

Using GNU Octave for *Numerical Methods*

by Dennis Pence*

September 2015

Contents

1	What are the alternatives to Matlab?	1
1.1	What is GNU Octave?	1
1.2	What is Scilab?	2
1.3	Other Alternatives?	3
2	Basic Operations with GNU Octave	5
2.1	Launching GNU Octave	5
2.2	Vectors	6
2.3	Getting Help	10
2.4	Matrices	13
2.5	Creating and Running Files	14
2.6	Comments	15
2.7	Plotting	16
2.8	Creating Your Own Functions	20
2.9	Printing	21
2.10	More Loops and Conditionals	22
2.11	Clearing Variables	22
2.12	Logging Your Session	23
2.13	More Advanced Commands	23

*If you have comments on or corrections to this documentation, please send them to dennis.pence@wmich.edu

3	Monte Carlo Methods	25
4	Solution of a Single Nonlinear Equation in One Unknown	26
4.1	Bisection	26

1 What are the alternatives to Matlab?

The textbook we now use here at Western Michigan University, *Numerical Methods: Design, Analysis, and Computer Implementation of Algorithms*, by Anne Greenbaum & Timothy P. Chartier, mostly assumes that you have access to MATLAB. It frequently gives actual MATLAB code for algorithms and sometimes asks in the exercises for you to explore something specific to MATLAB. Since we have a license for MATLAB in the Rood Hall Computer Lab and the Parkview Campus Computer Lab, you can have such access. However many students find it more convenient to work on their own personal computers. The student edition of MATLAB costs about \$100, and many of our graduate students in the Applied and Computational Mathematics Program find it a very reasonable expense, given that they can use it in many later courses. However if you plan to only take one course, this might be more than you want to pay. Thus this document explores alternatives, particularly free alternatives!

1.1 What is GNU Octave?

Octave was originally conceived by John W. Eaton as a numerical supplement for a chemical engineering textbook. Instead, it has become a more general numerical mathematical application that is now distributed by GNU Free Software Foundation. To avoid confusion with the musical term “octave”, it is best to always search for the term “GNU Octave”, and most people now use this full name. Octave is the other most popular open-source alternative to the commercial product MATLAB. Much more than Scilab (described next), Octave strives to be a virtual clone of MATLAB. The syntax is almost identical, and this is possible because of the 1996 U. S. Supreme Court Case, *Lotus v. Borland*, which said that software copyrights do not extend to the text or menu layout (hence the syntax) of a program.

While the developers of GNU Octave attempt to have this program do the same operations as will be done by the same Matlab commands, there is no guarantee that the code underneath used to accomplish these operations will be the same. There may be situations where both developers use some of the same open-source code, but we cannot know when this happens. I recently read an article somewhere about the computer programming language “Julia”, and I went to the website <http://julialang.org/> to learn more. The claim for this new computer programming language is that it is as easy to code as high-level languages such as Python or as mathematical application

packages such as Matlab or GNU Octave, but that it compiles to run nearly as fast as low-level languages such as C. The Home page of this website had a nice comparison of how long it took for certain computationally intensive tasks on various platforms, and Matlab and Octave are in the table. Of course, “Julia” comes out very well in this comparison. My surprise was how poorly Octave performed in this table compared to Matlab. Thus we will probably need to consider GNU Octave as mostly an educational tool for reasonably small problems. I will note below when I notice the very slow performance of Octave compared to Scilab (which, unfortunately was not included in the comparison table). The article I was reading predicted that more and more numerical linear algebra tasks will be coded in Julia in the future.

You can download a free copy of the latest version of GNU Octave at <https://gnu.org/software/octave/>, with the product available on many platforms. Beginning with version 4.0.0 (May 29, 2015), the default interface is a GUI, which makes it work on the various platforms in a much more user-friendly manner (as Scilab and MATLAB have done for some time). The older more “command line” interface is still available, but you will probably not want to use it. With the download, you will get documentation in many formats. I generally open the PDF version (currently 966 pages long) to view whenever I am working with Octave. Chapter 1 (A Brief Introduction to Octave) and Chapter 2 (Getting Started) of the Octave Documentation constitute an very nice tutorial.

This document will attempt to point out significant differences needed for GNU Octave code compared to the MATLAB code of the text *Numerical Methods: Design, Analysis, and Computer Implementation of Algorithms*, by Anne Greenbaum & Timothy P. Chartier.

1.2 What is Scilab?

Scilab is a numerical mathematical application originally created by INRIA and École nationale des ponts et chaussées (ENPC) in France. It is one of the most popular open-source alternatives to the commercial product MATLAB. Scilab is now supported by a company called Scilab Enterprises that also sells services to help other companies conduct their business computations using Scilab and related products.

You can download a free copy of the latest version of Scilab at <http://www.scilab.org/>, with the product available on many platforms. The download can even take advantage of the processors available on your computer to do

faster matrix and vector computations. The “documentation” that comes with the download is unfortunately little more than a list of commands in Scilab. There is also a “help” file giving the conversion of many common MATLAB commands into slightly different Scilab commands. Thus it is recommended that you begin with some Scilab tutorials such as those found at <http://www.scilab.org/resources/documentation/tutorials>, particularly the ones titled *Scilab for very beginners* and *Introduction to Scilab*. The internal “help” files will be more useful after you get some experience working through the tutorials.

There will be a similar document that will attempt to point out significant differences needed for Scilab code compared to the MATLAB code of the text *Numerical Methods: Design, Analysis, and Computer Implementation of Algorithms*, by Anne Greenbaum & Timothy P. Chartier.

1.3 Other Alternatives?

While I do not recommend that you use anything else, it is certainly possible to use other mathematical applications for the project tasks of the course, particularly if you are already very familiar with one of these products. I will make no attempt to include code in this document for all of these possible alternatives, but I will list some of them here.

