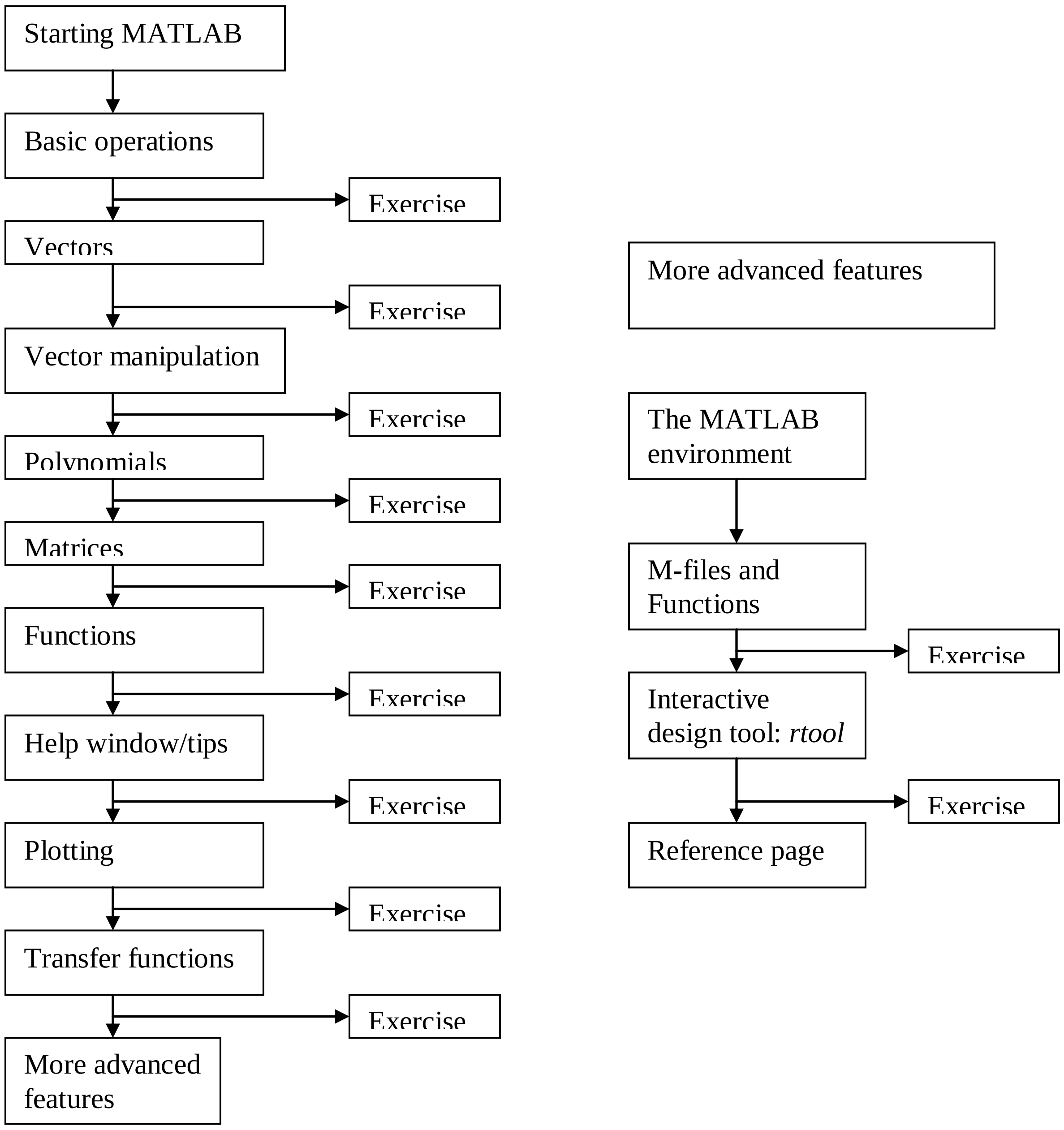
Software Toolkit: MATLAB

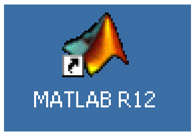
1) Introduction to MATLAB:

MATLAB is a high level language for technical computing, which is often used by engineers to help them design systems or analyze a system's behavior. Following is a simple beginner guide for the MATLAB package.



