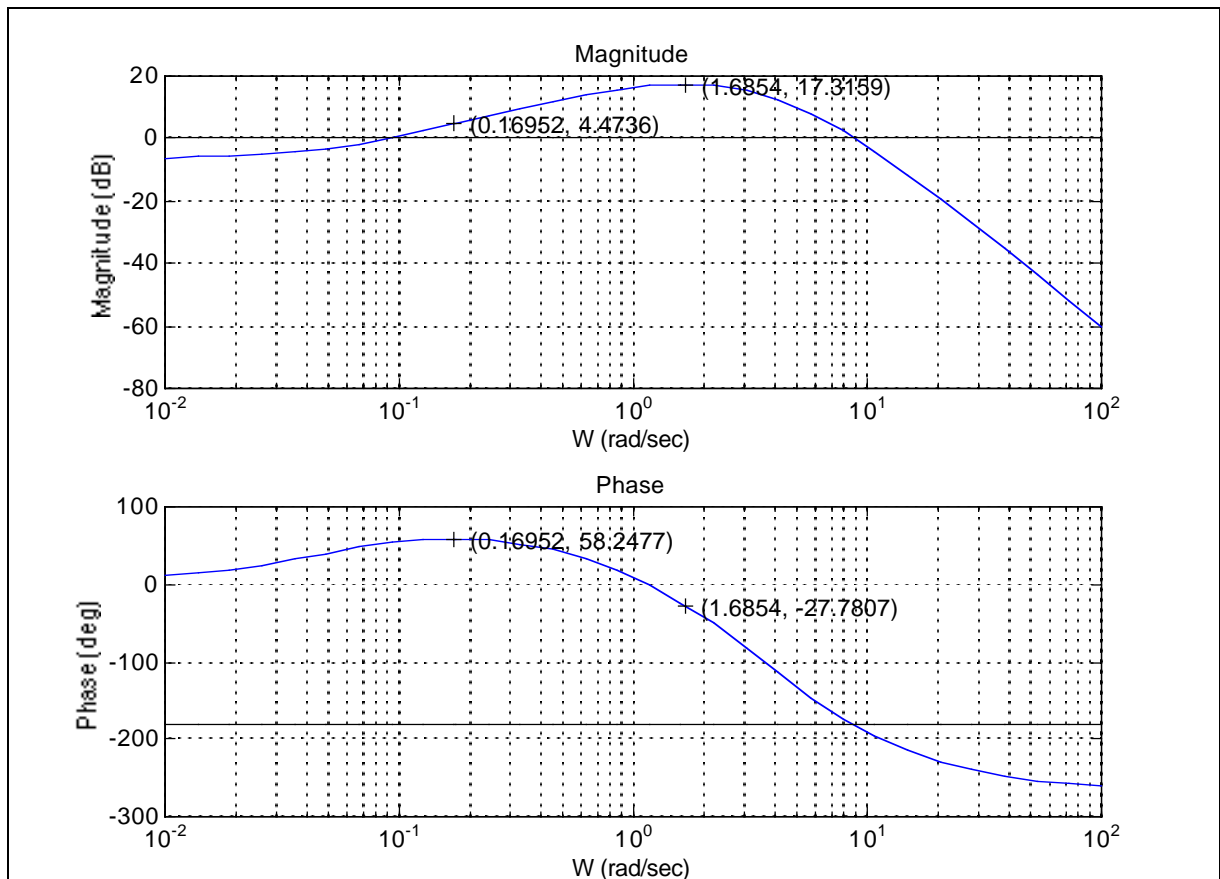
Shown in the command window:

W=1.5721Hz, Mag=17.296dB, Phase=-22.3356deg  
 slope= 0.30314 mag/Hz, 1.3012 dB/decade

W=0.16952Hz, Mag=4.4736dB, Phase=58.2477deg  
 slope= 8.786 mag/Hz, 17.7975 dB/decade

Notice that the slope is always for the magnitude line. You can continue to click and list coordinates until you press ENTER while having the plot window active. This will return you to the menu in the command window.

Now, to **label** the coordinates of the plot, use the right mouse button instead. The best way to go about this is to position the plot so that the command window is visible. Repeatedly **left**-click on the plot until you have the mouse pointer positioned over the coordinates you wish to label, then click the **right** mouse button. The coordinates will be labeled directly on the plot. See Figure 5.



**Figure 5: Labeling the bode plot**

Notice that when you click on a point, the point at that frequency is labeled on both the magnitude and phase plot window. A "+" labels the exact point that was clicked.

Related: [ezbode](#), [menubode](#), [menurlocus](#), [menutime](#)

## Zoom in and out of plots from "menubode"

In order to more closely examine certain regions of the curve, menubode offers a zoom option. At the menu, select "Z) Zoom in on the chart."

Type in your choice: **z**

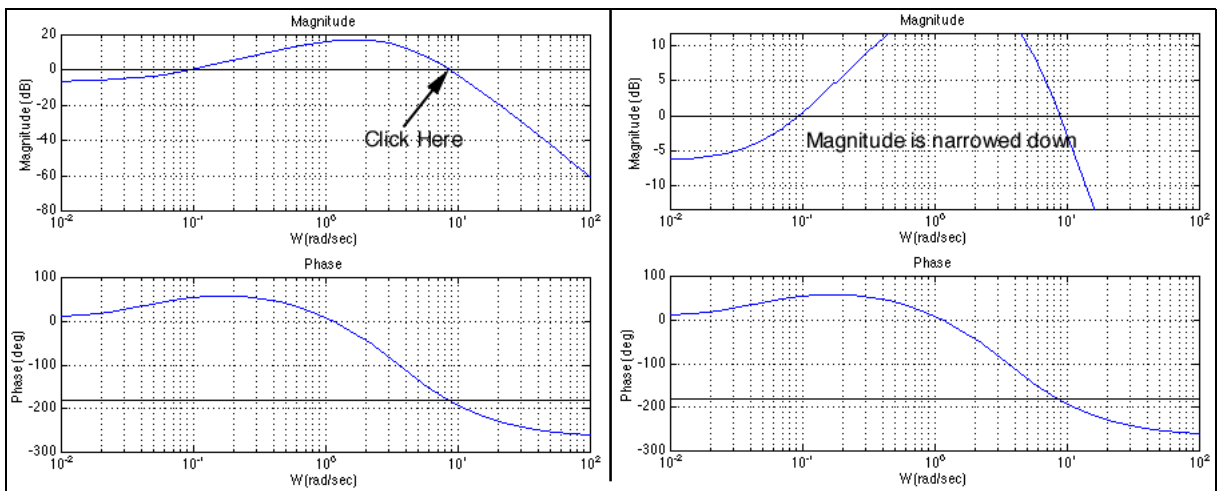
Zoom (X)-axis, (Y)-axis, or (B)oth?: **y**

Begin clicking on the areas to zoom in on...

Right-click zooms out

Zoom will be active until another choice is selected

Notice that it will immediately return you to the menu list. Don't worry; the zoom function will remain active until you select a different menu item. To zoom in, simply click on the region to zoom into. Since **y** was selected, only the magnitude or phase gets zoomed; the frequency axis remains constant. Selecting **x** or **b** will enable you to zoom the frequency axis as well. To ensure that the magnitude and phase diagrams do not go out of sync with each other, the frequency(X) axis will always be zoomed identically no matter which window you click in. See Figure 6.



**Figure 6: Zooming in the bode plot**

Related: [ezbode](#), [menubode](#), [menurlocus](#), [menutime](#)

## Gain and Phase Margins within "menubode"

Sometimes the gain and phase margins of a transfer function are needed to evaluate the stability and possible gains of a system. **Menubode** has the ability to calculate and plot the margins of an existing, plotted transfer function. Withing the main menu of **menubode**, select "List and plot the gain and phase Margins."

Type in your choice: **m**

It will then prompt you for which function to find the margins for (in case there is more than one function plotted). In this case, the transfer function is called **g**.

Type in the function name to show margins of: **g**

It will then display the transfer function and calculate the margins. The calculations may take some time, so don't worry if the program seems to hang for a bit. When finished, it will display the stable gain ranges, the gain margin(Gm), the frequency at where phase=180 (W180), the phase margin(Pm), the crossover frequency (Wc), and then plot the margins on the plot. See Figure 7.

