Diagnosing and Dealing with Errors in MATLAB

Mike Hansen

6-4-2011

A fundamental law of programming is that all programmers make stupid mistakes (good programmers just make fewer ones). Another law is that stupid mistakes often have easy fixes but disastrous-looking results. A lot of the errors being dealt with in the help sessions can fit into this category of mistakes. Diagnosing and fixing such errors is not only an important skill for you to learn (now!), it will also allow us to address more important topics in the help sessions.

When you encounter an error in MATLAB, and your program terminates with an ugly red error message, a simple procedure for diagnosing and fixing it is as follows:

(1) Check to see if the error message is old. If you don't clear the command window when executing your program, your error messages may be old ones that don't still apply. There's nothing more frustrating than trying to fix an error which doesn't exist (and screwing up good code in the process!).

(2) If your error message is current, then read it carefully. It will give you a basic description of the error as well as the code line and m-file name where your program terminates. If you can't figure out the solution from the error message and a review at your code, move on to step 3.

(3) If your error concerns a MATLAB function like *plot* or *linspace* then you can use MATLAB's help directory. If that doesn't work, try a search engine or MATLAB's website.

(4) If you can't resolve the error with the error message, the help directory, or the internet, then send one of us an email and/or come to the help session(s).

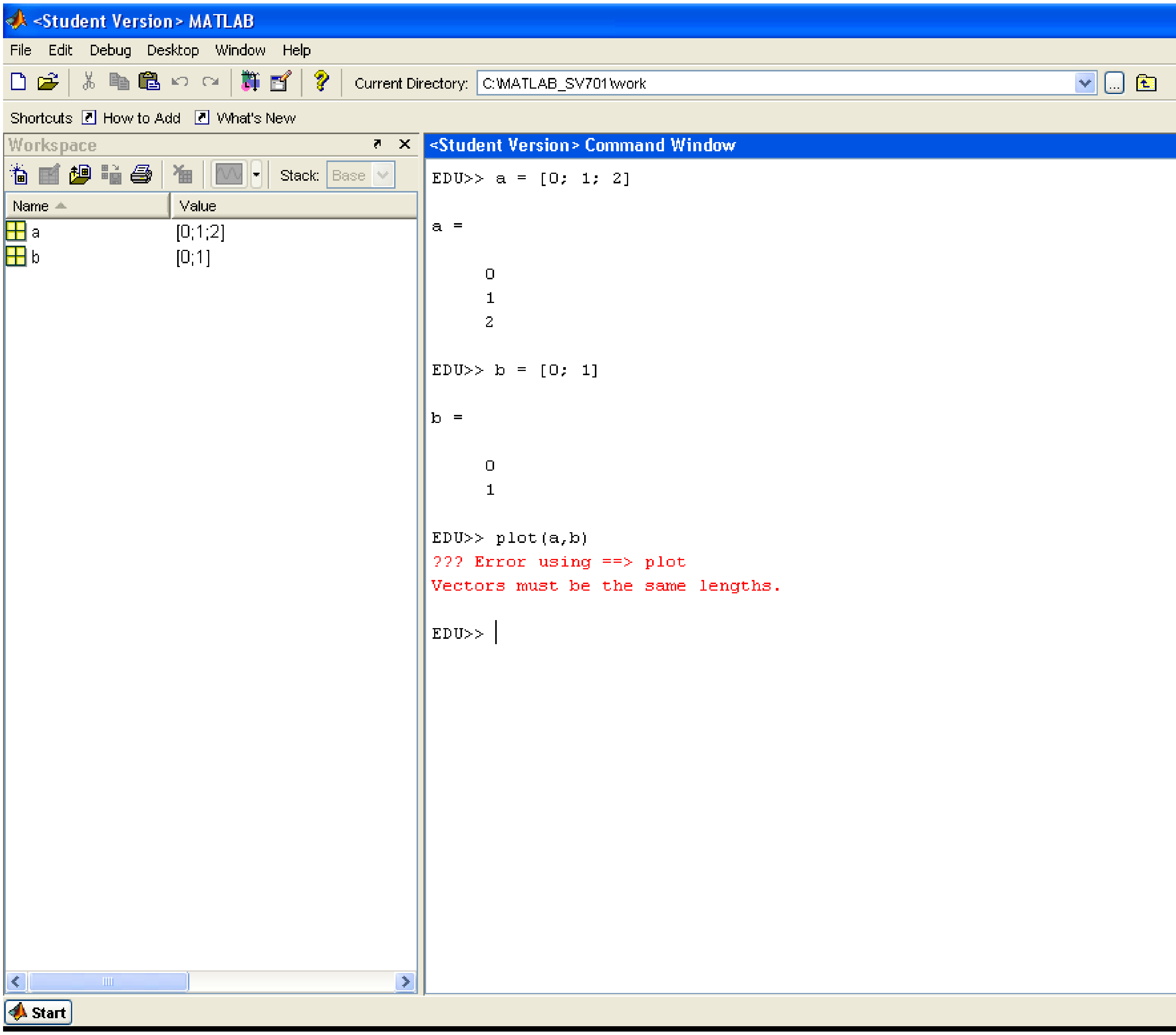
Below are three examples of diagnosing an error by using the error message and MATLAB's help directory. The third example in particular should be helpful.