2) Starting MATLAB:

To run MATLAB, move the cursor to MATLAB icon and double-click on the left-hand mouse button.



The MATLAB desktop will open. This is an integrated development environment for working with MATLAB suite of toolboxes and programs. There are three open windows, which represent:

- Command window
- Launch pad and workspace
- Command history and current directory.

Command window: We type all our commands in this window at the prompt (>>) and press 'enter' key to see the results of our operations.

3) Basic Operations:

We can assign a numerical value (data) to a variable using the equal (=) sign.
For example,

```
>> a=2

a =

    2

OR

>> b= -2.5

b =

   -2.5000
```

4) Vectors:

Enter each element of the row vector (separated by a space) between square brackets, and set it equal to a variable. For example, to create the row vector x, enter into Matlab:

```
>> x=[1 2 3 4]

x =

     1     2     3     4
```

To enter a column vector, separate the elements by a semicolon '; '.
For example:

```
>> y=[1;2;3;4]

y =

     1
     2
     3
     4
```

We can determine the size of the vectors x and y by using the size command.

```
>> size(x)

ans =

     1     4

>> size(y)

ans =

     4     1
```

If we want to create a vector with elements between 0 and 5 evenly spaced in increments of 0.5, we can use:

```
>> t=0:0.5:5

t =

  Columns 1 through 7

       0    0.5000    1.0000    1.5000    2.0000    2.5000    3.0000
  Columns 8 through 11

    3.5000    4.0000    4.5000    5.0000
```

5) Vector Manipulation

Manipulating vectors is almost as easy as creating them. Try the following operations:
>> *a=[1 2 3 4]*
>> *x=a+2*   Result:
>> *x=a-2*   Result:
>> *x=a*2*   Result:
>> *x=a/2*   Result:

Let b= [1 2 3].
What happen if we try to add two variables (vectors: a+b) of different dimensions? What is the error message?

(We note that in the example x=a+2, variable a is size 1 x 4 and 2 is size 1 x 1. Technically they could not be added together, as the dimensions are not compatible. However, MATLAB assumers, when you ass/ subtract/multiply/divide a variable by a number, that all elements of the vector should be operated on. This is not true with two 'variables', as we see now.

For x=[1 2 3 4] and y=[5;6;7;8], would the following operations be acceptable?

x+y ?   Yes/no
x*y ?   Yes/ no
y*x?   Yes/no

In this we note that we could not add x+ y since they do not have appropriate dimensions. However, if we transpose ***x=x'*** (where ' indicates the transpose), we can then evaluate:
x+y'
The size command could be used to verify the dimensions of the vectors.

6) Polynomials

We can enter the coefficients of a polynomial as a row vector. The coefficients should be entered in descending order of the powers of x. For example:

$p(x)=x^2 + 10x + 3$

Can be represented by:

p= [1 10 3]

Matlab can interpret a vector of length n+1 as the coefficients of an nth order polynomial. Thus, if a polynomial is missing any coefficients, we must enter zeros in the appropriate place in the vector. For example

q(s) = 6s^4 + 1
would be represented in MATLAB as:
q = [6 0 0 0 1]

Finding roots: Use the following command:

Roots ([6 0 0 01]) or ***roots(q)***

Multiplying/ dividing polynomials:
The product of two polynomial is found by taking the convolution of their coefficients. The function ***conv*** will do this:

Polynomial: X(s) = s+2	Matlab representation: ***x = [1 2]***
Polynomial: Y(s) = s^2+4s+8	MATLAB representation: ***y = [1 4 8]***
Polynomial: Z(s) = (s+2)(s2+4s+8)	MATLAB representation: ***z = conv(x,y)***

If Z(s) = Q(s)Y(s) + R(s), then given Z(s) and Y(s) the ***deconv*** function will return the result, Q, as well as the remainder R. Note that if there is more than one output for a function, they should be put inside square brackets and be separated by commas. Try to divide Z(x) by Y(x):
[Q R] = ***deconv(x,y)***
How do we check the result?

7) Matrices

Entering matrices in to Matlab is the same as entering a vector, except that each row of elements is separated by a semicolon. Try:
B = [1 2 3 4; 5 6 7 8; 9 10 11 12]

Matrices in Matlab can manipulated in many ways. For example, we can find the transpose of a matrix using the apostrophe key (‘). Try:

C = B’
Remember that the order of multiplication is important when dealing with matrices. Would the following operations be allowed?