Transfer function:

1000 s + 100

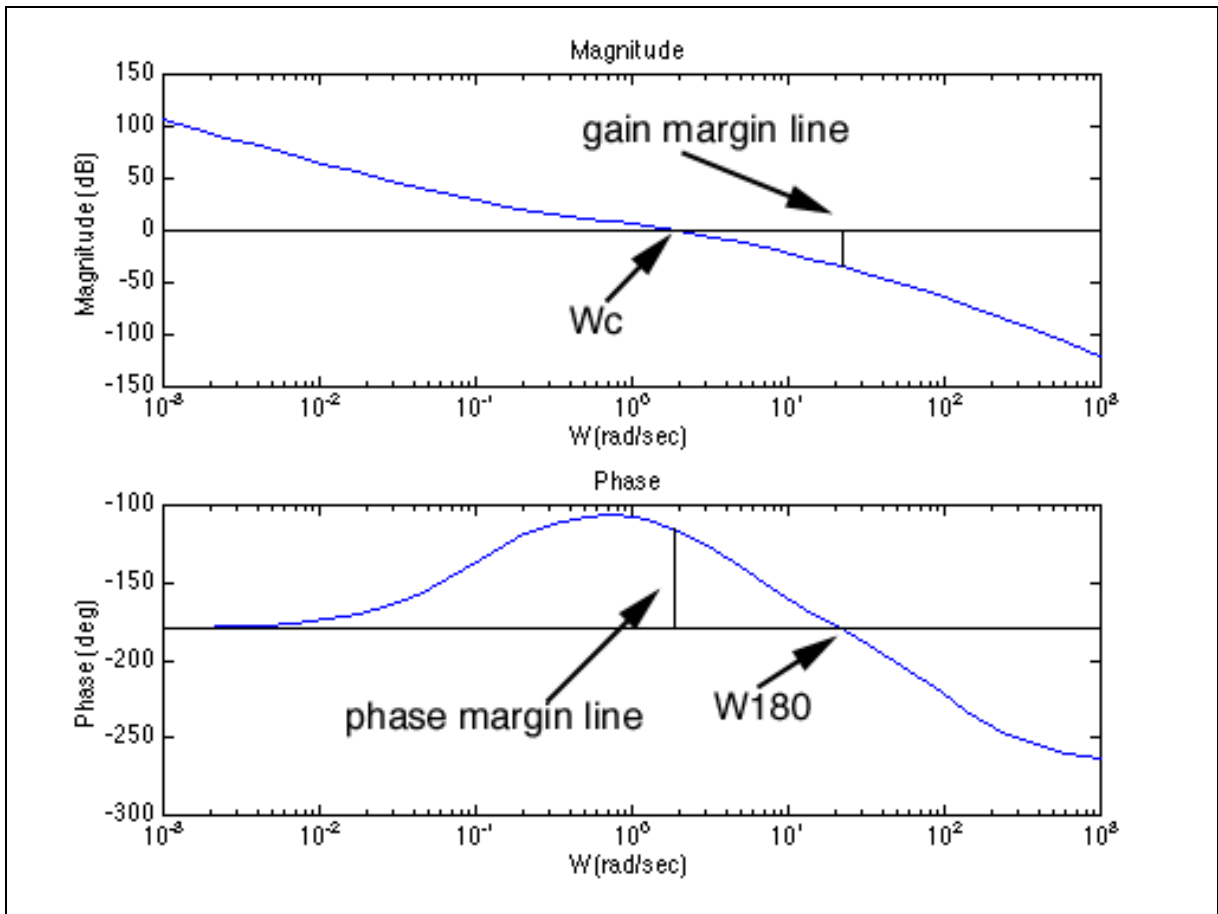
-----  
s<sup>4</sup> + 105 s<sup>3</sup> + 500 s<sup>2</sup>

Stable for gain ranges: -5e-06 to 51.3975

GM=34.2188dB (51.3975), at W180=22.1246 rad/sec (Hz)

Pm=65.317deg, at Wc= 1.875 rad/sec (Hz)

Gm= 34.2188, Pm=65.317, Wc=1.875, W180=22.1246



**Figure 7: The gain and phase margins are plotted on the bode plot**

Notice that the grid lines are removed so that the margin lines are easier to see. The grid can be turned back on from the main menu of `menubode` in the command window.

Related: `ezbode`, `ezmargin`, `menubode`, `menurlocus`, `menutime`

## Asymptote Bode Approximation using "menubode"

An approximation of a bode plot can be made by looking at the zero/pole format of a transfer function. Every zero increases the slope by 20 decibels/decade while poles drop the slope by the same amount. Because of this, the zeros and poles of a function can be estimated by examining where the slope changes on an asymptotic approximation bode plot. As an example, let's say we have the following transfer function:

$$g = \frac{100 (s+0.01)}{s (s+10) (s+1)} = \frac{100 s + 1}{s^3 + 11 s^2 + 10 s}$$

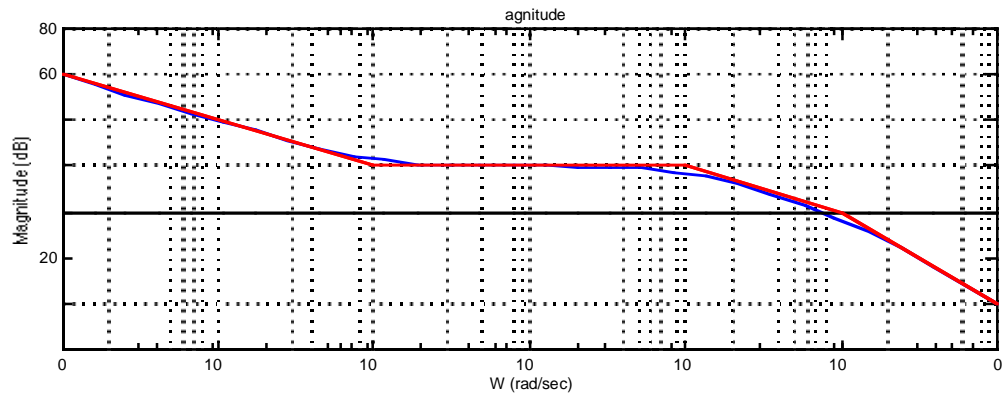
To find the starting magnitude, evaluate the function substituting the starting frequency in for  $s$ . For instance, to start plotting at  $\omega=.0001\text{Hz}$ , substitute in  $s=.0001$ . This gives a magnitude of 1000. Using  $[\text{dB}=20*\log_{10}(\text{magnitude})]$ , we have a magnitude of 60dB at the start of the plot.

From there on, it's using the poles and zeros to adjust the slope. At the start, the solo  $s$  drops the slope to -20 dB/decade. At  $\omega=.01$ , the zero,  $(s+0.01)$ , ups the slope to 0 dB/decade. Then the pole at  $\omega=1$  drops the slope to -20 dB/decade. The pole at  $\omega=10$  drops it further to -40 dB/decade.

To have the aercontrols toolbox automate this process for you, use `menubode` and select "Add the asymptote bode approximation."

Type in your choice: `s`

Like a regular bode plot, it will prompt for the transfer function to plot, and then give options for line style. It will then give the asymptotic approximation of the bode plot. Figure 8 shows the approximation superimposed over the exact bode plot.



**Figure 8: Asymptote approximation in red. Exact bode plot in blue.**

Related: [ezbode](#), [menubode](#)

## ***bodeslope: for Calculating the Slope of a Bode Plot***

syntax: `slope=bodeslope(g,w)` where `g` is a transfer function, `w` is the frequency to evaluate the slope, and `slope` is the magnitude bode slope in units (dB/decade).

In some cases it is necessary to calculate the slope of a bode plot at a given frequency. For example, to test if the slope is -40 dB/decade at the crossover point. This function calculates the magnitude-slope of a transfer function at a specified frequency.

For example, take the following transfer function `g`:

$$\frac{100 s + 1}{s^3 + 11 s^2 + 10 s}$$

To calculate the slope at `w=10Hz` and store that value in `gslope`, the command would be:

```
>>gslope=bodeslope(g,10)
```

which produces the following:

```
slope= -0.10484 mag/Hz, -29.802 dB/decade
```

```
gslope =  
-29.8020
```

Notice that it displays the slope in two forms of units, but stores the dB/decade unit in `gslope`.

Figure 9 clarifies the meaning of the bode slope.

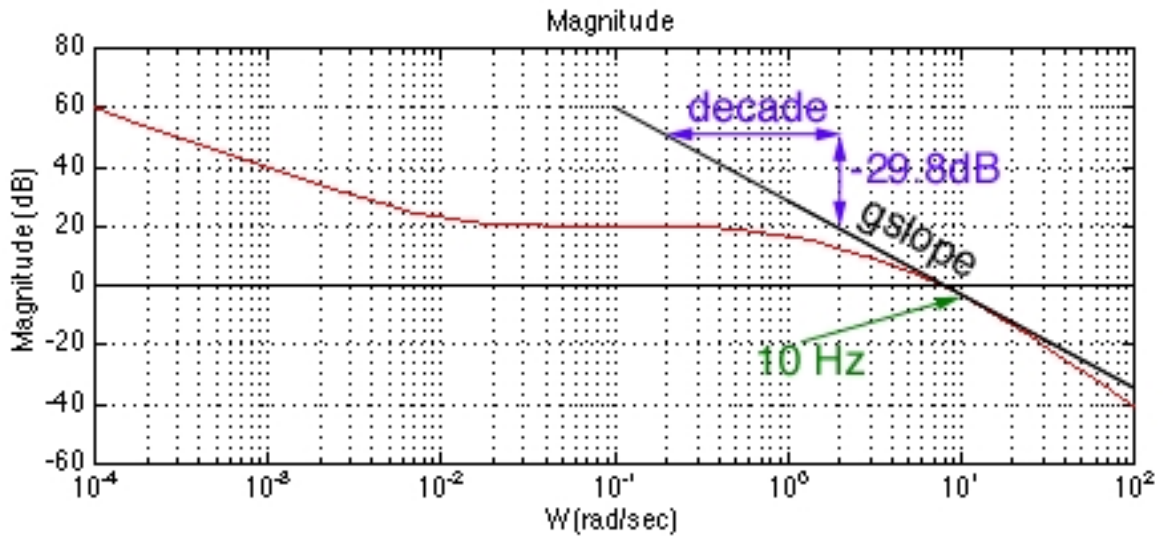


Figure 9: Slope of transfer function "g" is -29.8 dB/decade at  $w=10\text{Hz}$

Related: [ezbode](#), [menubode](#)

# Root Locus Functions

## *ezrlocus: for creating bode plots*

syntax: **ezrlocus** (*g*, *range*, '*style*')

where:

**g** is the transfer function you wish to plot (required).

**range** is the gain range to plot in a cell array (optional). Ex: `ezbode(g, {1e-1, 1e5})`

**'style'** is the style of the line you wish to draw (in single quotes) (optional).

Styles are as follows:

|   |         |   |                  |    |         |
|---|---------|---|------------------|----|---------|
| y | yellow  | . | point            | -  | solid   |
| m | magenta | o | circle           | :  | dotted  |
| c | cyan    | x | x-mark           | -. | dashdot |
| r | red     | + | plus             | -- | dashed  |
| g | green   | * | star             |    |         |
| b | blue    | s | square           |    |         |
| w | white   | d | diamond          |    |         |
| k | black   | v | triangle (down)  |    |         |
|   |         | ^ | triangle (up)    |    |         |
|   |         | < | triangle (left)  |    |         |
|   |         | > | triangle (right) |    |         |
|   |         | p | pentagram        |    |         |
|   |         | h | hexagram         |    |         |

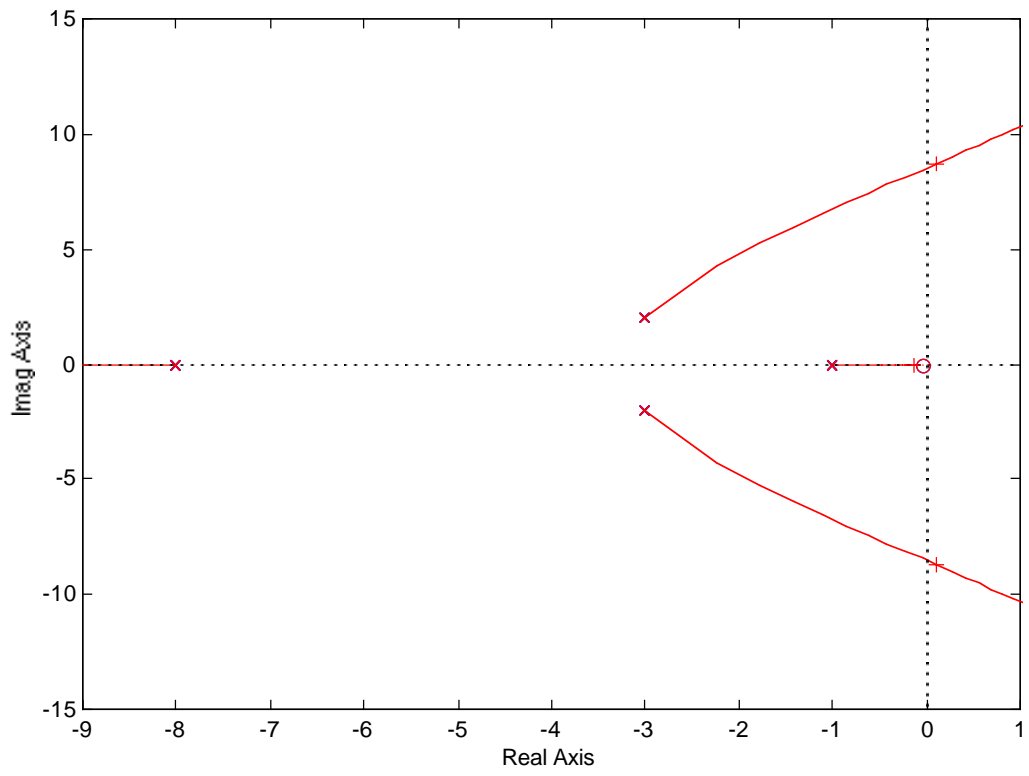
This command displays a root locus plot for a transfer function. The main advantage of this function over the built-in, Matlab function: `rlocus`, is that you can control the color and style of each plot. This makes comparing two plots much easier. If you had the following transfer function for *g*:

```
g=
Transfer function:
          1000 s + 50
-----
s^4 + 15 s^3 + 75 s^2 + 165 s + 104
```

The command:

**»ezrlocus** (*g*, {1,10}, '*r*')

will create a root locus plot for *k*=1 to *k*=10 using a red dashed line with squares as shown in figure 10. Using the command again will add additional plots to the window. Use the **figure** command to start a separate plot window.



**Figure 10: Root locus plot of function g**

Related: [menurlocus](#)

## ***menurlocus: for creating root locus plots***

syntax: **menurlocus**

This script provides a menu interface for creating and manipulating root locus plots. It provides an interactive menu for plotting different style/color plots, zooming, and displaying coordinates.

**»menurlocus**

Select from the following

- A) Add a root locus plot to the chart
- C) Display coordinates of mouse click
- Z) Zoom in on the chart
- G) Toggle the grid on/off
- N) New Figure
- Q) Quit this menu

Type in your choice:

Related: **ezrlocus**, **menubode**, **menutime**

## Add a plot in "menurlocus"

To add a root locus plot, type in **menurlocus** and select "A) Add a root locus plot to the chart."

Type in your choice: **A**

Then type in the name of the function ("g" in this case).

Type in the function name to add: **g**

It will then display the function and provide a menu for plotting color. In this example, red is chosen.

```
Transfer function:
                1000 s + 50
-----
s^4 + 15 s^3 + 75 s^2 + 165 s + 104
```

```
(y)ellow, (m)agenta, (c)yan, (r)ed, (g)reen, (b)lue, (w)hite, blac(k)
What color? (single letter, Enter for (b)lue): r
```

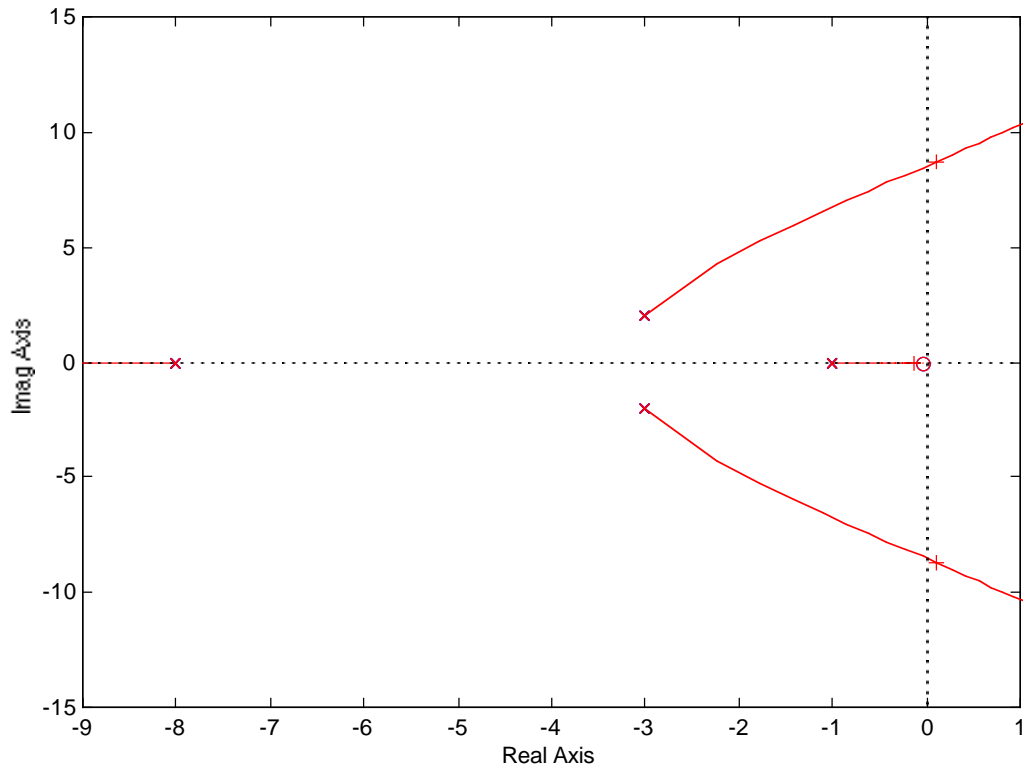
Next, a menu of line-styles is displayed. In this example, let's choose a solid line for plotting.

```
(-)solid, (:)dotted, (-.)dashdot, (--)dashed
Style of the line? Use the characters in (), Enter for (-)solid:-
```

Next, it prompts for a maximum gain range to plot to (starting from 0). Negative values are permissible if you wish to plot the negative of a function. In this example, let's plot to a gain of 10.

```
What maximum gain range do you wish to plot to? (Enter for default): 10
```

Figure 11 shows the resulting root locus plot of transfer function **g**. Notice that the lines are red and solid. Poles are marked with an "x" while zeros are marked with an "o." The location where gain is 1 is labeled with a "+"." Notice that this function is unstable at unity gain. Selecting **Edit=>Copy** will copy the plot to the clipboard for pasting into another application. The command window brings you back to the menu where you can add additional plots to the same window for comparison.



**Figure 11: Root Locus plot with poles(x), zeros(o), and unity gain(+)** labeled.

Related: [ezrlocus](#), [menubode](#), [menurlocus](#), [menutime](#)

## Display and Label Coordinates in "menurlocus"

Sometimes you need to find the coordinates of a specific point on the plot. Under **menurlocus**, the menu item, "Display coordinates of mouse click," will show the coordinates, gain, dampening, and natural frequency of any point clicked on the chart. Let's say we have a plot of the following transfer function **g**:

$$\frac{1000 s + 50}{s^4 + 15 s^3 + 75 s^2 + 165 s + 104}$$

Within the **menurlocus** menu, select "C) Display coordinates of mouse click."

Type in your choice:**c**

You will then be prompted for the transfer function to get coordinate information from. In this example, the transfer function is called **g**.