Since we are a Maple campus, we have the computer algebra system (CAS) Maple in most of the open labs, and we use Maple in many courses, you may already have experience using Maple (and may have purchased a student edition for your personal computer). The “trick” for doing numerical projects in Maple is to make sure that you are forcing floating-point computations rather than the exact symbolic computations that a CAS will attempt by default. There are also special commands to direct Maple to skip a symbolic calculation and go directly to a numerical approximation, particularly with vector and matrix computations. When it switches to floating-point numerical work, Maple now does very well by using highly-regarded NAG algorithms rather than its own algorithms designed for symbolic work. Since some of the methods we study are also mentioned in typical calculus courses (like Newton’s method), you may find some of the things we study in the “clickable” categories of Maple.

It is certainly possible to do many of the things we study on a programmable graphing calculator. It would be best to use something more than a high-school level tool (TI-84 Plus) to get more programming (including the passing

of parameters to a program). I often use a TI-89 or Voyage 200 to demonstrate small things. The newer TI-nspire has more limited programming (unless you do Lua programming on a computer and move it to the handheld), and since we do not really need CAS, either version of the TI-nspire will work. Unfortunately it is more difficult to get your work into a report generated in some computer word processing program from the calculators. Thus you might find it appropriate to use a graphing calculator for some small homework tasks, and you might include what happened using a calculator in a report primarily done using one of the more complete tools.

If you are already familiar with the statistical computing application R, that can also do most of what we need to do. This is also free, it is nicely available if you want to try it. It can be downloaded at <https://www.r-project.org/>. Since R is designed to do advanced things in statistics, most people don't think to use it for other things. But many statistical computations reduce to matrix computations, and so R has very good matrix computations built in.

It is not unusual for Wikipedia to have excellent information on computer science and computational mathematics topics. Thus you can find a brief description of everything above (Scilab, GNU Octave, Maple, R) in that resource. Also very nice is the Wikipedia page giving a "Comparison of numerical analysis software" https://en.wikipedia.org/wiki/Comparison_of_numerical_analysis_software. Certainly many of the products in this extensive list can do the tasks we desire. Our textbook mentions Sage in the Preface as a possible alternative. Sage (included in the Wikipedia comparison above) is a massive open-source project that essentially uses almost anything else open-source that might be helpful. In particular, R is included in Sage, and the numerical packages SciPy and NumPy for the Python programming language are included into Sage for numerical work as well as many other sources. I have even had one student program directly in Python (an open-source programming language) frequently using things in the SciPy and NumPy packages. Since Sage is mostly programmed in Python, these relationships are easily understood.

While it is not unreasonable for you to briefly explore some of these alternatives, it is best for a student in a numerical analysis course to quickly settle on learning one tool to use for the semester. No matter which tool you select, you will get more and more comfortable with the tool as you use it repeatedly. It is best to pick something either you already know or something you know others in the class will be using. Thus I repeat again that I recommend Scilab and GNU Octave as the best alternatives to the commercial product MATLAB. Of course if you are willing to spend the money for it, MATLAB

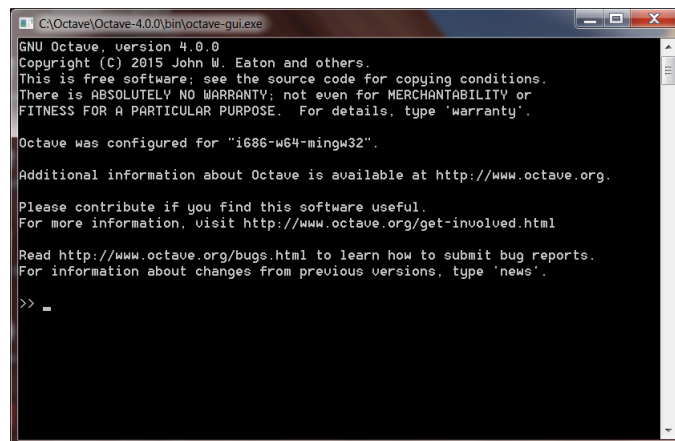
is certainly the standard product for research in numerical analysis.

2 Basic Operations with GNU Octave

I will try in each section to give Octave code that corresponds to the MATLAB code of that chapter in the textbook. Since I am most familiar with using this product in Windows, the work will mostly follow that platform. But this application operates in a similar manor on the Macintosh or Linux computer platform.

2.1 Launching GNU Octave

When you click on the Octave (CLI) icon, you get a screen that looks like this.



```
C:\Octave\Octave-4.0.0\bin\octave-gui.exe
GNU Octave, version 4.0.0
Copyright (C) 2015 John W. Eaton and others.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE. For details, type 'warranty'.

Octave was configured for "i686-w64-mingw32".

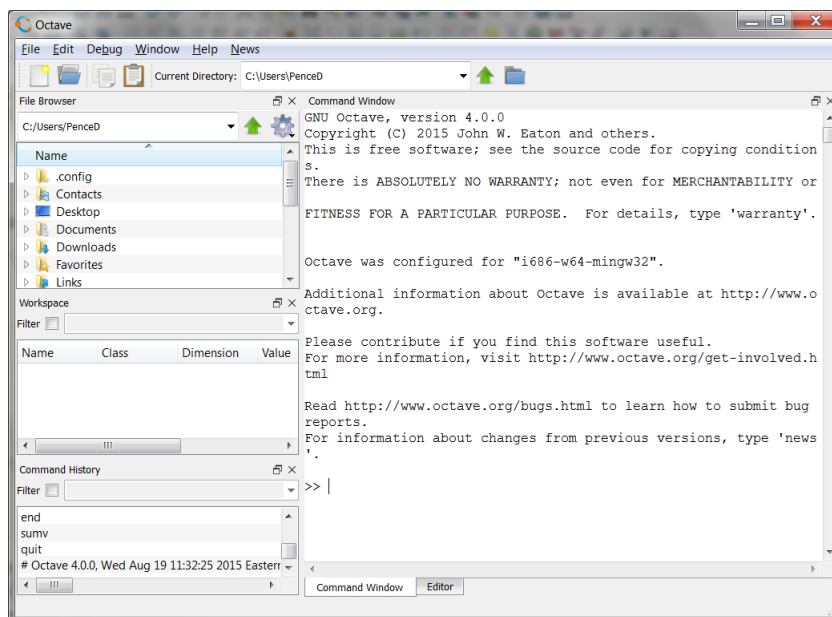
Additional information about Octave is available at http://www.octave.org.

Please contribute if you find this software useful.
For more information, visit http://www.octave.org/get-involved.html

Read http://www.octave.org/bugs.html to learn how to submit bug reports.
For information about changes from previous versions, type 'news'.

>> _
```