EXAMPLES:

1. An error with plot.
2. An error with matrix operations.
3. The larger error message associated with a problematic function.

An error with the *plot* command.

In the command window below, I've defined vectors **a** and **b**. Then I have tried to plot **b** over **a**. This gives me an error message:



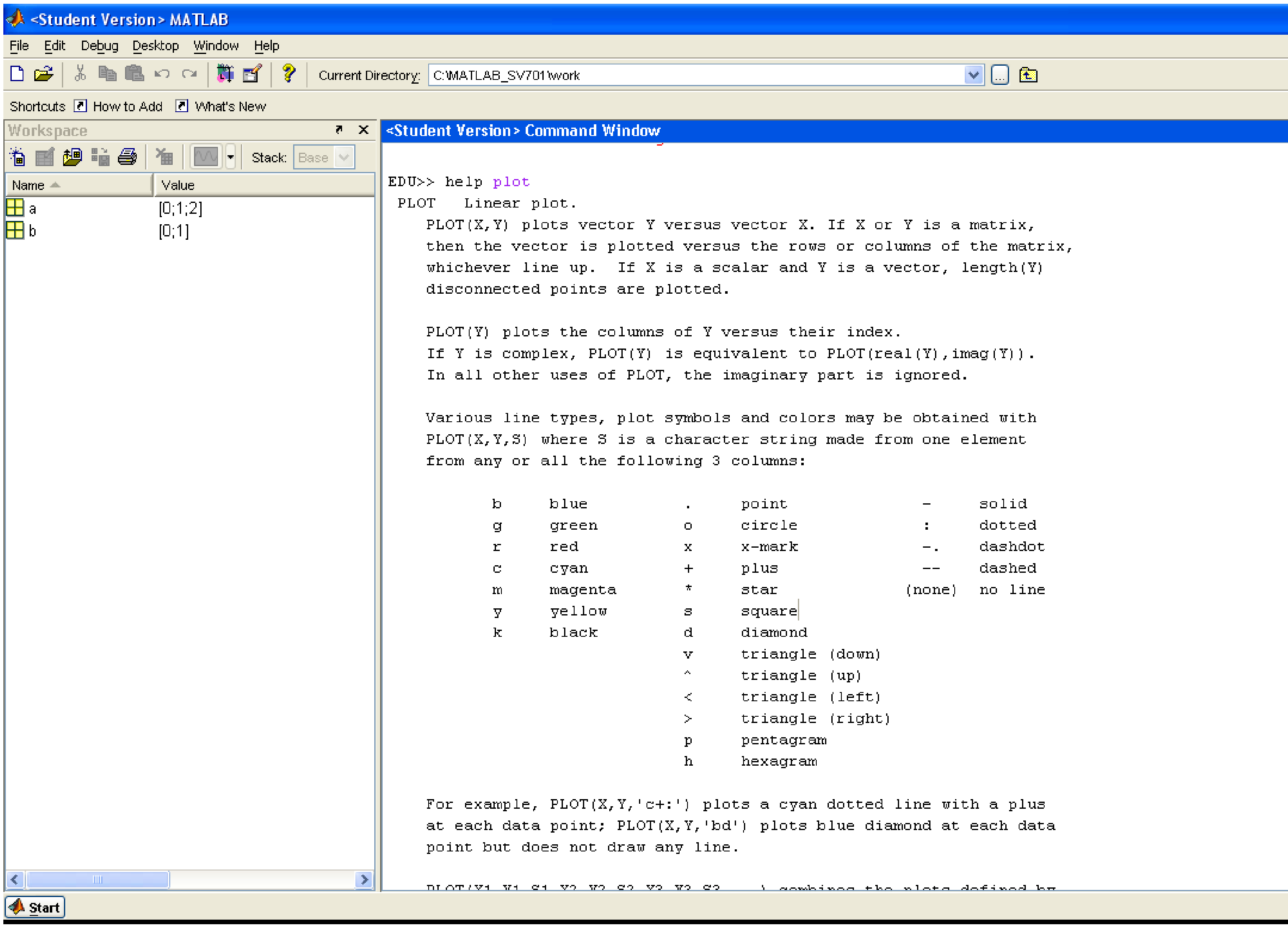
<Student Version> MATLAB
File Edit Debug Desktop Window Help
Current Directory: C:\MATLAB_SV701\work
Shortcuts How to Add What's New
Workspace
Stack: Base
<Student Version> Command Window
Start