Type in the function name to get coordinates from:**g**

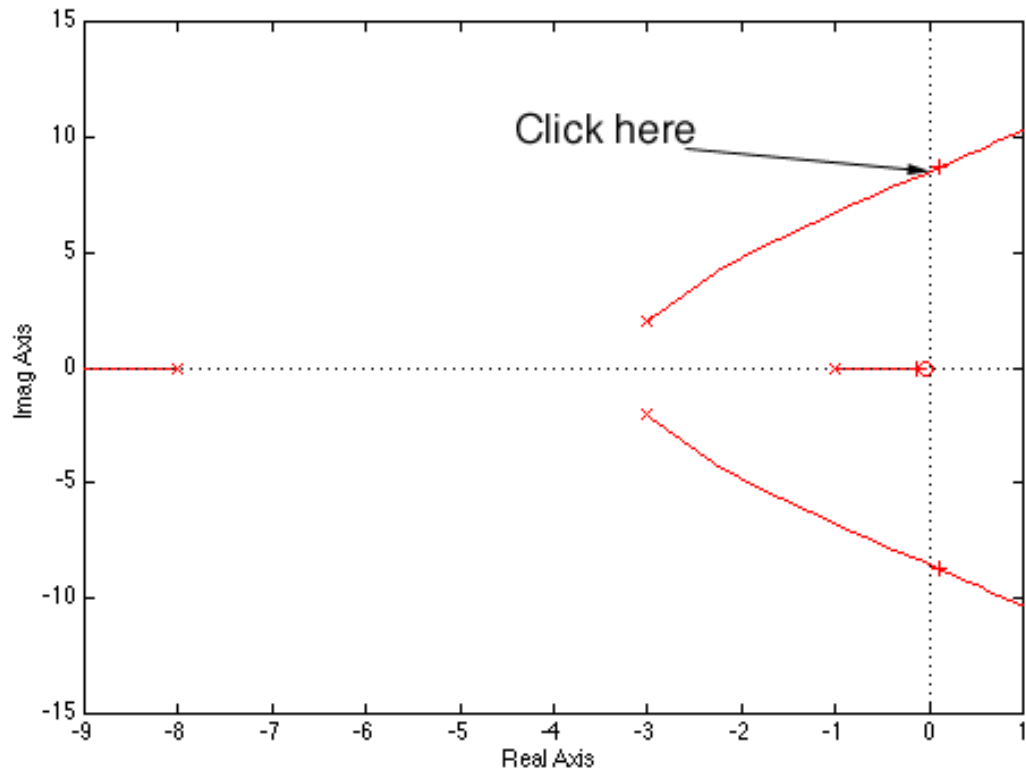
It will then display the transfer function, give the following instructions, and switch you to the plot.

Transfer function:

$$\frac{1000 s + 50}{s^4 + 15 s^3 + 75 s^2 + 165 s + 104}$$

Left-Click on a point to display coordinates in the command window.  
Right-Click to label the point on the plot itself.  
Press ENTER while on the chart to Quit.

To elaborate: let's say you want to know the maximum stable gain you can apply to the function. Left-click where the line intersects the imaginary axis and the information will be shown in the command window. See Figure 12.



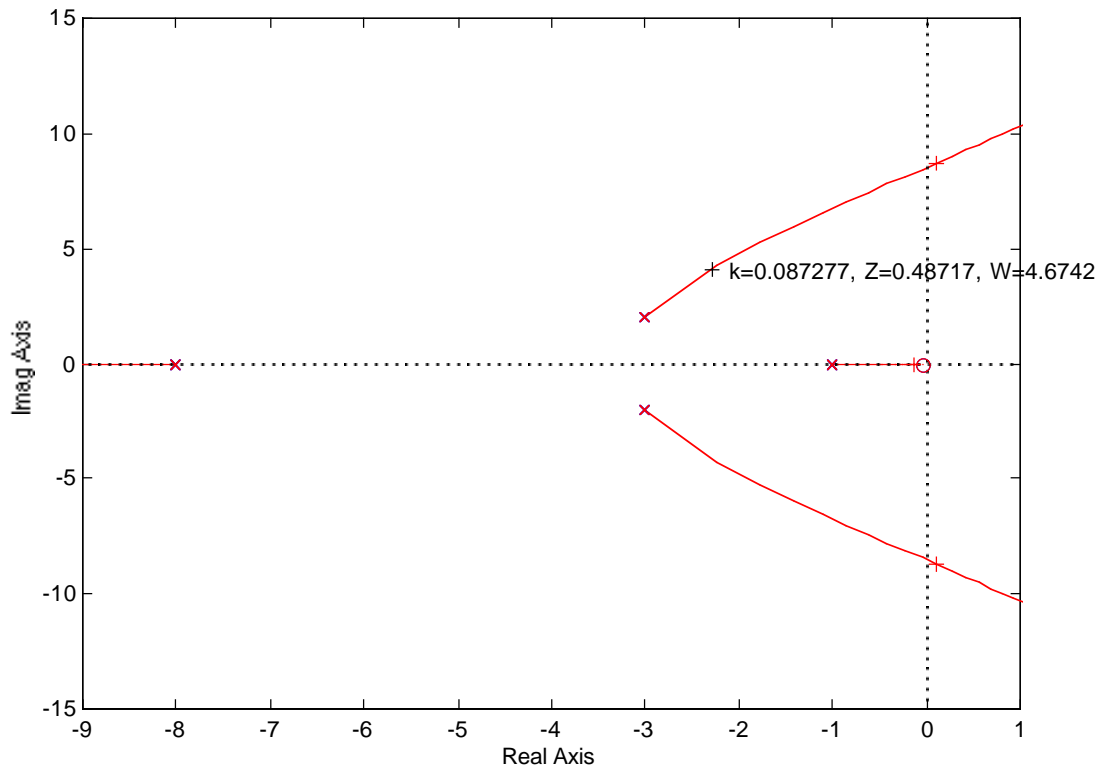
**Figure 12: Clicking for information**

Shown in the command window:

$x+yi = -0.012048+8.5082i$  gain(k)=0.91916 zeta= 0.0014161 Wn= 8.5082

You can continue to click and list coordinates until you press ENTER while having the plot window active. This will return you to the menu in the command window.

Now, to **label** the coordinates of the plot, use the right mouse button instead. The best way to go about this is to position the plot so that the command window is visible. Repeatedly **left**-click on the plot until you have the mouse pointer positioned over the coordinates you wish to label, then click the **right** mouse button. The target will be labeled with a black "+" and the coordinates will be labeled directly on the plot. See Figure 13.



**Figure 13: Labeling the bode plot**

Related: `ezrlocus`, `menubode`, `menurlocus`, `menutime`

## Zoom in and out of plots from "menurlocus"

In order to closely examine certain regions of the curve, menurlocus offers a zoom option. At the menu, select "Z) Zoom in on the chart." Then select which axis you wish to zoom. In this case, the plot will be zoomed on both axis.

Type in your choice: **z**

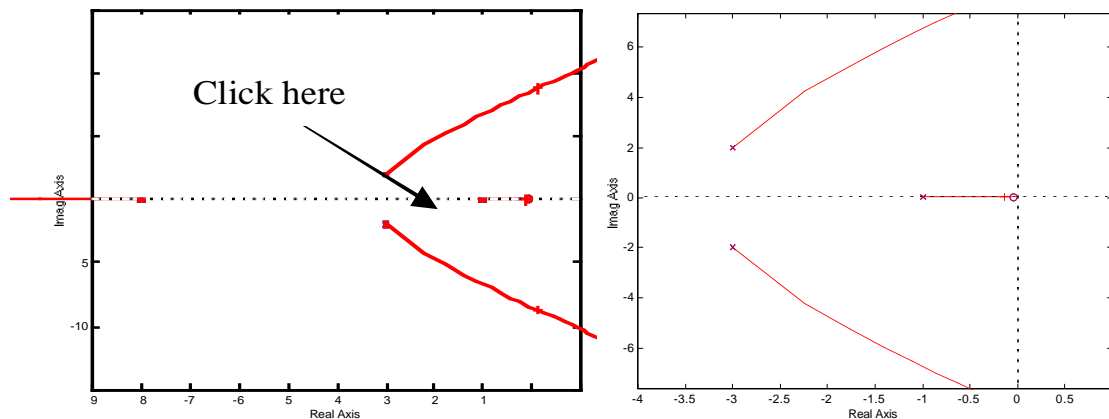
Zoom (X)-axis, (Y)-axis, or (B)oth?: **b**

Begin clicking on the areas to zoom in on...

Right-click zooms out

Zoom will be active until another choice is selected

Notice that it will immediately return you to the menu list. Don't worry; the zoom function will remain active until you select a different menu item. To zoom in, simply click on the region to zoom into. Notice that you can not zoom out farther than what was originally plotted. See Figure 14.



**Figure 14: Zooming in the root locus plot**

Related: [ezrlocus](#), [menubode](#), [menurlocus](#), [menutime](#)

# Time Response Functions

## *menutimes: for creating time response plots*

syntax: **menutime**

This script provides a menu interface for creating and manipulating time response plots. It provides an interactive menu for plotting different style/color plots, zooming, and displaying coordinates.

**>menutime**

Select from the following

- A) Add a root locus plot to the chart
- C) Display/Label coordinates from mouse click
- Z) Zoom in on the chart
- G) Toggle the grid on/off
- N) New Figure
- Q) Quit this menu

Type in your choice:

Related: **menubode**, **menurlocus**

## Add a plot in "menutime"

To add a time response plot, type in `menutime` and select "A) Add a time response to the chart."

Type in your choice: **A**

Then type in the name of the function ("g" in this case).