This old-fashioned command line interface (CLI) is somewhat crude to use. It is even hard to “copy-and-paste” out of it and into it. Much nicer now is the Octave (GUI) version.



You perform your immediate work in the command window, typing where there is the prompt `>>`. You can use Octave like a calculator. For instance, if you type at the prompt

```
>> 1+2*3
```

then Octave returns with the answer

```
ans = 7
```

Since you did not give a name to your result, Octave stores the result in a variable called `ans`. You can do further arithmetic using the result in `ans`.

```
>> ans/4
```

and Octave will return with the result

```
ans = 1.7500
```

2.2 Vectors

Octave can store row and column vectors. The commands


```

>> v = [1;2;3;4]
v =
    1
    2
    3
    4
>> w = [5, 6, 7, 8]
w =
    5    6    7    8

```

create a column vector \mathbf{v} of length 4 and a row vector \mathbf{w} of length 4. In general, when defining a matrix or vector, semicolons are used to separate rows, while commas or spaces are used to separate entries within a row. Space nearly anywhere have no effect, and so you can use them to make the work more readable. You can refer to an entry in a vector by giving its index.

```

>> v(2)
ans = 2
>> w(3)
ans = 7

```

Octave can add two vectors of the exact same dimension, but when you add two vectors with different dimensions, it now does something called “broadcasting” with the following result.

```

>> v+w
ans =
    6    7    8    9
    7    8    9   10
    8    9   10   11
    9   10   11   12

```

Here is a section from the Octave Documentation describing this feature (which was new with version 3.6.0, and apparently is also in Matlab now).

Broadly speaking, smaller arrays are “broadcast” across the larger one, until they have a compatible shape. The rule is that corresponding array dimensions must either

1. be equal, or
2. one of them must be 1.

In case all dimensions are equal, no broadcasting occurs and ordinary element-by-element arithmetic takes place. For arrays of higher dimensions, if the number of dimensions isn't the same, then missing trailing dimensions are treated as 1. When one of the dimensions is 1, the array with that singleton dimension gets copied along that dimension until it matches the dimension of the other array.

Thus in our example above, since v had dimension 4×1 and w had dimension 1×4 , each needed to be “expanded” so that the addition can take place. What we got above was the following.

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 \end{bmatrix} + \begin{bmatrix} 5 & 6 & 7 & 8 \\ 5 & 6 & 7 & 8 \\ 5 & 6 & 7 & 8 \\ 5 & 6 & 7 & 8 \end{bmatrix}$$

I have no idea why someone would desire this new feature. There is a complicated way to instead turn on a warning when this happens, but, apparently, when you do it also warns about traditional multiplication by a scalar. Here is one more example of “broadcasting.”

```
>> w + 2
ans =
    7    8    9   10
```

The transpose of w is denoted w' :

```
>> w'
ans =v
    5
    6
    7
```

You can add v and w' using ordinary vector addition (i.e. no broadcasting is needed):

```
>> v+w'
ans =
    6
    8
   10
   12
```

Suppose you wish to compute the sum of entries in `v`. One way to do this is as follows:

```
>> v(1)+v(2)+v(3)+v(4)
ans = 10
```

Another way is to use a `for` loop:

```
>> sumv = 0;
>> for i=1:4, sumv = sumv + v(i); end
>> sumv
sumv = 10
```

This code initializes the variable `sumv` to 0. It then loops through each value `i = 1, 2, 3, 4` and replaces the current value of `sumv` with that value plus `v(i)`. The line with the `for` statement actually contains three separate Scilab commands. It could have been written in the following form. (Notice how you do not get another “prompt” after you begin a `for` structure until you complete the structure with an `end`.)

```
>> sumv=0;
>> for i = 1:4
    sumv = sumv + v(i);
end
>> sumv
sumv = 10
```

Octave follows all of the conventions mentioned in the textbook on p. 22. In particular, putting a semicolon at the end of a line suppresses output, but the computation is still completed. For example, within the `for` loop above, we really do not need to see every step printed out. There is not much editing that you can do directly in the Console. Of course, before you press `Enter`, you can change things on the line where you are typing. If you make a mistake, you can press the “Up Arrow” to have the last command line brought back so that you can edit it. Repeatedly pressing the “Up Arrow” scrolls back (and the “Down Arrow” scrolls forward later) through the history of commands until you get to the one you want to edit and complete again.

2.3 Getting Help

The entries in a vector in Octave are most easily summed using a built-in Octave function called `sum`. If you are unsure of how to use an Octave function or command, you can always type `help` followed by the command name, and an explanation will be provided.

```
>> help sum
'sum' is a built-in function from the file libinterp/corefcn/data.cc

-- Built-in Function:  sum (X)
-- Built-in Function:  sum (X, DIM)
-- Built-in Function:  sum (... , "native")
-- Built-in Function:  sum (... , "double")
-- Built-in Function:  sum (... , "extra")
Sum of elements along dimension DIM.
```

If DIM is omitted, it defaults to the first non-singleton dimension.