Name	Value
a	[0;1;2]
b	[0;1]

```
EDU>> a = [0; 1; 2]

a =

     0
     1
     2

EDU>> b = [0; 1]

b =

     0
     1

EDU>> plot(a,b)
??? Error using ==> plot
Vectors must be the same lengths.

EDU>> 
```

Reading the error message, I see that the vectors I'm plotting are not the same length. Looking at **a** and **b**, we see that this is the case.

If I didn't immediately understand what the error message meant, I could try MATLAB's help directory. Because we're having an issue with plot, we type *help plot*, and get the following result:


<Student Version> MATLAB
File Edit Debug Desktop Window Help
Current Directory: C:\MATLAB_SV701\work
Shortcuts How to Add What's New
Workspace
Stack: Base
<Student Version> Command Window
Start

Name	Value
a	[0;1;2]
b	[0;1]

```
EDU>> help plot
 PLOT   Linear plot.
    PLOT(X,Y) plots vector Y versus vector X. If X or Y is a matrix,
    then the vector is plotted versus the rows or columns of the matrix,
    whichever line up.  If X is a scalar and Y is a vector, length(Y)
    disconnected points are plotted.

    PLOT(Y) plots the columns of Y versus their index.
    If Y is complex, PLOT(Y) is equivalent to PLOT(real(Y),imag(Y)).
    In all other uses of PLOT, the imaginary part is ignored.

    Various line types, plot symbols and colors may be obtained with
    PLOT(X,Y,S) where S is a character string made from one element
    from any or all the following 3 columns:

           b     blue          .     point              -     solid
           g     green         o     circle             :     dotted
           r     red           x     x-mark             -.    dashdot
           c     cyan          +     plus               --    dashed
           m     magenta       *     star             (none)  no line
           y     yellow        s     square
           k     black         d     diamond
                               v     triangle (down)
                               ^     triangle (up)
                               <     triangle (left)
                               >     triangle (right)
                               p     pentagram
                               h     hexagram

    For example, PLOT(X,Y,'c+:') plots a cyan dotted line with a plus
    at each data point; PLOT(X,Y,'bd') plots blue diamond at each data
    point but does not draw any line.

    PLOT(X1,Y1,S1,X2,Y2,S2,X3,Y3,S3,...) combines the plots defined by
```