Type in the function name to add: **g**

It will then display the function and provide a menu for selecting the type of time response plot. In this example, a closed-loop step response is selected.

```
Transfer function:
              1000 s + 50
-----
s^4 + 15 s^3 + 75 s^2 + 165 s + 104

(1) Impulse Open-loop
(2) Impulse Closed-loop
(3) Step Open-loop
(4) Step Closed-loop
Select the type of plot (1,2,3,4) (default=3): 4
```

If you select a closed-loop response, such as in this example, it will prompt for the function to close with. You can simply enter a gain, or provide the name of another transfer function. Here, the function is closed with a gain of 3.

Type in the function to close with (default=1): **3**

Next, it will provide a menu for plotting color. In this example, red is chosen.

```
(y)ellow, (m)agenta, (c)yan, (r)ed, (g)reen, (b)lue, (w)hite, blac(k)
What color? (single letter, Enter for (b)lue): r
```

Next, a menu of symbols are provided. These symbols will be displayed along the line. In this example, squares are chosen.

```
(.) (o) (x) (+) (*) (s)quare (d)iamond (p)entagram (h)exagram
Triangles in respective directions: (v) (^) (<) (>)
Select a shape for marks along the line (Enter for none): s
```

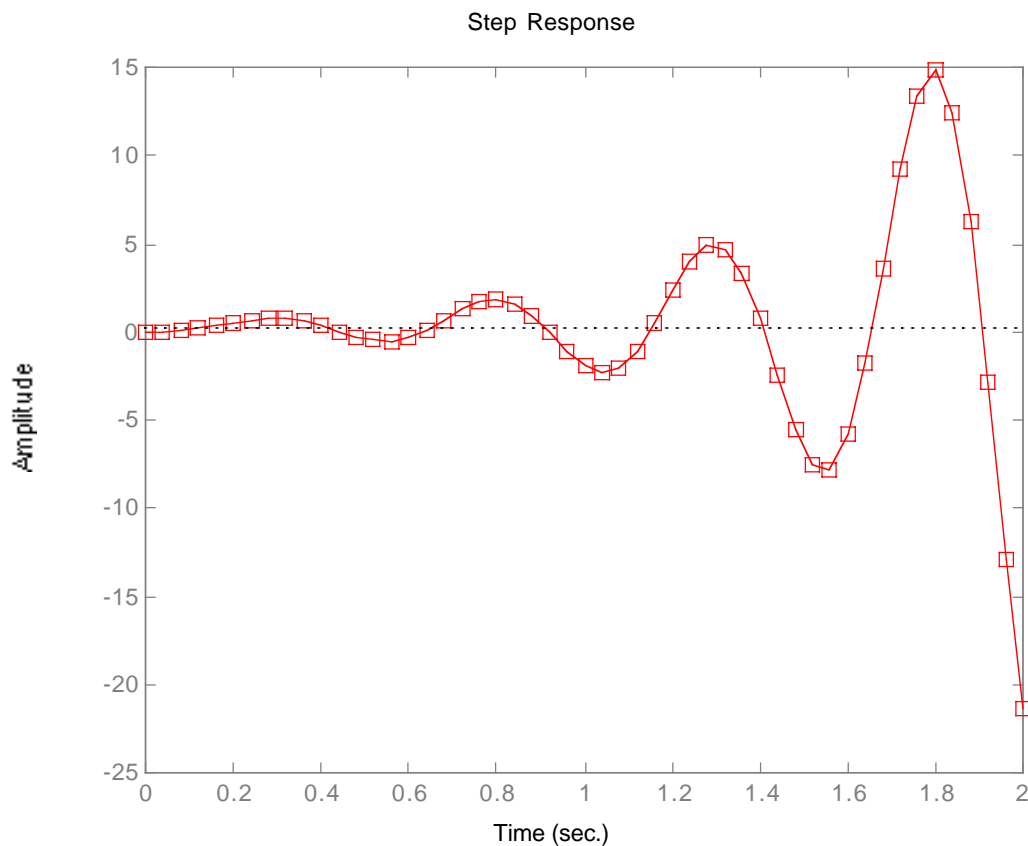
Next, a menu of line-styles is displayed. In this example, let's choose a solid line for plotting.

```
(-)solid, (:)dotted, (-.)dashdot, (--)dashed  
Style of the line? Use the characters in (), Enter for (-)solid:-
```

Next, it prompts for a time range to plot to (starting from 0). In this example, let's plot to 2 seconds.

```
What maximum gain range do you wish to plot to? (Enter for default): 2
```

Figure 15 shows the resulting root locus plot of transfer function **g**. Notice that the line is red, solid, and is labeled with squares. Selecting **Edit=>Copy** will copy the plot to the clipboard for pasting into another application. The command window brings you back to the menu where you can add additional plots to the same window for comparison.



**Figure 15: Closed-loop step response of the transfer function.**

Related: [menubode](#), [menurlocus](#), [menutime](#)

## Display and Label Coordinates in "menutime"

Sometimes you need to find the coordinates of a specific point on the plot. Under **menutime**, the menu item, " Display/Label coordinates from mouse click," will show the time and amplitude of any point clicked on the chart. Let's say we have a plot of the following transfer function **g**:

$$\frac{1000 s + 50}{s^4 + 15 s^3 + 75 s^2 + 165 s + 104}$$

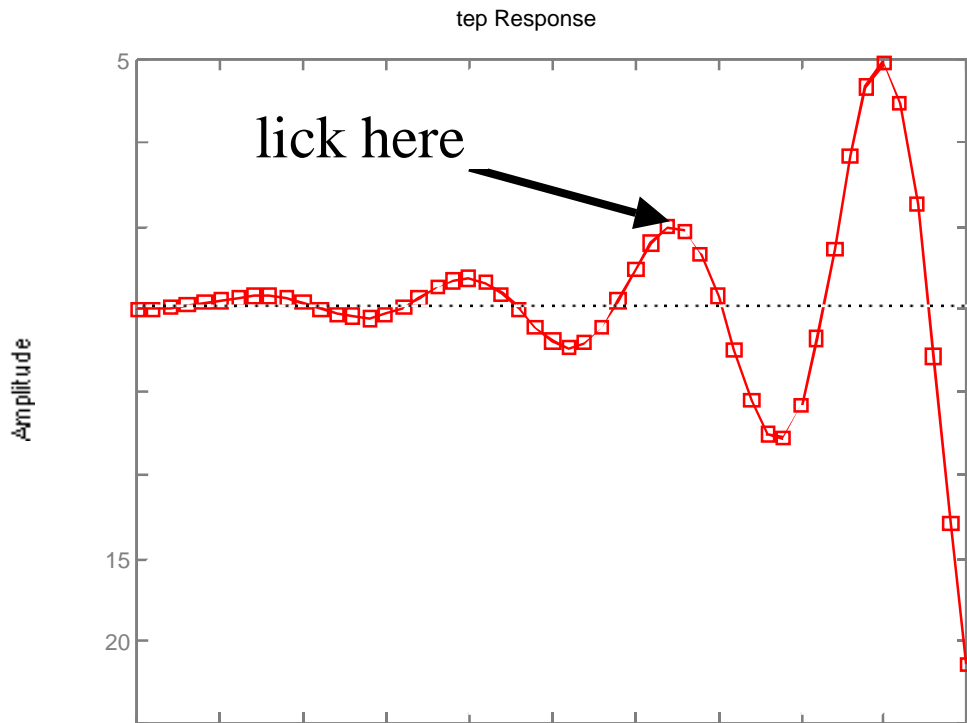
Within the **menutime** menu, select "(C) Display coordinates of mouse click."

Type in your choice:**c**

It will then provide the following instructions and switch you to the plot window with a set of cross-hairs.

Left-Click on a point to display coordinates in the command window.  
Right-Click to label the point on the plot itself.  
Press ENTER while on the chart to Quit.

Since this function can't detect if a line is impulse or step, it will return the coordinates of exactly where you clicked, rather than return coordinates of the closest point on the line. So, use a steady hand. For every point you click, the coordinates will be printed to the command window as seen in figure 16.

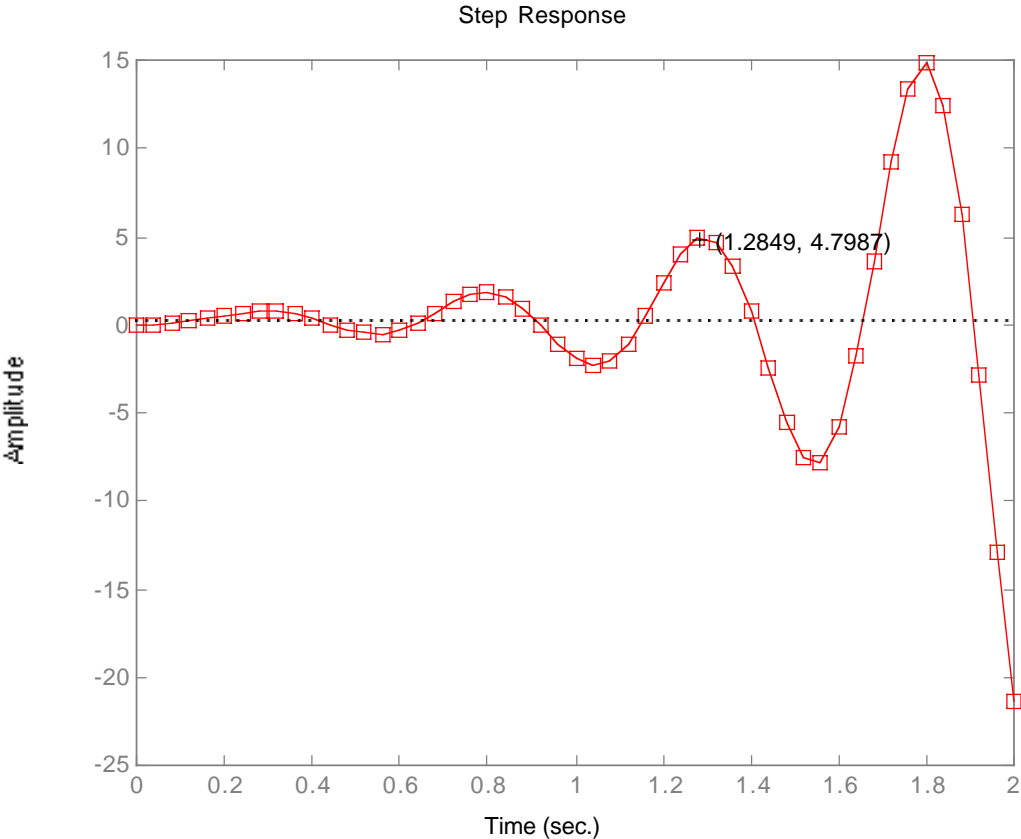


**Figure 16: Clicking for information**

Shown in the command window:  
 $T(\text{sec})=1.2849$ ,  $\text{Amplitude}=4.7987$

You can continue to click and list coordinates until you press ENTER while having the plot window active. This will return you to the menu in the command window.