The optional "type" input determines the class of the variable used for calculations. If the argument "native" is given, then the operation is performed in the same type as the original argument, rather than the default double type.

For example:

```
sum ([true, true])
=> 2
sum ([true, true], "native")
=> true
```

On the contrary, if "double" is given, the sum is performed in double precision even for single precision inputs.

For double precision inputs, the "extra" option will use a more accurate algorithm than straightforward summation.

For single precision inputs, "extra" is the same as "double".

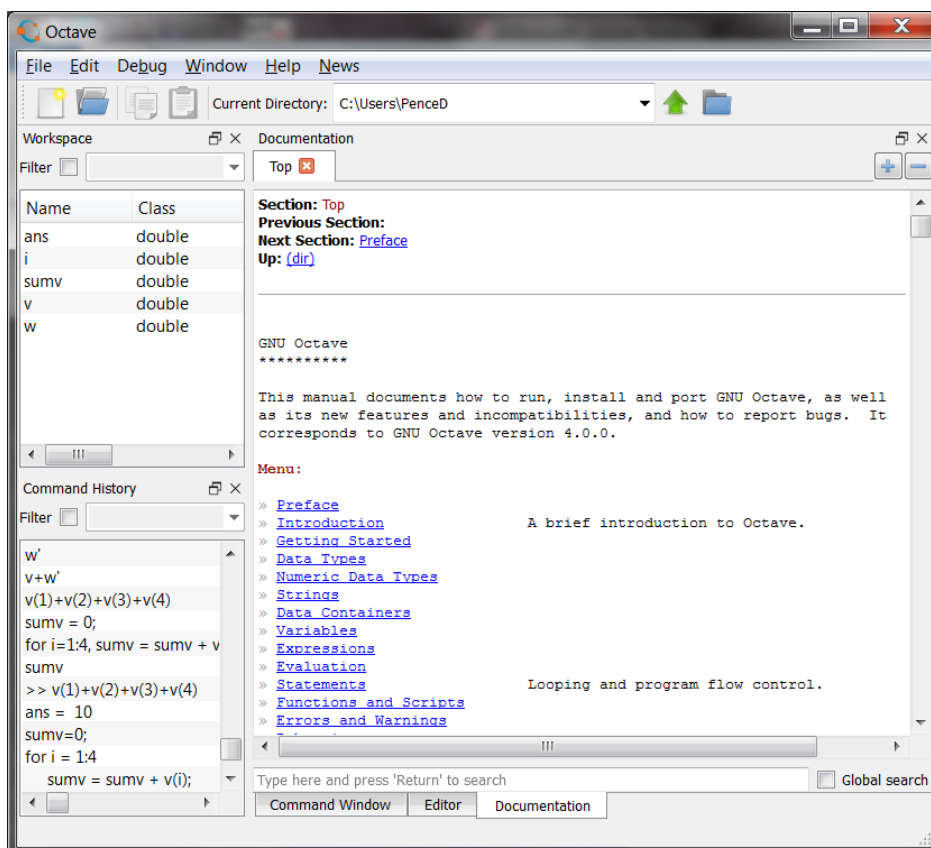
Otherwise, "extra" has no effect.

See also: `cumsum`, `sumsq`, `prod`.

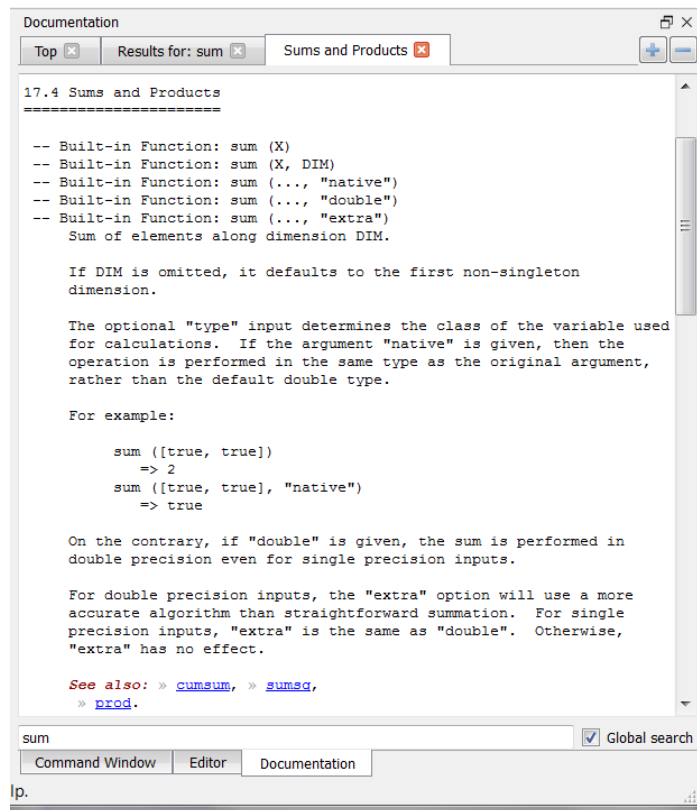
Additional help for built-in functions and operators is available in the online version of the manual. Use the command `'doc <topic>'` to search the manual index.

```
-- less -- (f)orward, (b)ack, (q)uit
```

Using the separate Octave Documentation in PDF-format, you can find `sum` in the Function Index. Then you can click on the hyperlinked page number and jump to the same explanation more nicely typeset. (When you installed GNU Octave you got this, and you can find this in Octave folder in the Programs Menu.) Finally, in the GUI interface, below the Command Window are three tabs (Command Window, Editor, Documentation). Selection the “Documentation” tab brings up another version of the Octave documentation, which is searchable (see the field near the bottom).