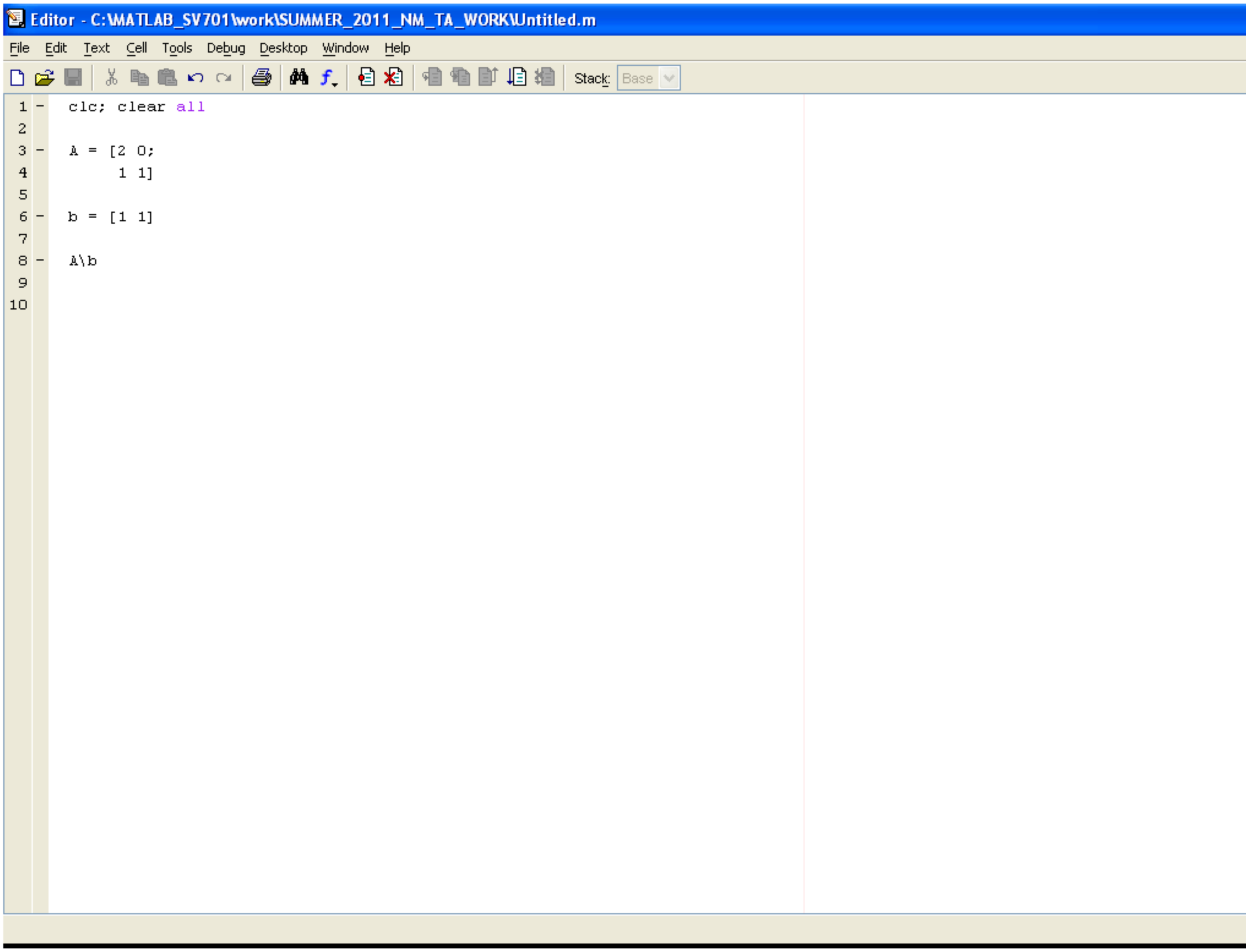
Even if you haven’t received any plotting errors so far, the text above is still a good read. Nobody can memorize all the customizations to the *plot* command, so “*help plot*” is a good one to remember.

An error with a matrix operation.

A lot of you have had at least one issue with MATLAB when dealing with rows and columns, and how they differ in matrix operations.