Now, to **label** the coordinates of the plot, use the right mouse button instead. The best way to go about this is to position the plot so that the command window is visible. Repeatedly **left**-click on the plot until you have the mouse pointer positioned over the coordinates you wish to label, then click the **right** mouse button. The target will be labeled with a black "+" and the coordinates will be labeled directly on the plot. See Figure 17.



**Figure 17: Labeling the time response plot**

Related: **menubode, menurlocus, menutime**

## Zoom in and out of plots from "menutime"

In order to more closely examine certain regions of the curve, menutime offers a zoom option. At the menu, select "Z) Zoom in on the chart." Then select which axis you wish to zoom. In this case, the plot will be zoomed on both axis.

Type in your choice: **z**

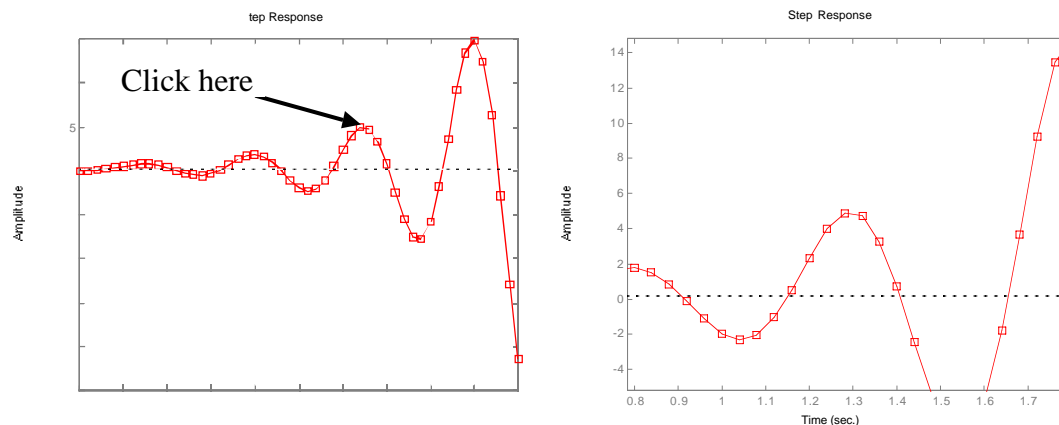
Zoom (X)-axis, (Y)-axis, or (B)oth?: **b**

Begin clicking on the areas to zoom in on...

Right-click zooms out

Zoom will be active until another choice is selected

Notice that it will immediately return you to the menu list. Don't worry; the zoom function will remain active until you select a different menu item. To zoom in, simply click on the region to zoom into. Notice that you can not zoom out farther than what was originally plotted. See Figure 18.



**Figure 18: Zooming in the time response plot**

Related: [menubode](#), [menurlocus](#), [menutime](#)

## Analysis Functions

### ***routh: for calculating stable gain ranges***

syntax: `gain=routh(g)` where `g` is a transfer function.

The assignment, `gain`, is optional and will result with an array of the gain ranges where `g` is stable with the following format: `[klow khigh]` or `[klow1 khigh1 klow2 khigh2]` if more than one stable range is found.

This function finds the range of gains where the transfer function is stable based on a root locus analysis. Note: this function only searches gains of -1000 to +1000.

Let's say you have the following transfer function:

`g=`

Transfer function:

$$1000 s + 50$$

-----  
$$s^4 + 15 s^3 + 75 s^2 + 165 s + 104$$

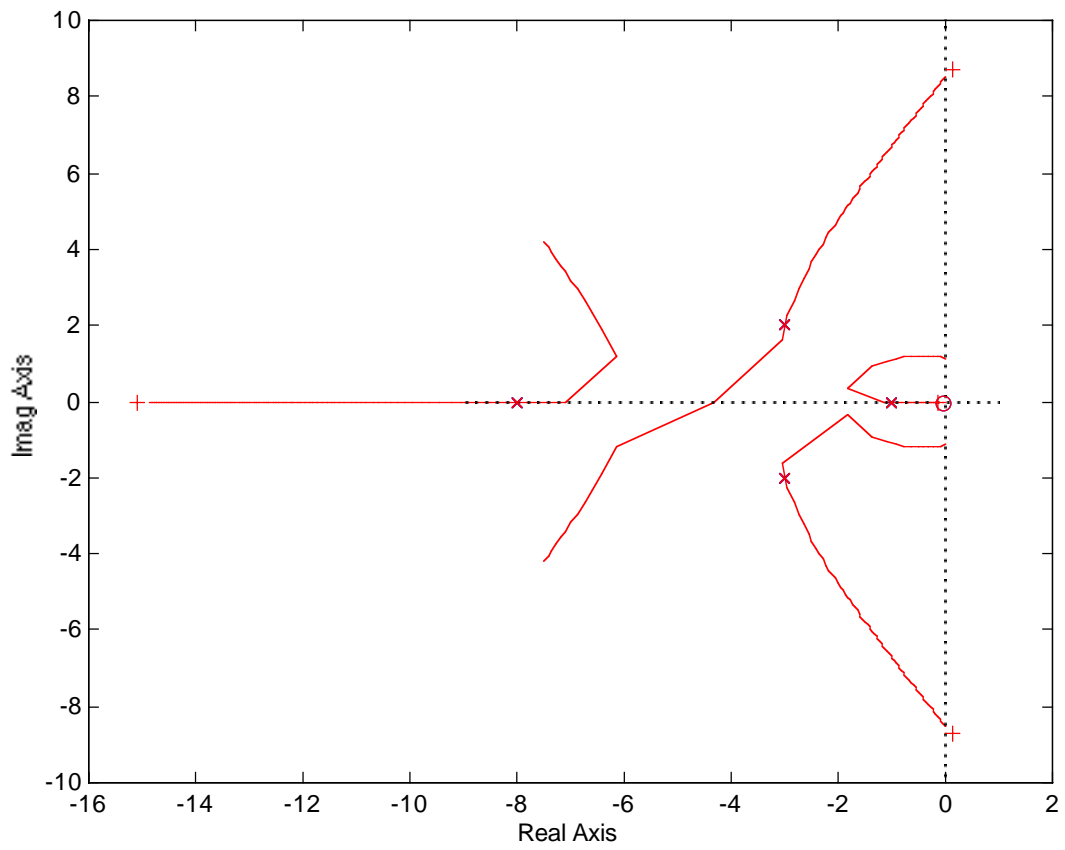
Typing in the command:

`»routh(g)`

will give you the following result:

Stable for gain ranges: -0.14531 to 0.92906

To confirm this, look at the root locus plot in figure 19. Notice how it touches the imaginary axis right at those two gain values.



**Figure 19: Root locus of  $g$  for gain of  $-0.14531$  to  $0.92906$**

Related: [survey](#), [ezmargin](#)

## **ezmargin: for calculating gain and phase margins**

syntax: `[GMinc, GMred, PM] = ezmargin(g)` where `g` is a transfer function.

The assignment, `[GMinc, GMred, PM]`, is optional. It results with the following:

`GMinc` = array of `[GM, Wl180]` where GM is the increase gain margin in dB at `freq=Wl180` rad/sec.

`GMred` = array of `[GM, Wr180]` where GM is the reduction gain margin in dB at `freq=Wr180` rad/sec.

`PM` = array of `[PM, Wc]` where PM is the phase margin in degrees at crossover `freq=Wc` rad/sec.

This function gives you the correct margins for a transfer function. Matlab comes with a built-in function, `margin`, but it does not give negative margins or the reduction gain margin. Although `ezmargin` is a little slower, it gives more information.

As an example, let's say you gave the following transfer function:

```
G=
Transfer function:
          1000 s + 50
-----
s^4 + 15 s^3 + 75 s^2 + 165 s + 104
```

Typing in the command:

```
»ezmargin(g)
```

will give you the following result:

```
Stable for gain ranges: -0.14531 to 0.92906
Increase GM=-0.63914dB (0.92906), at Wl180=8.5403 rad/sec
Reduction GM=NaNdB (NaN), at Wl180=NaN rad/sec
Pm=-2.328deg, at Wc=8.8106 rad/sec
```

The first line gives you the possible gain ranges that `g` is stable in. The second line gives you the increase gain margin and frequency where the phase drops below the 180deg line. The third line gives you the reduction margin and frequency where the phase first rises above the 180deg line. The last line gives you the phase margin and frequency for where the magnitude drops to 0dB and below.