Notice in the above Documentation that the *Introduction* and *Getting Started* sections constitute a short tutorial. If we search for “sum”, we get (among other things) the same text as given above in the Command Window, but now more hyperlinked to other topics.



There is also a Frequently Asked Questions (FAQ) webpage that might be helpful. <http://wiki.octave.org/FAQ> In particular, where it attempts to explain the few differences between MATLAB and Octave, it has the following quote:

There are still a number of differences between Octave and Matlab, however in general differences between the two are considered as bugs. Octave might consider that the bug is in Matlab and do nothing about it, but generally functionality is almost identical. If you find an important functional difference between Octave behavior and Matlab, then you should send a description of this difference (with code illustrating the difference, if possible) to <http://bugs.octave.org>.

Furthermore, Octave adds a few syntactical extensions to Matlab that might cause some issues when exchanging files between Matlab and Octave users.

As both Octave and Matlab are under constant development, the information in this section is subject to change.

2.4 Matrices

Octave also works with matrices:

```
>> A = [1, 2, 3; 4, 5, 6; 7, 8, 0]
A =
   1   2   3
   4   5   6
   7   8   0
>> b = [0;1;2]
b =
   0
   1
   2
```

There are many built-in functions for solving matrix problems. For example, to solve the linear system $Ax = b$, type `A\b` to get:

```
>> x = A\b
x =
  6.6667e-001
 -3.3333e-001
  2.4672e-017
```

Note that the solution is printed out in scientific notation with only five significant digits. It is actually stored in more places (see chapter 5). To see more, use the `format` command. The default number of “places,” excluding the sign of the number and the sign of the exponent, but including the decimal point and the exponent, is 10. This is (default) `format short`. To see more, switch to `format long`.

```
>> format long
>> x
x =
  6.666666666666667e-001
 -3.333333333333333e-001
  2.46716227694479e-017
```

Other options include `format short e` and `format long e` to display numbers always in scientific notation. The command `format` with nothing else restores to the default.

We can check the answer in Octave to see the true precision (and also to compare with the textbook MATLAB result).

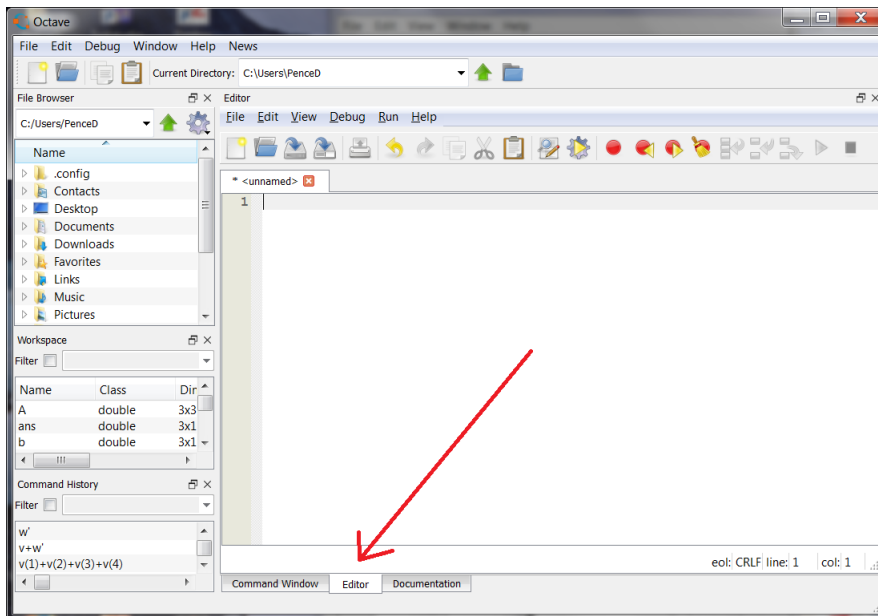
```
>> format
>> b - A*x
ans =
  3.7007e-017
  3.3307e-016
  4.4409e-016
```

You will notice that this result does not have exactly the same digits as printed in our textbook. The compatibility between GNU Octave and Matlab means that the similar commands try to do the same operations. It does not extend to exactly the same code underneath and exactly the same numerical results.

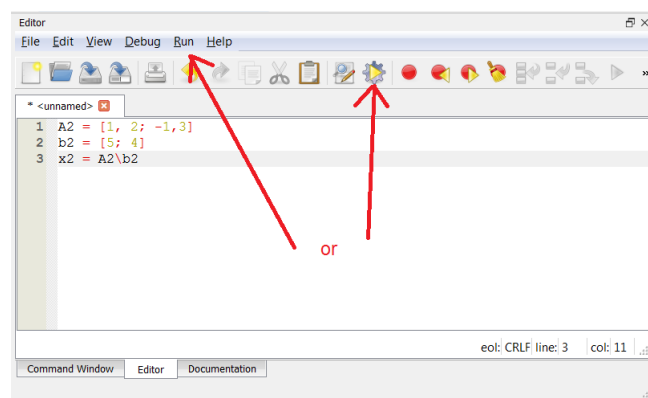
2.5 Creating and Running Files

Octave allows you to save a collection of commands that you have typed in a file to be executed later. Prior to version 4.0.0 (and still in the CLI-version), you needed to find your own text editor as a separate program. Now in the GUI-version, a text editor is included. However it is so new (April 2015) that there is almost no documentation about how to use this editor. Still the editing commands and options are fairly obvious.

To open the editor from the Octave console, click on the Editor toggle at the bottom of the GUI screen. The editor opens with a default file named “*`<unnaved>`”.