D= B + C	yes/no
D = B +C’	yes/no
D = C*B	yes/no

D = B*C	yes/no
E = B^3 (power of a matrix)	yes/no
X = inv(B) (inverse of a matrix)	yes/no

8) Functions

Matlab includes many standard functions (and constants). Each function is a block of code that accomplishes a specific task.

Common functions
Sin, cos, log ($\log_e$), exp, sqrt, mean, std (standard deviation)

Common constants
Pi = ∏ returns 3.1416.The variables i or j represent the square root of –1 (complex numbers).
Try the following sine function:
X= sin (pi/4) result =?

We wish to plot the sine wave of frequency 3rad/sec and amplitude 1 over a period of time between 0 to 20 seconds (in steps of 0.1 sec):

(a) Write down the sine wave expression y(t)=?
(b) Calculate the vector y for the values of time given. Graph the results using an appropriate plot command.

Function inputs and outputs

Each function in MATLAB has a number of inputs and outputs variables. We should define all the necessary inputs before we use a function. We can use the ***help*** function to obtain information about any function we want to use. For example, if we wanted to find out about the use of the ***abs*** function, we could type

Help abs

Matlab returns:

```
ABS   Absolute value.
   ABS(X) is the absolute value of the elements of X. When
   X is complex, ABS(X) is the modulus (magnitude) of X.
```

Therefore by typing
Y=abs(x);
The variable y would hold the absolute values of x.

If we want to find the modulus and the angle of a complex number, we can enter the following:

S = j*pi/4 define the complex number s
G = 1/s define complex function g
Mag = abs(g) find the magnitude: output: mag input: g
Ang=angle(g) find the angle: output: ang input: g

Once we have defined the output variables, they will be stored in Matlab workspace and we can manipulate them as wish.

9) Help window/ tips

To determine the usage of any function, enter: ***Help*** function name

Try: >> ***help sin***

To see the help window, type >>***help win***

It lists the directories for help for many of the other toolboxes that come with Matlab that we may use later.

For example, click on Matlab\el fun to see some basic functions. With more practice we can write special MATLAB files (**m-files**) to save re-typing the same commands.

10) Plotting

It is easy to create plots in MATLAB. Suppose we make a time vector and then compute the sine values of the vector at each time point.

t= 0:0.05:10
y = sin(t)
Plot(t,y)

Plot command has extensive add-on capabilities. Some useful features are given here.

Figure command

When we plot a graph, MATLAB opens a window called the figure Window. Every time we plot a graph, this figure window is updated. If we want to keep the old graphs, we can open a new window by using the command.

>> ***figure***

Try: >> ***plot(t,y)*** >> ***figure*** >> ***plot(t,2*y)***

Plotting several responses on same axis

Plot the first response graph required and then enter the command:
>> hold on

Try these commands:
>> figure(2)
>> plot(t,y)
>> hold on
>> plot(t,2*y)
>> plot(t,0.5*y)
Enter hold off to return to the default mode.

Finding the coordinates of a point on a graph

For graphs using the plot command, use
>> ginput(N)
N is the number of points at which coordinates may be found.
This command will give a cursor which can move to a point on the figure using the mouse, and then press the left-hand button to see the coordinates in MATLAB windows. Alternatively, click on any point on the graph and a text window pops up with the coordinates' value.

Labeling plots and axes

We can use the insert pull-down menu to place Text, labels and lines on a graph.
Use the sine wave as previously generated.

```
t=0:0.05:10;
y=sin(t);
plot(t,y)
```

And then insert:
X Label, Ylabel, Title, Text, Line.

Frequency response graphs
The command ***semilogx*** is the same as plot except that a logarithmic (base 10) scale is used for the x-axis. This command is used in the exercise below.

11) Transfer functions in MATLAB

A transfer function can be entered in MATLAB using different commands.