When "NaN" is given as a value, it means that there is no such value. For instance, the gain margin will equal "NaN" if the phase line never drops below 180 degrees.

As another example, let's say you have the following function:

```
G=
Transfer function:
          200 s + 10
-----
s^4 + 15 s^3 + 75 s^2 + 16 s + 104
```

Typing in the command:

```
» [GMinc, GMred, PM] =ezmargin(g)
```

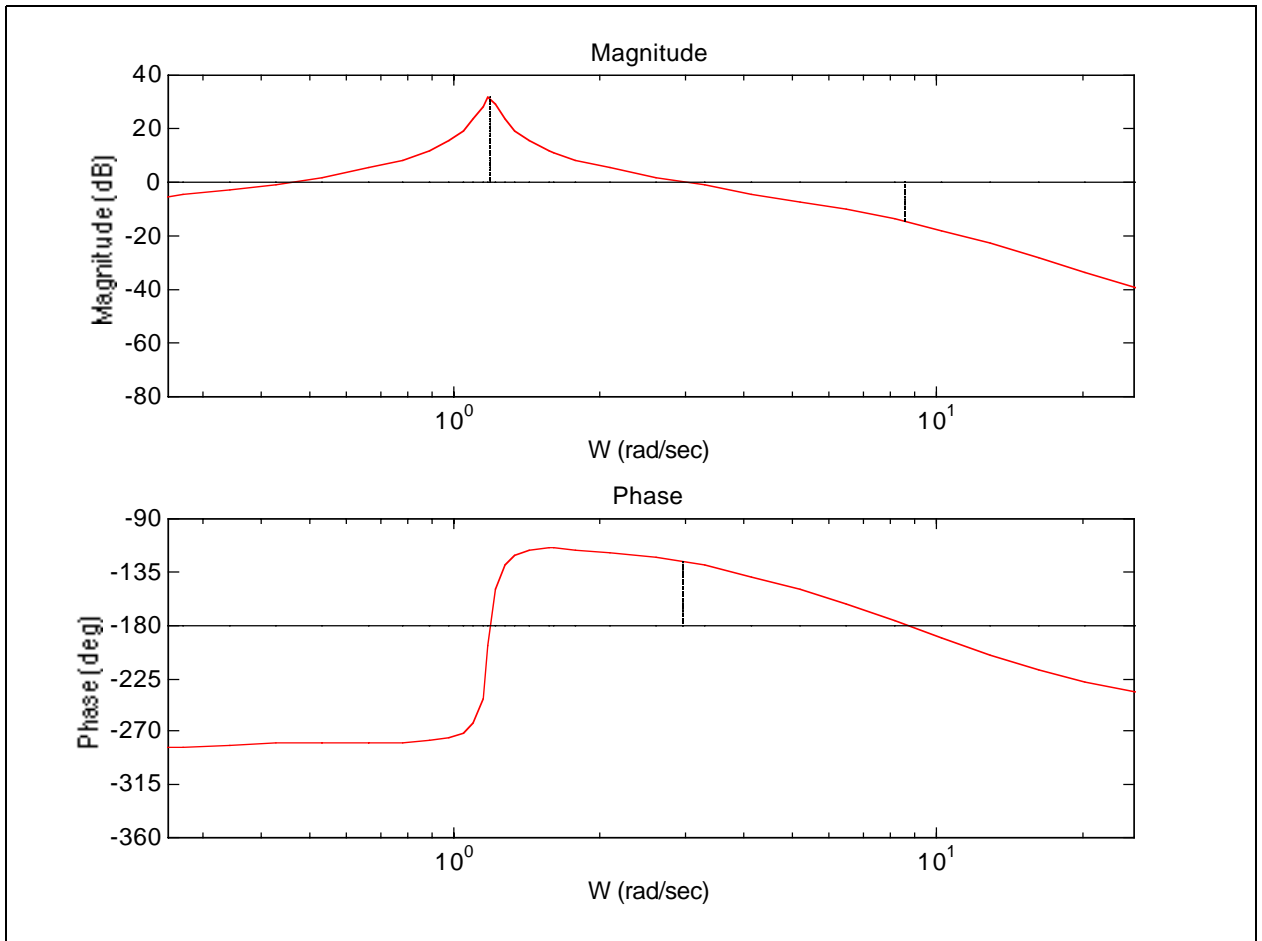
will give you the following result:

```
Stable for gain ranges: 0.026271 to 5.3825
Increase GM=14.6196dB (5.3825), at W180=8.5342 rad/sec
Reduction GM=31.6106dB (38.0656), at W180=1.1904 rad/sec
Pm=53.1799deg, at Wc=2.9796 rad/sec
```

```
GMinc =
    14.6196    8.5342
GMred =
    31.6106    1.1904
PM =
    53.1799    2.9796
```

Notice how each variable is a 2D array with the margin and frequency that it occurs at.

To confirm these values, look at the bode plot in figure 20 created from the `menubode` margins command. Notice that the top frame shows the reduction margin and increase gain margin (both positive). The bottom frame shows the phase margin.



**Figure 20: Bode plot of  $g$  with reduction and increase gain margins above, phase margin below.**

Related: [menubode](#), [menurlocus](#), [menutime](#)

## ***survey: for an overall analysis***

syntax: **survey(g)** where **g** is a transfer function.

This function does a multitude of analysis on a single transfer function without you having to execute each command individually. Let's say you wish to do an analysis of the following transfer function **g**:

```
>>g
```

```
Transfer function:
```

```
          1000 s + 50
-----
s^4 + 15 s^3 + 75 s^2 + 165 s + 104
```

Type in the following command:

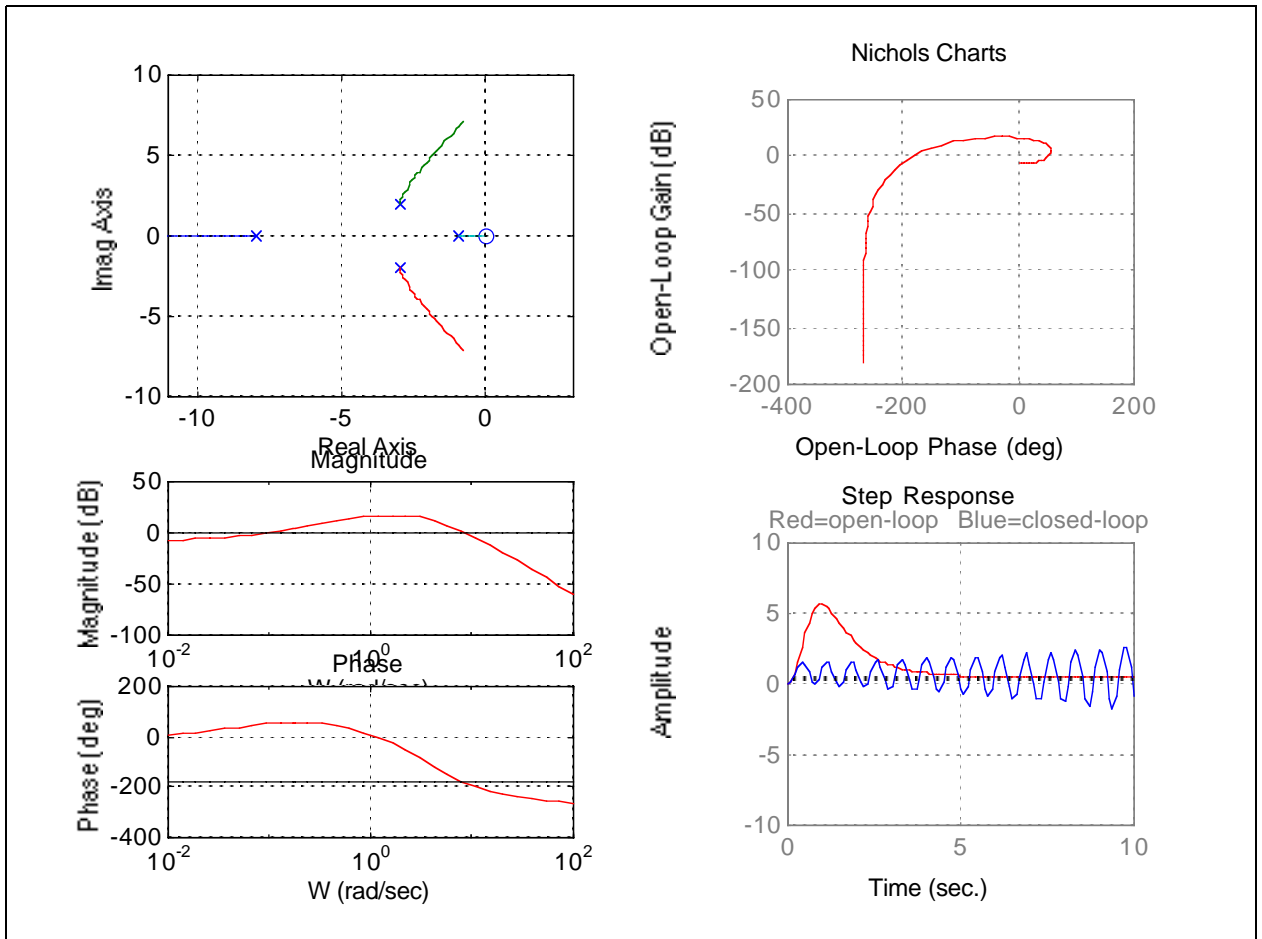
```
>>survey(g)
```

First, it will present the transfer function in shorthand notation. Then it will give the possible stable gain ranges. Next, it will provide the gain and phase margins and the accompanying frequencies.

```
1000 (0.05)
-----
(1) [0.83205, 3.6056] (8)
```

```
Stable for gain ranges: -2.08 to 0.92906
GM=-0.63913dB (0.92906), at W180=8.5403 rad/sec (Hz)
Pm=-2.328deg, at Wc= 8.8106 rad/sec (Hz)
```

Finally, it will display a four-part window providing the root locus, nichols, bode, and step response plots as shown in Figure 21.



