Names

SUMMARY FİLE

Exercise 1:

The purpose of this exercise is illustrate editing a downloaded file.

1.1 I started up Matlab and used the "Current Directory" window to

create the directory "math2070" and then the directory "lab01"

inside "math2070." Then I typed in the commands

diary diary.txt

% Lab 1 summary. M. M. Sussman

the previous line is just a comment that will show up in the diary.txt file.

1.2 From the browser, I right-clicked on the file demoscript.m,

and followed directions to download a file to "math2070/lab01".

This file appeared in the "Current Directory" file listing and

also when I did a dir command and got

. .. demoscript.m diary.txt exercise1

1.3 I was able to edit the file with the "edit demoscript.m" command.

1.4 I executed the file from the Tools-->Run menu choice. The

result was (after eliminating blank lines):

x =

1.3000

xsquared =

1.6900

p =

1.8540

ans =

3.1416

adjacent =

0.7071

z =

3.0000 + 2.0000i

zsquared =

5.0000 +12.0000i

logz =

1.2825 + 0.5880i

ans =

The following is a list of cube roots.

cuberoots =

Columns 1 through 7

1.0000 1.2599 1.4422 1.5874 1.7100 1.8171 1.9129

Columns 8 through 10

2.0000 2.0801 2.1544

Exercise 2. The point of this exercise is to demonstrate the

debugging trace utility.

2.1 I found the menu picks: Debug-->"Stop if Error" and

Debug-->"Stop if NaN or Inf" and activated them.

2.2 I found the statement bad=1/(x-1); near the bottom of the

file, took away the percent character, and saved the file.

2.3,2.4 I re-ran the script. I saw the same output as previously,

but it ended with

Warning: Divide by zero.

> In /afs/pitt.edu/home/m/m/mms125/lab01/demoscript.m at line 54

bad =

Inf

and a blue arrow appeared next to the bad= statement in the

edit window. The special debugging prompt "K>>" appeared in

the command window.

2.5 I hovered the mouse pointer over "x" and saw its value = 1.

2.7 I exited debugging mode with the Debug-->"Quit Debugging" menu

choice.

Exercise 3. The point of this exercise is to illustrate how to

trace execution using the debugger.

3.1 I typed "clear".

3.2 I set a breakpoint on the first executable line. A red dot

appeared there.

3.3 I selected "Run" from the debug menu. An arrow appeared, pointing

to the line with the red dot.

3.4,5 I used the "Step" button (One button in the last

group of buttons at the top of the "Edit" window. It has an "L"-shaped

arrow on it and an image of a single page.) to step through the

file. As new variables were defined, they appeared in the

"Current Workspace" windowpane, and results appeared in the

"Command" windowpane.

4.6 The loop to compute cube roots was tedious, so I put another

breakpoint just after the loop and used the "Continue" button

(I could have used a menu pick) to jump directly to the new

breakpoint.

4.7 I used the "Exit Debug Mode" button.

THE FOLLOWING IS A SUMMARY OF AN EXERCISE SIMILAR TO ONES THAT MIGHT

APPEAR LATER IN THIS COURSE.

5. The point of this exercise is to illustrate using the Matlab

"polyfit" command to construction a polynomial curve approximately

through given data points.

5.1 I downloaded the data points from the web page and read them

into Matlab. The result was two vectors, xdata and ydata, so

that each data point is represented as (xdata(i),ydata(i)) for

i between 1 and 10.

5.2 I used the command "c=polyfit(xdata,ydata,3)" to find the coefficients

of a parabola that passes near the given data points. These coefficients

are given by c=[1.1 3.2 -5 12].

5.3 I constructed a set of abscissae for plotting. Since the data points

lie between x=0 and x=1, I chose 100 points between x=-.5 and x=+1.5

with the command "xplot=linspace(-.5,1.5,100);". 100 points will make a