Method 1:

1. We can use the command ***tf(num,den),*** where num and den are vector coefficients of the numerator and denominator polynomials, respectively. Enter the TF:

g1 = (3s+1}/(s^2+3s+2)
num = [3,1];
den = [1 3 2]
g1 = tf(num,den)

2. Alternatively, we can define '***s***' as a TF variable and then use it to 'write' the transfer function directly.

   s = tf('s')
   g1 = (3*s+1)/(s^2+3*s+2)

Method 2

1. We can use the command ***zpk(zeros,poles,gain),*** where zeros, poles and gain are vectors of the zeros, poles and gain of the transfer function. For example, we can enter the following transfer function in the form:

$$g2 = \frac{4(s+5)}{(s+3)(s+10)}$$

   >> ***zeros = [-5]***
   >> ***poles = [-3,10]***
   >> ***gain = 4***
   >> ***g2 = zpk(zeros,poles,gain)*** enter the transfer function
   or equivalently
   >> ***g2 = zpk([-5], [-3 –10],4)***

2. Alternatively, we can define s as a transfer function and then use it to 'write' the transfer function directly:

s = zpk('s')
g2 = 4*(s+5)/((s+3)*(s+10))

If we have a factor with complex poles in the transfer function,

$$g3 = \frac{4(s+5)}{(s+2+3i)(s+2-3i)} = \frac{4(s+5)}{(s^\wedge 2+4s+13)}$$

The complex roots of the denominator can be entered easily as:
g3 = 4*(s+5)/((s+2+3*j)*(s+2-3j))

11.1 Transfer function manipulation

Once various transfer functions have been entered, we can combine them together. Use the transfer functions
g6 = 1/(s+2) and g7 = 5/(s+3)

Operation	Result
Addition	***g6+g7***
Subtraction	***g6 – g7***

Multiplication	***g6*g7***
Division	***g6/g7***
Combination of Operations	***g6/(1+g6)***

11.2 Time responses

If g represents a system transfer function, then:
1. For a unit impulse response use the command: ***impulse(g)***
2. For a unit step response, use the command***:*** ***step(g)***
3.For a step response of magnitude K, simply multiply the transfer function, g, by the numerical value of K: ***step(K*g)***

For some functions, such as ***step*** or ***bode***, MATLAB graphs are automatically generated. For, these graphs, we can use the cursor to find the coordinates of different points.
Example: Type the following

```
>> s = tf('s')
>> g = 1/(s+1)
>> step(g)
```

11.3 Poles and zeros

1. To see the roots of the numerator (called the zeros) of g(s) use the command: ***zero(g)***
2. For the roots of the denominator (called the poles) of a transfer function use the command: ***pole(g)***
3. For a pole-zero map use the command: ***pzmap(g)***

11.4 Feedback control

To determine the closed-loop control transfer function from a system with negative feedback we can use the command:

G = feedback(g,h)

Where g is the forward transfer function and h represents the transfer function of the components in the feedback path. If positive feedback is required then the following command can be used:

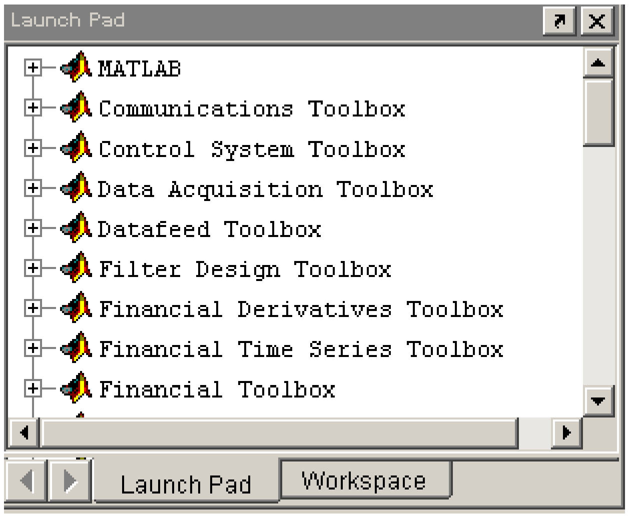
G = feedback(g,h,1)

12) MATLAB environment

We return now to examine the MATLAB window environment. This can be achieved choosing View, Desktop layout and then default. The default Desktop windows are displayed once more.

Launch Pad:

We use this window to execute MATLAB demos and programs. It has a tree structure similar to windows explorer. Double clicking on a displayed icon runs the program associated with it.