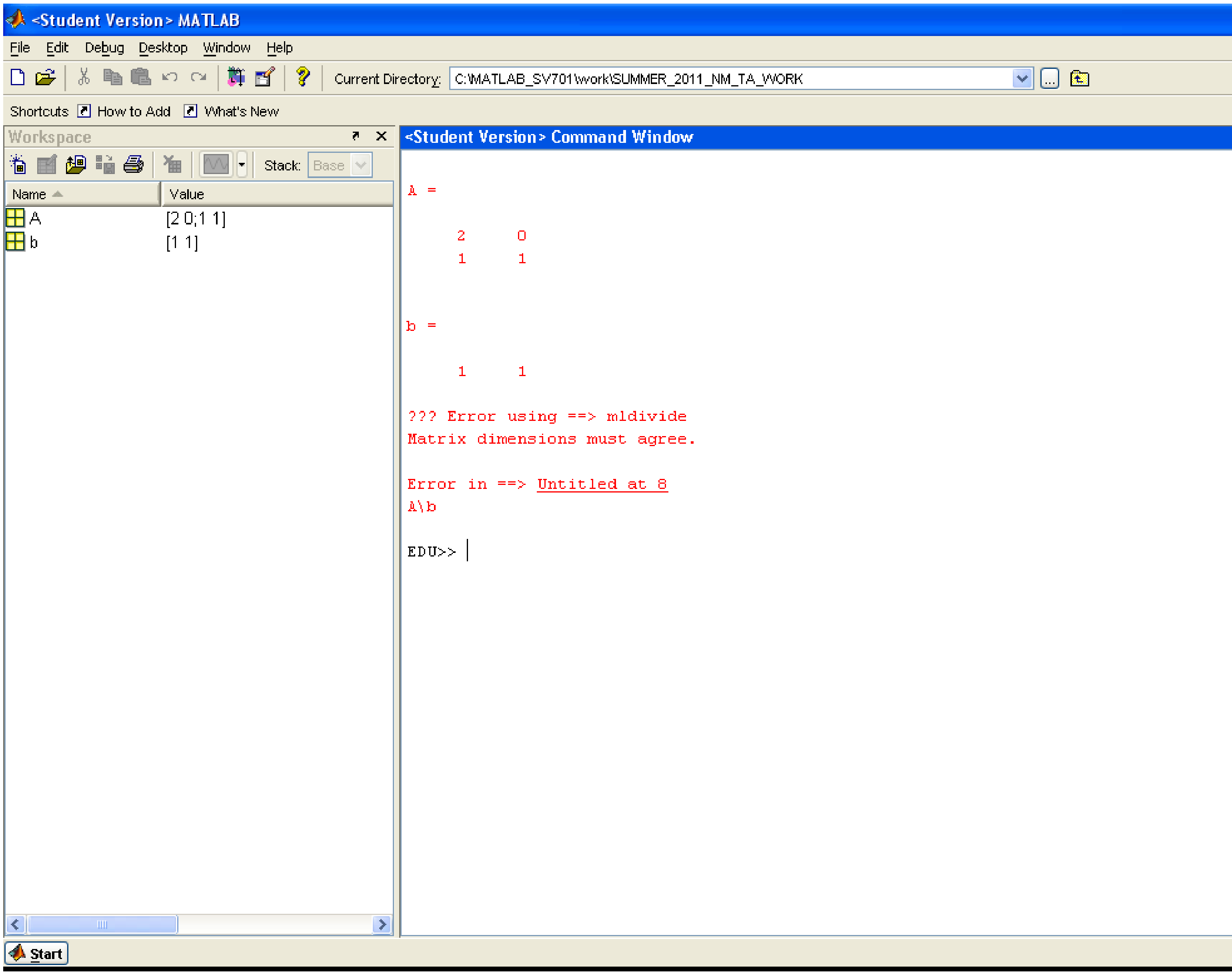
First of all, remember the rule for matrix multiplication: **A*B** only is possible if the number of columns of **A** equals the number of rows of **B**. This is why we can't multiply an NxN matrix by a 1xN row vector.

Looking at the m-file below, we see a 2x2 matrix **A** and a 1x2 vector **b**. Suppose we have a system of equations represented by **Ax** = **b**. In attempting to solve this equation for **x**, we use the backslash command:


Editor - C:\MATLAB_SV701\work\SUMMER_2011_NM_TA_WORK\Untitled.m
File Edit Text Cell Tools Debug Desktop Window Help
Stack: Base

```
clc; clear all

A = [2 0;
     1 1]

b = [1 1]

A\b
```

When we execute this code, we should expect an error because we've defined **b** as a row vector, and not as a column. If **b** is a row vector then a matrix multiplication of **A** and **x** cannot possibly result in **b**, which means that there cannot be a solution. MATLAB's backslash command sees this and gives us an error message:



<Student Version> MATLAB
File Edit Debug Desktop Window Help
Current Directory: C:\MATLAB_SV701\work\SUMMER_2011_NM_TA_WORK
Shortcuts How to Add What's New
Workspace
Stack: Base
<Student Version> Command Window
Start