Any file can be saved under the **File - Save** and the files will be given the extension “.m” so that they can be associated with Octave (and probably be usable in Matlab). Back in the Octave Command Window, you must remember to change the working directory of Octave to the directory in which that file resides. From the editor, you can also have the commands executed. (The Run and Save asks you automatically if you want to change the working directory.)



2.6 Comments

Octave allows “comments”, and these are particularly important to embed in code that you type in a file so that you explain the code. Such comments

have no effect on the code to be executed. You can either make an entire line a comment or you can append a comment to the end of a line of commands.

Octave generally prefers to use a sharp sign character (#) to denote a comment. But for MATLAB compatibility, it also allows the percent sign (%).

```
>> % Solve Ax = b
>> x = A\b; #This solves Ax = b and stores the result in x.
```

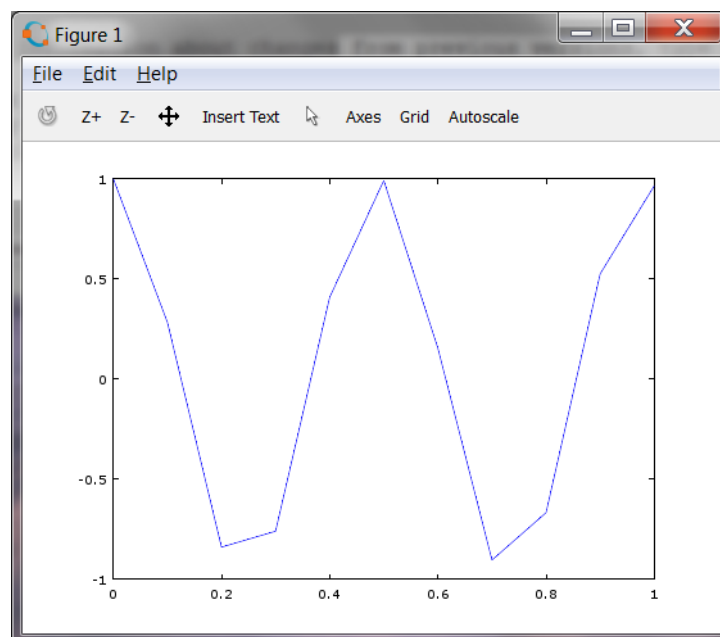
Anything following either sign is ignored in the rest of the line. Comments are particularly important in text files that you are going to save, so that you can later understand what is intended and so that others who might use them can understand.

2.7 Plotting

Here we show the commands and the results to reproduce the plots in Figure 2.2 and Figure 2.3 in the textbook in Octave.

In Octave we do the following:

```
>> x = 0:0.1:1; #Form the (row) vector of x values.
>> y = cos(50*x); #Evaluate cos(50*x) at each of the x values.
>> plot(x,y) #Plot the results.
```

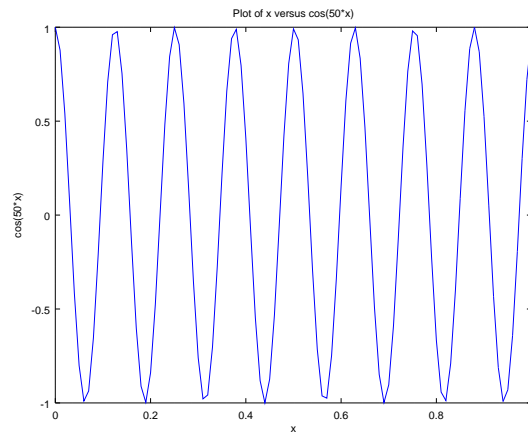


The graphic window that appears (above) contains the plot. There are then many interactive actions that you can make with the figure. While the above figure was simply obtained by a screen capture in Window and some slight editing in Paint, we will explore the options for saving plots under the File menu in the further plots below.

Obviously, this is a very poor plotting of this function, with only 11 points sampled. Notice that by default, straight lines connect the plotted pairs (x_i, y_i) passed to the `plot` command. Next we increase the size of the plot vectors, and we add a title and labels for the axes.

```
>> x = 0:0.01:1; #Create a vector of 101 x values.
>> plot(x,cos(50*x)) #Plot x versus cos(50*x)
>> title('Plot of x versus cos(50*x)')
>> ylabel('cos(50*x)')
>> xlabel('x')
```

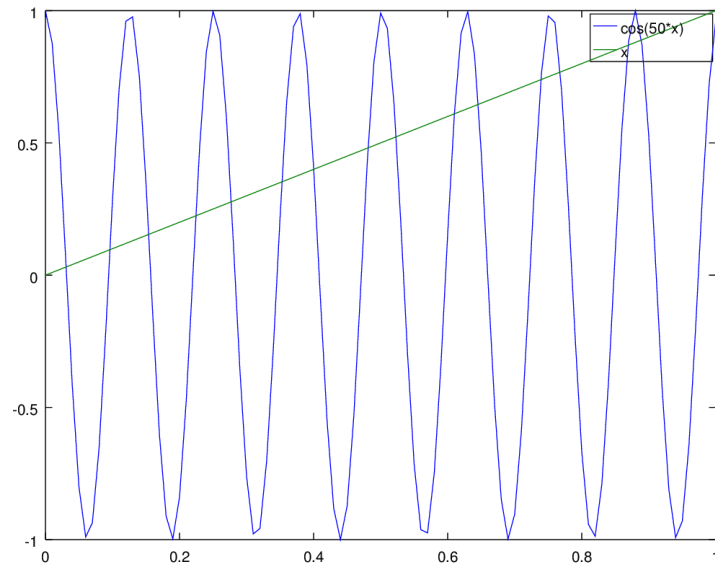
If you watched the graphics window when you executed these new commands, you found that the new plot command *replaced* the previous plot that was there, and then the further details (title, labels) each appear one-at-a-time. The default format for saving the plot is as a PDF. I have done that, and then imported the resulting file into my \TeX editor (\LyX) below.