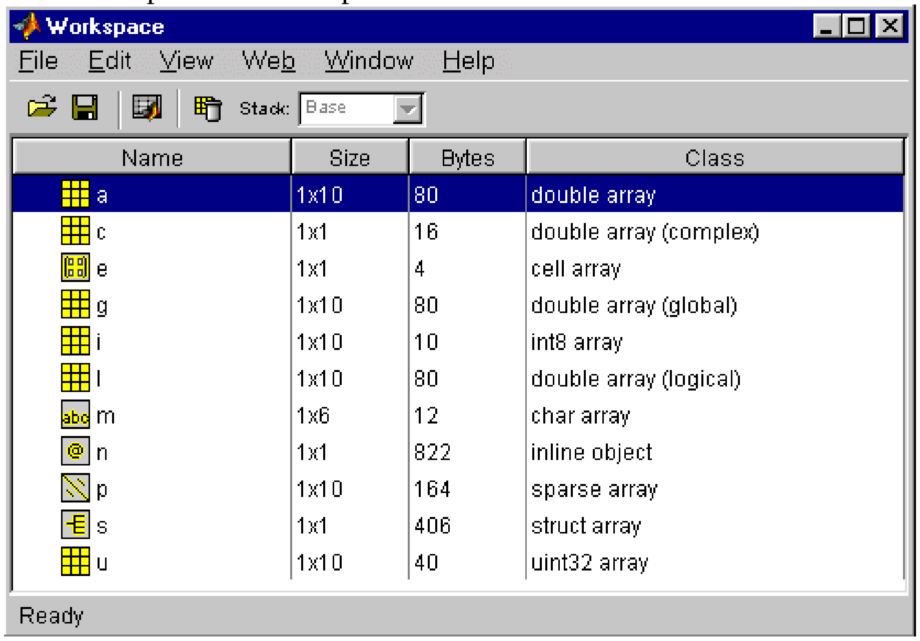
Workspace Browser

Use the Workspace browser to perform operations on the MATLAB workspace. Equivalent functions are available and are documented for each feature of the Workspace browser.

To open the Workspace browser, do one of the following:

i. From the View menu in the MATLAB desktop, select Workspace.
ii. In the Launch Pad, under MATLAB, double-click Workspace.
iii. Type ***workspace*** at the Command Window prompt.

The Workspace browser opens.



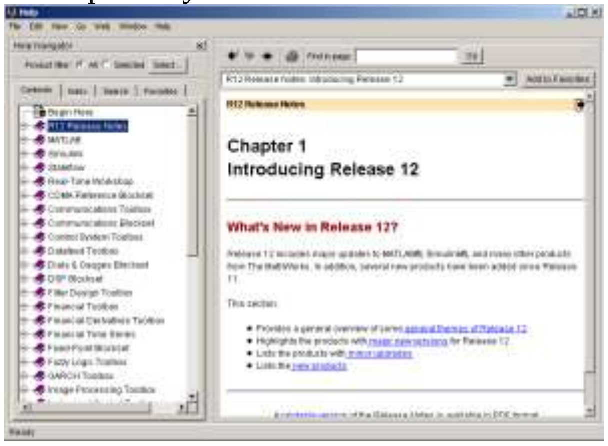
Name	Size	Bytes	Class
a	1x10	80	double array
c	1x1	16	double array (complex)
e	1x1	4	cell array
g	1x10	80	double array (global)
i	1x10	10	int8 array
l	1x10	80	double array (logical)
m	1x6	12	char array
n	1x1	822	inline object
p	1x10	164	sparse array
s	1x1	406	struct array
u	1x10	40	uint32 array

Command history window:

This window contains a record of all the commands that we type in the command window. By double clicking on any command, we can execute it again.