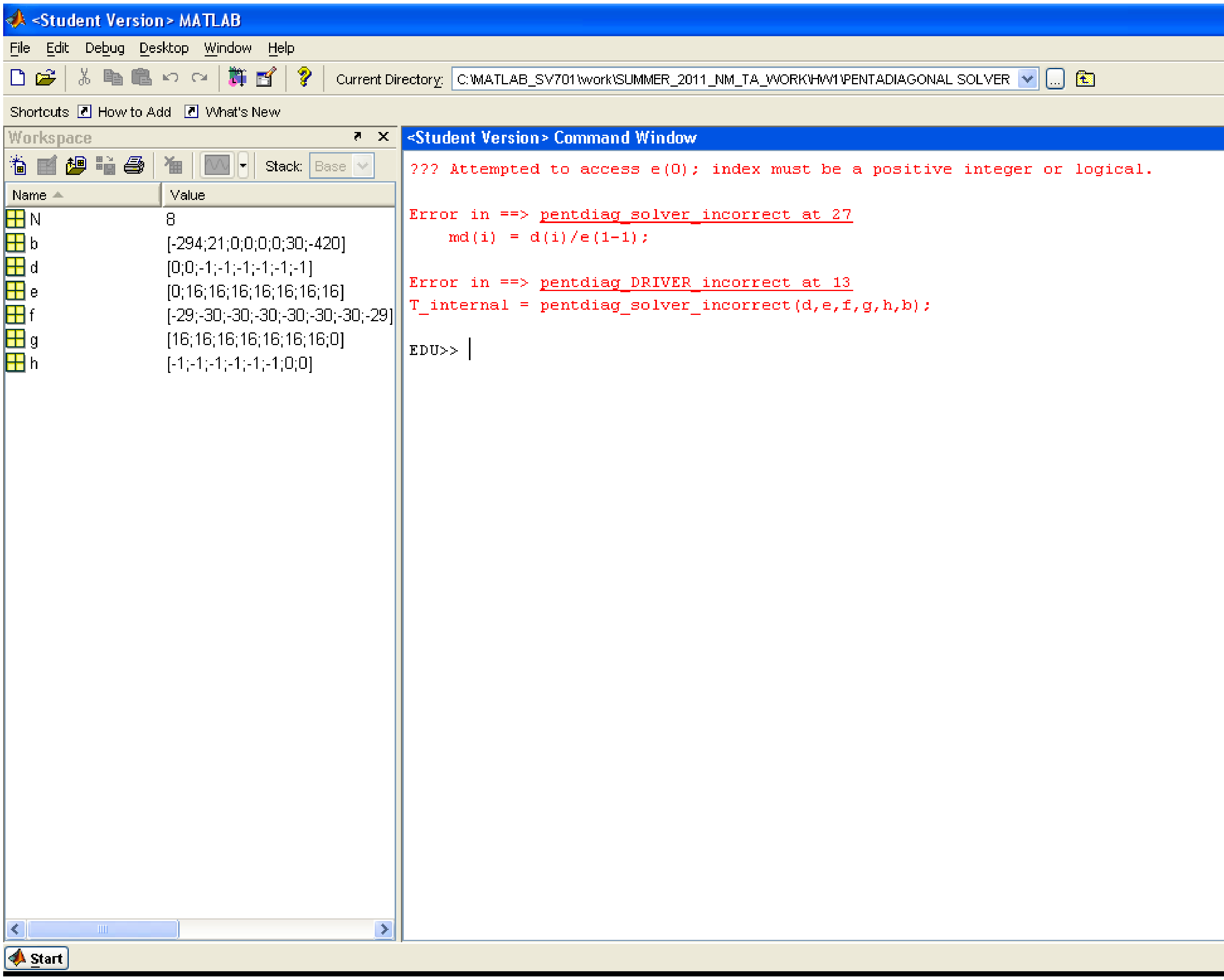
Name	Value
A	[2 0;1 1]
b	[1 1]

```
A =

     2     0
     1     1


b =

     1     1

??? Error using ==> mldivide
Matrix dimensions must agree.

Error in ==> Untitled at 8
A\b

EDU>>
```

Now that we have the red error message, we read it and find that the error concerns matrix dimensions, and it occurs on line 8 of Untitled (the name of the m-file), where we tried to do $A\backslash b$. When you see an issue about matrix dimensions, most of the time it's either a size issue or a row vs. column issue. We solve the problem by making **b** a column vector.

An error in a function.