The result has excessive “white space” surrounding the desired plot. Here are the commands to get Figure 2.3 for the textbook.

```
>> plot(x,cos(50*x),x,x)
>> legend('cos(50*x)', 'x')
```

Instead of saving as a PDF, here the **Copy** command under the **Edit** menu was used. This was then **Pasted** into Paint, and saved as a PNG file (but there are many other graphic file options in Paint). This is the result, with not so much wasted “white-space” all around.

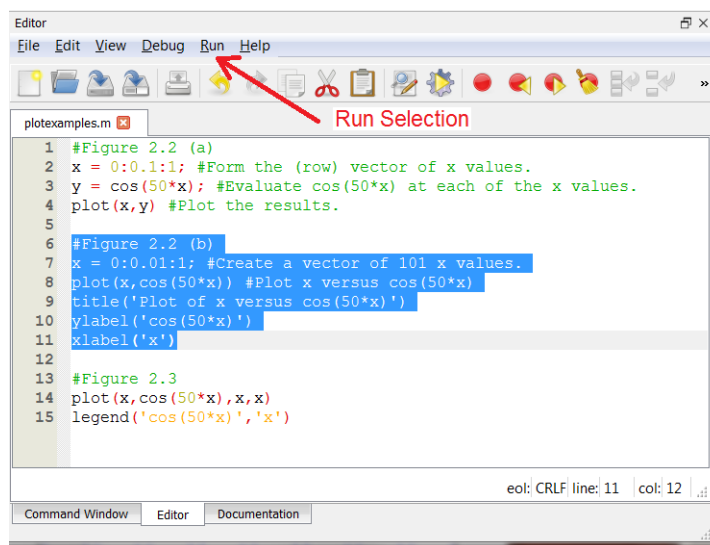


You can get the animation suggested in the textbook with the Octave commands below (but it seems to occur much more slowly, and so I have used a smaller x -vector below compared to the textbook or what I could use in Scilab). Unfortunately, this seems to indicate that more elaborate plotting might take longer to be generated in Octave.

```
>> x = 0:0.01:10;  
>> comet(x,cos(3*x))
```

Perhaps there are other ways to save and manipulate the resulting plots, but here, at least, are the ways to get the figures from the textbook on the screen.

For all of the typing above, it makes more sense to work in the editor, where you can easily save your work and change it. You can use the mouse to select a section of code to be executed (and there is even a **Save and Execute** command and a **Run Selection**).



2.8 Creating Your Own Functions

You can create your own functions in Octave. The simplest way is demonstrated here.

```

>> function y = f(x)
      y = x.^2 + 2*x;
endfunction
>>

```

Notice in the above that the section of code defining a function begins with the command `function` and ends with the command `endfunction`, and you do not get another prompt (`>>`) until after the `endfunction` command. The variable `y` signifies the output and the variable `x` will be the input. The name of the function created is the variable name `f`. All functions *expect* the input to be a vector or matrix. Thus if you typed only `x^2` in the above function formula, it would attempt to find the square of the matrix `x` (or otherwise to multiply the vector by itself, resulting in an error). Instead we desire the square of individual elements in the variable `x`, and that is what is indicated by the “dot carrot” notation. Thus not only can we evaluate `f(1.5)`, but we can evaluate a whole vector. Notice also that we have suppressed output at the end of the function formula with a semicolon.

```

>> f(1.5)
ans = 5.2500

```

```
>> f([0:5])
ans =
    0    3    8   15   24   35
>>
```

It is possible to “pack” very simple formulas into one line, but there is no real advantage.

```
>> function y=g(x), y=5*x.^3+2*x.^2-x; endfunction
>> g(2.3)
ans = 69.115
```

Of course you will often define functions in the editor (.m file), and it is very appropriate to include comments explaining the definition.

Octave does have an `inline` command like MATLAB, but the documentation has the following:

Caution: MATLAB has begun the process of deprecating inline functions. At some point in the future support will be dropped and eventually Octave will follow MATLAB and also remove inline functions. Use anonymous functions in all new code.

So here is an anonymous function example (where we have also named the “anonymous function”, but in many uses you do not).

```
>> newf = @(x) (4*x.^2+5*x);
>> newf(-5.6)
ans = 97.440
```

2.9 Printing

To be really honest, I seldom worry much about fancy “printing” of output in the command window. I am usually moving the data to a word processor for a more polished format anyway. [It brings back bad memories of what I had to do when I used to program in FORTRAN!] Still Octave has most of the printing commands of Matlab. For Octave see in the documentation *Chapter 14, Input and Output* (where you will find `display`, `disp`, `printf`, `fprintf` and `sprintf` that behave almost exactly like in Matlab).

```

>> x = 0:0.5:2;
>> display(x)
    0.00000    0.50000    1.00000    1.50000    2.00000

>> disp(x)
    0.00000    0.50000    1.00000    1.50000    2.00000

>> disp(['x = ',num2str(x)])
x = 0          0.5          1          1.5          2

>> disp('    Score 1    Score 2    Score 3'), disp(rand(5,3))
    Score 1    Score 2    Score 3
    0.485476    0.837004    0.160101
    0.856768    0.230565    0.393887
    0.930513    0.247282    0.893566
    0.795853    0.012911    0.837617
    0.173847    0.260161    0.108334

>> fprintf('    x          sqrt(x)\n=====\\n'), ...
for i=1:5, fprintf('%f          %f\\n',i,sqrt(i)), end
    x          sqrt(x)
=====
1.000000      1.000000
2.000000      1.414214
3.000000      1.732051
4.000000      2.000000
5.000000      2.236068

```

2.10 More Loops and Conditionals

Loops that pass through a section of code a predetermined number of times or with some kind of conditional termination are available in every programming environment. For Octave see in the documentation *Chapter 10, Statements* (where you will find `if`, `switch`, `while`, `do-until` and `for` that behave almost exactly like in Matlab).