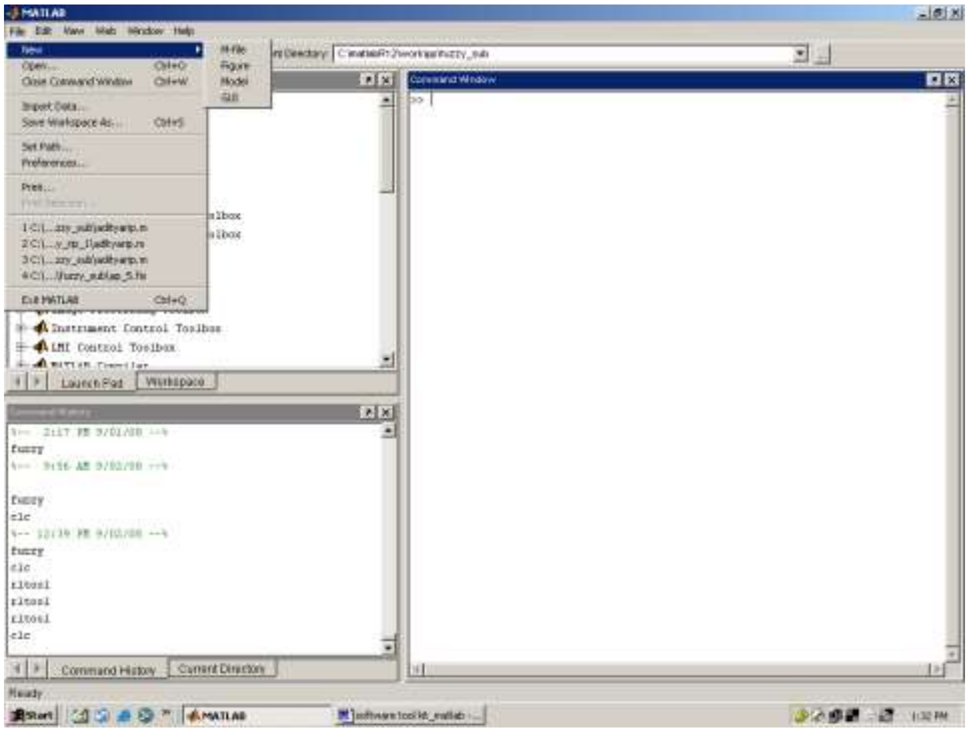
Current directory window:
This window shows the directory on the hard disk, which is used to run or save our programs. The default directory is the MATLAB\work.

Help window:
This can be activated by choosing 'Help' (or F1) from the menu. We can use it to search for help on any command or function.



13) M-files and functions

We can create files that contain MATLAB code that save us the effort of rewriting the commands. These files are called M-files. We can create M-files using the Matlab editor or any other text editor. Using the MATLAB editor has the advantage that it can also be used as a debugger to find any possible errors in our programs.



We can write two types of M-files:
1) Programs that do not need any input or output arguments. They operate on the data in the workspace.
2) Functions which require input data and return output arguments. Any internal data in the function will then be local to the function and cannot be accessed from the workspace.

Example:
Write an m-file which enters the following transfer function, plots the two step responses corresponding to k = 1 and k = 2 on the same plot and titles the plot.

G1(s) = k/(s+1)

Select **file**, **New** and the **M-file** to open the editor,
Enter the following code

```
S=tf(‘s’);
k=1;
g=k/(s+1);
step(g);
hold;
k=2;
g=k/(s+1);
step(g);
title(‘step response for k=1 and k=2’)
```

Now select **File**, **save as** and save the file as twoplot.m in the working or your own directory. Return to the Matlab command window. By typing

```
>> Twoplot
```

at the Matlab prompt it should run and produce the plot.

14) SISO design tool: rltool

The SISO Design Tool
The SISO Design Tool is a graphical user interface (GUI) that facilitates the design of compensators for single-input-single-output feedback loops. The SISO Design Tool allows you to iterate rapidly on your designs and perform the following tasks:

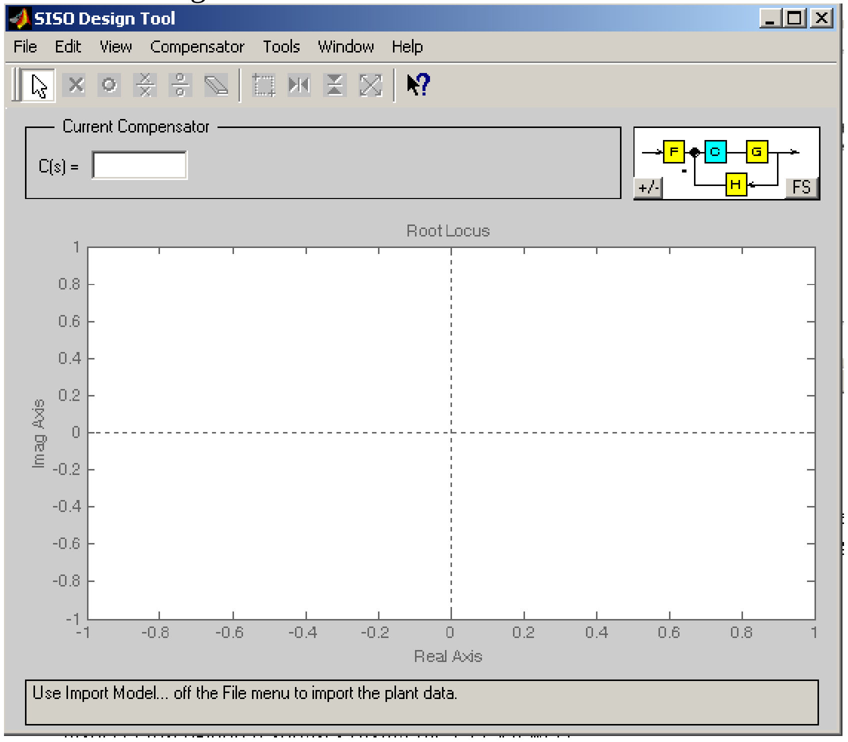
1. Manipulate closed-loop dynamics using root locus techniques
2. Shape open-loop Bode responses
3. Add compensator poles and zeros
4. Add and tune lead/lag networks and notch filters
5. Inspect closed-loop responses (using the LTI Viewer)
6. Adjust phase and gain margins

7. Convert models between discrete and continuous time

Opening the SISO Design Tool

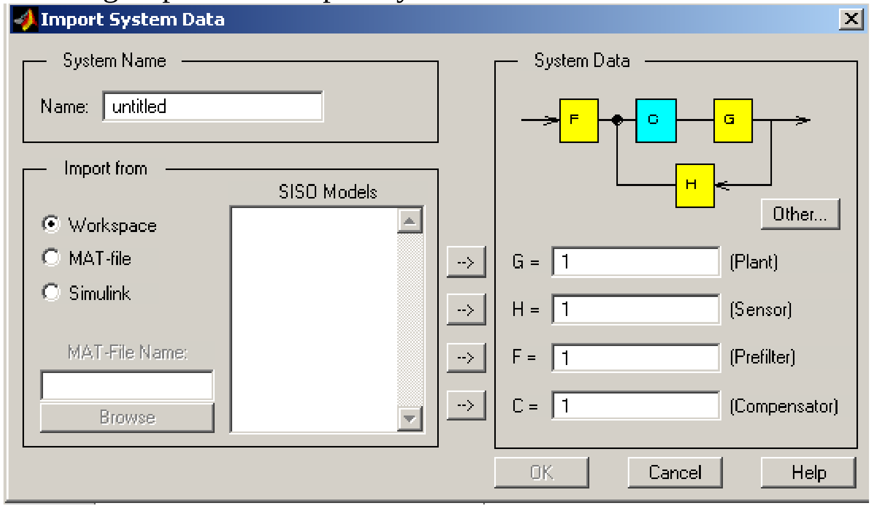
SISO design tool is activated by typing ***rltool*** in the Matlab command window. This provides a convenient method for linking time, frequency and root locus designs together.

The SISO design tool window:



Use Import Model... off the File menu to import the plant data.

We can import models into the design tool by clicking on the file menu and then choosing Import. The Import system Data window:



Name: untitled

Import from: [✓ Workspace] MAT-file Simulink

G = 1 (Plant)

H = 1 (Sensor)

F = 1 (Prefilter)

C = 1 (Compensator)

There is a basic feedback loop with the transfer function blocks representing the plant or process, G, the sensors or measurement, H, the compensator or controller, C, and a prefilter, f, which acts on the reference input.

Note that models should be entered into Matlab environment before we can import them into the design tool. We can either import the compensator (controller) or enter it using the red cross(poles) or the circle(zero) on the top left-hand corner of the SISO design tool window.
By choosing the tools menu and loop responses, we can display the Step, Impulse, Bode, Nyquist or Nichols plot for the system.