The examples above give error messages that aren't too ugly. But when dealing with functions we'll often get something uglier like this:



<Student Version> MATLAB
File Edit Debug Desktop Window Help
Current Directory: C:\MATLAB_SV701\work\SUMMER_2011_NM_TA_WORK\HW1\PENTADIAGONAL SOLVER
Shortcuts How to Add What's New
Workspace
Stack: Base
<Student Version> Command Window
Start

Name	Value
N	8
b	[-294;21;0;0;0;0;30;-420]
d	[0;0;-1;-1;-1;-1;-1;-1]
e	[0;16;16;16;16;16;16;16]
f	[-29;-30;-30;-30;-30;-30;-30;-29]
g	[16;16;16;16;16;16;16;0]
h	[-1;-1;-1;-1;-1;-1;0;0]

```
??? Attempted to access e(0); index must be a positive integer or logical.

Error in ==> pentdiag_solver_incorrect at 27
    md(i) = d(i)/e(i-1);

Error in ==> pentdiag_DRIVER_incorrect at 13
T_internal = pentdiag_solver_incorrect(d,e,f,g,h,b);

EDU>>
```

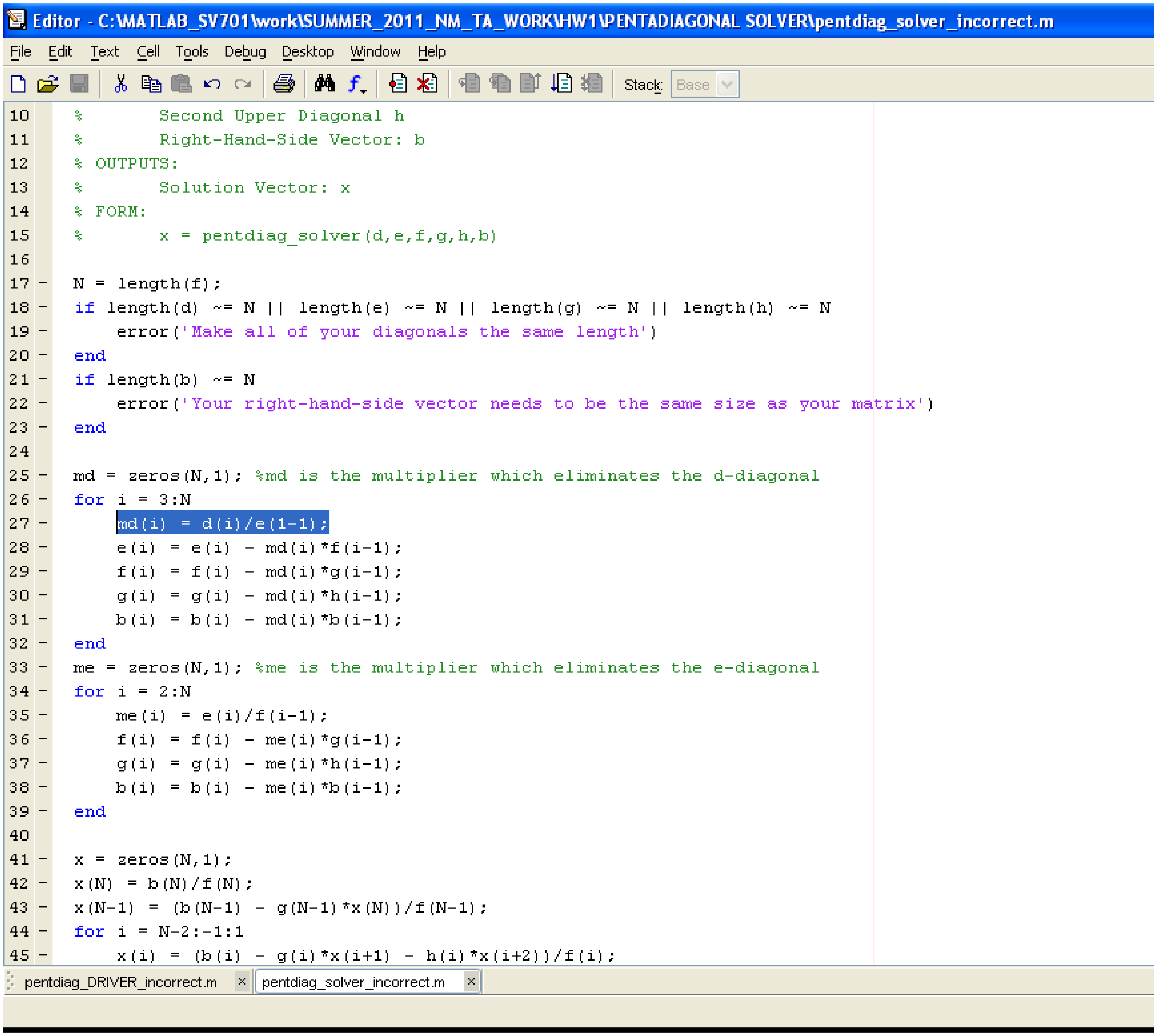
This error message has not one but two errors. The code that gave this result is an incorrect version (not a student's, just a modification of my own) of the pentadiagonal matrix solver from the first homework assignment. (The incorrect function is called pentdiag_solver_incorrect and the driver m-file is called pentdiag_DRIVER_incorrect.)

