Matlab - Very basic intro for first-time users, i.e. new grad students.

EE5200 - Basics
EE5240 - Advanced.

Matlab is an interpreter, not a compiler.

Matlab has many advanced functions.

- SimPower
- Simulink.

Usage:
- Interactive via line prompt.
- Create and run .m files.
Example code: \[ r2p.m \]
\[ ft.m \]
Will e-mail to ee5200-list@...
The MATLAB *Command* window has numerous features, some of which were introduced in prior chapters.

1 Managing the MATLAB Workspace

The data and variables created in the *Command* window reside in what is called the *MATLAB workspace*. To see what variable names are in the MATLAB workspace, issue the command *who*:

```
EDU> who
Your variables are:
initial_con  min_con
lost         n
```

These are the variables used in the acid–water bath example. For more detailed information, use the command *whos*:

```
EDU> whos
Name      Size         Bytes   Class
initial_con 1x1          8 double array
lost        1x1          8 double array
min_con     1x1          8 double array
n           1x1          8 double array

Grand total is 4 elements using 32 bytes
```

Each variable is listed along with its size, the number of bytes used, and its class. In this particular example, our variables are all scalars having double precision representation. The command *whos* will be more useful later, after the introduction to arrays and other data types.

In addition to these functions, the *Show Workspace* item in the *File* menu creates a GUI window, called the *Workspace Browser*, that contains the same information as the *whos* command. Moreover, it gives you the ability to delete or clear selected variables. This window is also created by pressing the *Workspace Browser* button on the *Command* window toolbar.

As shown earlier, the command *clear* deletes variables from the MATLAB workspace. For example:

```
EDU> clear lost n
EDU> who
Your variables are:
initial_con  min_con
```

deletes the variables *lost* and *n*. Other options for the *clear* function can be found by asking for help:

```
EDU> help clear
```

- *CLEAR* clears variables and functions from memory.
- *CLEAR* removes all variables from the workspace.
- *CLEAR VARIABLES* does the same thing.
- *CLEAR GLOBAL* removes all global variables.
- *CLEAR FUNCTIONS* removes all compiled M-functions.
- *CLEAR MEX* removes all links to MEX-files.
- *CLEAR ALL* removes all variables, globals, functions, and MEX links.

*CLEAR* removes variables specified. The wildcard character `*` can be used to clear variables that match a pattern. For instance, *CLEAR X* clears all the variables in the current workspace that start with *X*. If *X* is global, *CLEAR X* removes *X* from the current workspace, but leaves it accessible to any functions declaring it global. *CLEAR GLOBAL X* completely removes the global variable X.

*CLEAR ALL* also has the side effect of removing all debugging breakpoints since the breakpoints for a file are cleared whenever the m-file changes or is cleared.

Use the functional form of *CLEAR* such as *CLEAR('name')*, when the variable name is stored in a string.

See also *WHO*, *WHOS*.

Clearly, the *clear* function deletes more than just variables. These other features will be demonstrated later.

Finally, when working in the MATLAB workspace, it is often convenient to save or print a copy of your work. The *diary* command saves user input and *Command* window output to an ASCII text file named *diary* in the current directory or folder. *EDU> diary* saves the diary to file *fname*. *EDU> diary off* terminates the diary command and closes the file. When the *Command* window is active, selecting *Print...* from the *File* menu prints a copy of the entire *Command* window. Alternatively, if you highlight a portion of the *Command* window using the mouse, selecting *Print Selection...* from the *File* menu prints the selected text.
Saving and Retrieving Data

In addition to remembering variables, MATLAB can save and load data from files on your computer. The Save Workspace... menu item in the File menu opens a standard file dialog box for saving all current variables. Similarly, the Load Workspace... menu item in the File menu opens a dialog box for loading variables from a previously saved workspace. Saving variables does not delete them from the MATLAB workspace. Loading variables of the same name as those found in the MATLAB workspace changes the variable values to those loaded from the file.

If the File menu approach is not available or does not meet your needs, MATLAB provides two commands—save and load—that offer more flexibility. In particular, the save command allows you to save one or more variables in the file format of your choice. For example:

```
EDU> save
stores all variables in MATLAB binary format in the file matlab.mat.
EDU> save data
saves all variables in MATLAB binary format in the file data.mat.
EDU> save data erasers pads tape
saves the variables erasers, pads, and tape in binary format in the file data.mat.
EDU> save data erasers pads tape -ascii
saves the variables erasers, pads, and tape in 8-digit ASCII format in the file data. ASCII-formatted files may be edited using any common text editor. Note that ASCII files do not get the extension .mat.
EDU> save data erasers pads tape -ascii -double
saves the variables erasers, pads, and tape in 16-digit ASCII format in the file data.
```

The load command uses the same syntax, with the obvious difference of loading variables into the MATLAB workspace.

Number Display Formats

When MATLAB displays numerical results, it follows several rules. By default, if a result is an integer, MATLAB displays it as an integer. Likewise, when a result is a real number, MATLAB displays it with approximately four digits to the right of the decimal point. If the significant digits in the result are outside this range, MATLAB displays the result in scientific notation, similar to scientific calculators. You can override this default behavior by specifying a different numerical format within the Preferences menu item in the File menu, if available, or by typing the appropriate MATLAB command at the prompt. Using the variable `average_cost` from an earlier example, these numerical formats are:

<table>
<thead>
<tr>
<th>Format</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>format short</td>
<td>50.833</td>
<td>5 digits</td>
</tr>
<tr>
<td>format long</td>
<td>50.833333333334</td>
<td>16 digits plus exponent</td>
</tr>
<tr>
<td>format short e</td>
<td>5.0833e+01</td>
<td>5 digits plus exponent</td>
</tr>
<tr>
<td>format long e</td>
<td>5.083333333334e+01</td>
<td>16 digits plus exponent</td>
</tr>
<tr>
<td>format short g</td>
<td>50.833</td>
<td>better of format short or format short e</td>
</tr>
<tr>
<td>format long g</td>
<td>50.833333333333</td>
<td>better of format long or format long e</td>
</tr>
<tr>
<td>format hex</td>
<td>40496aaaaaaaaaa</td>
<td>hexadecimal</td>
</tr>
<tr>
<td>format bank</td>
<td>50.83</td>
<td>2 decimal digits</td>
</tr>
<tr>
<td>format +</td>
<td>+</td>
<td>positive, negative, or zero</td>
</tr>
<tr>
<td>format rat</td>
<td>305/6</td>
<td>rational approximation</td>
</tr>
</tbody>
</table>

It is important to note that MATLAB does not change the internal representation of a number when different display formats are chosen; only the display changes.