**Figure 21: Survey window showing root locus, nichols, bode, and step response.**

This provides a quick way to get an overview of the stability and response of a function.

Related: [menubode](#), [menurlocus](#), [menutime](#)

## systemroot: for Solving a Solvent Matrix

syntax: `result=systemroot(outfactor, infactor)`

where `outfactor` is the solvent matrix, `infactor` is the matrix of control deflections, and `result` is a cell matrix of the control-ratio transfer functions.

This function gives the control ratios derived from the root of a controls system. This system consists of 2 matrices created from the linearized controls equations: a 2X2 or 3X3 cell matrix with the factors stored as polynomials, and another matrix with the force elements. Both matrices are "cell" matrices (Use { } instead of []). Let's say you have a system as follows:

$$\begin{array}{|c|c|c|} \hline (-0.344) & (4.71s) & (4.71s + 0.6) \\ \hline (0.02725s^2 + 0.0553s) & (-0.0128s) & (0.057) \\ \hline (0.00338s) & (0.058s^2 + 0.0158s) & (-0.096) \\ \hline \end{array} \quad \times \quad \begin{array}{|c|} \hline \phi(s) \\ \hline \psi(s) \\ \hline \beta(s) \\ \hline \end{array} =$$
  

$$\begin{array}{|c|c|} \hline 0 & 0.171 \\ \hline 0.6 & 0.0131 \\ \hline -0.01 & -0.08 \\ \hline \end{array} \quad \times \quad \begin{array}{|c|} \hline \text{delta.a}(s) \\ \hline \text{delta.r}(s) \\ \hline \end{array}$$

The first matrix is the airframe dynamics (solvent) matrix (A).

The second is the state vector (X).

The third is the controls dynamics matrix (B).

The fourth is the controls matrix (U).

(1) To create the outfactor matrix, use the "cell" format as follows:

```
>>A={-.344, 4.71*s, (4.71*s+0.6)
      (.02725*s2+.0553*s), -.0128*s, 0.057
      .00338*s, (.058*s2+.0158*s), -0.096}
```

```
A =
    [-0.3440]    [1x1 tf]    [1x1 tf]
    [1x1 tf]    [1x1 tf]    [ 0.0570]
    [1x1 tf]    [1x1 tf]    [-0.0960]
```

```
>>B={0, .171
      .6, .0131
      -.01, -.08}
```

```
B =
```

```

[      0]    [ 0.1710]
[ 0.6000]    [ 0.0131]
[-0.0100]    [-0.0800]

```

(2) use the function as follows

```

>>results=systemroot (A,B)
results =
[1x1 tf]    [1x1 tf]
[1x1 tf]    [1x1 tf]
[1x1 tf]    [1x1 tf]

```

notice that results is a cell matrix with the control ratios:

(3) To get (phi/delta.r) which is the roll due to rudder deflection:

```

>>results{1,2}

```

Transfer function:

$$\frac{0.003579 s^3 - 0.003958 s^2 - 0.01599 s}{0.007444 s^5 + 0.01808 s^4 + 0.01882 s^3 + 0.0276 s^2 - 0.0001129 s}$$

(3b) Or, to get (beta/delta.a) which is the slip due to aileron deflection:

```

>>results{3,1}

```

Transfer function:

$$\frac{0.001283 s^3 + 0.02413 s^2 + 0.003217 s}{0.007444 s^5 + 0.01808 s^4 + 0.01882 s^3 + 0.0276 s^2 - 0.0001129 s}$$

Related: N/A

## Exchang Transfer Functions with CC

The aerocontrols toolbox provides the ability to exchange transfer functions between Matlab and Peter Thompson's CC controls-analysis program. This allows you to seamlessly use tools from both Matlab and CC to analyze transfer functions.

### ***getcc: to put a CC function into Matlab***

syntax: `gmat=getcc('gcc')` where :  
gmat = the matlab function name to be created  
gcc = the CC function name to be converted

This function enables you to retrieve a transfer function from CC and assign it to a Matlab variable.

### Set up getcc to work for the first time

Before `getcc` can work, you need to specify the path to CC's FILES directory. To do so, you need to edit the getcc file. Start by typing in the following to edit the getcc file:

```
>>edit getcc
```

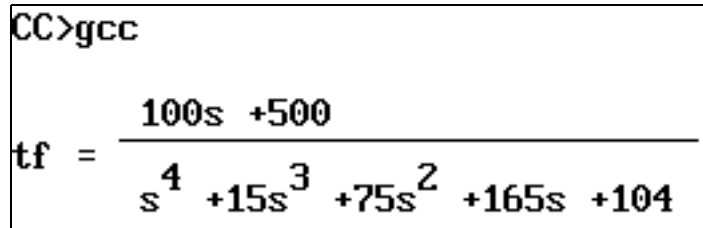
A window of getcc will popup. Look for the following lines:

```
% Edit this path to the CC variables. ":" for Mac, "\" for PC.  
ccpath='Sorenson:Applications:Aero:CC:FILES:'
```

Type in the path to the CC FILES directory using "\" on a PC or ":" for a Mac. In this example, the path to CC's FILES directory is Sorenson:Applications:Aero:CC:FILES:. Save and close the file. It's now ready to use.

## Use getcc

Let's say you have a function called "gcc" in CC, such as in Figure 22.



```
CC>gcc
tf =
      100s + 500
-----
      s4 + 15s3 + 75s2 + 165s + 104
```

**Figure 22: Transfer function in CC to be transferred to Matlab**

To transfer this function to a variable named "gmat" in Matlab, type in the following command in matlab:

```
»gmat=getcc('gcc')
```

Matlab will now have the same transfer function saved in the variable "gmat."

```
»gmat
```

```
Transfer function:
```

```
      100 s + 500
```

```
-----
s4 + 15 s3 + 75 s2 + 165 s + 104
```

Related: [putcc](#)

## ***putcc: to put a Matlab function into CC***

syntax: `putcc(gmat, 'gcc')` where :  
`gmat` = the matlab function source to send  
`gcc` = the CC function to be created

This function enables you to send a transfer function from Matlab into CC.

### Set up putcc to work for the first time

Before `putcc` can work, you need to specify the path to CC's FILES directory. To do so, you need to edit the putcc file. Start by typing in the following to edit the putcc file:

```
>>edit putcc
```

A window of putcc will popup. Look for the following lines:

```
% Edit this path to the CC variables. ":" for Mac, "\" for PC.  
ccpath='Sorenson:Applications:Aero:CC:FILES:'
```

Type in the path to the CC FILES directory using "\" on a PC or ":" for a Mac. In this example, the path to CC's FILES directory is Sorenson:Applications:Aero:CC:FILES:. Save and close the file. It's now ready to use.

## Use putcc

Let's say you have a function called "gmat" in Matlab that you wish to send to CC.

```
»gmat
```

```
Transfer function:
```

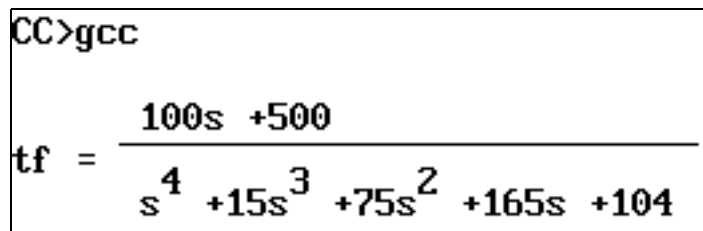
$$100 s + 500$$

-----  
 $s^4 + 15 s^3 + 75 s^2 + 165 s + 104$

To transfer this function to a variable named "gcc" in CC, give the following command:

```
»putcc(gmat, 'gcc')
```

To confirm that this worked, run CC. You should get results like in figure 23.



```
CC>gcc
```

$$tf = \frac{100s + 500}{s^4 + 15s^3 + 75s^2 + 165s + 104}$$

Figure 23: Transfer function in CC transferred from Matlab

Related: [putcc](#)

## Miscellaneous Functions

These are leftover functions that are not necessarily controls-oriented that are tossed in for the purpose of making things a little easier.

### ***dispmat: for displaying matrices***

syntax: `dispmat(M)` where **M** is a large matrix.

This function takes a large matrix and displays it in a more readable fashion. It displays it without any wrapping. Also, it applies scientific notation to each number, as necessary, rather than apply multiple to the entire matrix.

## Resources

Biezad, Dan Aero 550: Study Guide for Flight Control Systems, El Corral Bookstore, ©N/A.

Huber, Tom Envision-It! Homepage, Gustavus Adolphus College, revised Jan 1997.

Mathworks Inc. Matlab 5.1, software that the toolbox was created for, ©1984-1997.

Nelson, Robert C. Flight Stability and Automatic Control 2<sup>nd</sup> Ed., WCB/McGraw-Hill, ©1998.

Nise, Norman S. Control Systems Engineering 2<sup>nd</sup> Ed., Addison-Wesley Publishing Company, ©1995.

Thompson, Peter CC, program used to verify Aerocontrols Toolbox, ©1984-1989.