We can see from the error message that there is a problem with BOTH the driver and the function. But, upon closer look at line 13 in the driver (or in the error message, which conveniently shows us line 13), we see that the error in the DRIVER is the call to the function, which means that the function is incorrect. **Look carefully at the error message above: this is the way MATLAB will show you an error that results from calling a faulty function. It prints the error in the driver (which calls the function) on the bottom, and the error in the function above it.**

Reading the function's error message, we see that the error is on line 27, and that MATLAB can't access e(0), because "index must be positive integer or logical." This probably sounds a little cryptic, but let's look at what's really going on:

Looking at our workspace (to the left of my command window) above, we see that e is a vector. We know that we can't access e(0) because the first element of any vector is indexed with 1. Now that we have an idea about how/why our program is screwing up, we need to check out the incorrect code:

The incorrect function, with line 27 highlighted, is shown below:


Editor - C:\MATLAB_SV701\work\SUMMER_2011_NM_TA_WORK\HW1\PENTADIAGONAL SOLVER\pentdiag_solver_incorrect.m
File Edit Text Cell Tools Debug Desktop Window Help
Stack: Base
pentdiag_DRIVER_incorrect.m
pentdiag_solver_incorrect.m

```
10      %       Second Upper Diagonal h
11      %       Right-Hand-Side Vector: b
12      % OUTPUTS:
13      %       Solution Vector: x
14      % FORM:
15      %       x = pentdiag_solver(d,e,f,g,h,b)
16
17 -    N = length(f);
18 -    if length(d) ~= N || length(e) ~= N || length(g) ~= N || length(h) ~= N
19 -        error('Make all of your diagonals the same length')
20 -    end
21 -    if length(b) ~= N
22 -        error('Your right-hand-side vector needs to be the same size as your matrix')
23 -    end
24
25 -    md = zeros(N,1); %md is the multiplier which eliminates the d-diagonal
26 -    for i = 3:N
27 -        md(i)  = d(i)/e(1-1);
28 -        e(i)  = e(i) - md(i)*f(i-1);
29 -        f(i)  = f(i) - md(i)*g(i-1);
30 -        g(i)  = g(i) - md(i)*h(i-1);
31 -        b(i)  = b(i) - md(i)*b(i-1);
32 -    end
33 -    me = zeros(N,1); %me is the multiplier which eliminates the e-diagonal
34 -    for i = 2:N
35 -        me(i)  = e(i)/f(i-1);
36 -        f(i)  = f(i) - me(i)*g(i-1);
37 -        g(i)  = g(i) - me(i)*h(i-1);
38 -        b(i)  = b(i) - me(i)*b(i-1);
39 -    end
40
41 -    x = zeros(N,1);
42 -    x(N)  = b(N)/f(N);
43 -    x(N-1)  = (b(N-1) - g(N-1)*x(N))/f(N-1);
44 -    for i = N-2:-1:1
45 -        x(i)  = (b(i) - g(i)*x(i+1) - h(i)*x(i+2))/f(i);
```

We can see the problem immediately. A simple typo of “1-1” instead of the correct “i-1”, which results in trying to access e(0).

This is an easy mistake to make, and if you couldn’t effectively read the error message it generated, your progress would be stalled.

Whenever you receive an error message, work with the procedure stated on the first page. Try to figure it out for yourself. Often you’ll have to learn some new aspects of MATLAB, but there’s a very good chance that the new information you gain will help you later in the course.

And when you come to the help sessions, it’s a lot easier for us to help you fix an error if you can tell us what you’ve already tried.