2.11 Clearing Variables

Octave's `clear` command works the same as in Matlab for clearing variables.

2.12 Logging Your Session

In Octave, the `diary` command work in the same way as in Matlab. I virtually never do this in the Windows environment, preferring to Copy and Paste the commands and results into my word processor directly. It is more an old UNIX procedure for saving the results of your session.

2.13 More Advanced Commands

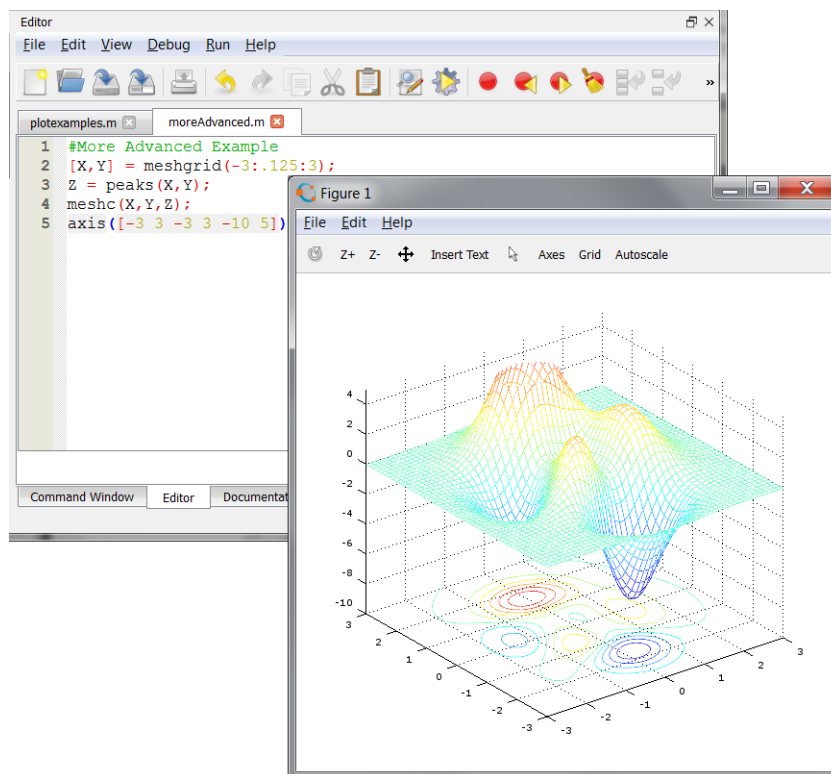
Here is the Octave documentation (from the PDF file, p. 368) for the `peaks` command (which is really a special example). The documentation actually has the wrong function below, but this is the correct one.

```
peaks () [Function File]
peaks (n) [Function File]
peaks (x, y) [Function File]
z = peaks (. . . ) [Function File]
[x, y, z] = peaks (. . . ) [Function File]
```

Plot a function with lots of local maxima and minima. The function has the form

$$f(x, y) = 3(1 - x)^2 e^{-(x^2 - (y+1)^2)} - 10 \left(\frac{x}{5} - x^3 - y^5 \right) e^{(-x^2 - y^2)} - \frac{1}{3} e^{-(x+1)^2 - y^2}$$

Called without a return argument, `peaks` plots the surface of the above function using `surf`. If `n` is a scalar, `peaks` plots the value of the above function on an n -by- n mesh over the range $[-3, 3]$. The default value for `n` is 49. If `n` is a vector, then it represents the grid values over which to calculate the function. If `x` and `y` are specified then the function value is calculated over the specified grid of vertices. When called with output arguments, return the data for the function evaluated over the meshgrid. This can subsequently be plotted with `surf (x, y, z)`. See also: `[sombbrero]`, page 351, `[meshgrid]`, page 316, `[mesh]`, page 306, `[surf]`, page 308.



Here is a summary of the elementary commands. (See Chapter 8 in the documentation.) First here are the elementary mathematical operators. Commands on the left below operate on matrices. Commands on the right below are element-wise.

Operator	Action	Operator	Action
+	addition	.+	element-wise addition
-	subtraction	.-	element-wise subtraction
*	multiplication	.*	element-wise multiplication
/	right division, i.e. xy^{-1}	./	element-wise right division
\	left division, i.e. $x^{-1}y$.\	element-wise left division
^	power, i.e. x^y	.^	element-wise power
**	power (same as ^)	.**	element-wise power
'	conjugate transpose	.'	transpose
c transpose()	conjugate transpose	t ranspose()	transpose

Most of the trigonometric functions are element-wise. (See Chapter 17 in the documentation.) The standard trigonometric functions assume the argument is in radians.

sin, cos, tan, cot, sec, csc

If you want the argument to be understood in degrees instead, the command ends in a “d”.

```
sind, cosd, tand, cotd, secd, cscd
```

The inverse (or *arc*) trigonometric functions add the letter “a” to the beginning of the command (with the result normally in radians, unless the command ends in a “d” to get degrees).

```
asin, acos, atan, acot, asec, acsc  
asind, acosd, atand, acotd, asecd, acscd
```

The hyperbolic trigonometric functions and their inverses (again *arc*), end with the letter “h”.

```
sinh, cosh, tanh, coth, sech, csch  
asinh, acosh, atanh, acoth, asech, acsch
```

The element-wise exponential, logarithmic, square root, and cub root functions are the following. (Note that `log` is the natural logarithm, `log10` is the common logarithm with base 10, and `log2` is the base 2 logarithm. Also note the the `cbrt` function will give correct negative results for a negative x , unlike $x^{1/3}$ which may give another complex result.)

```
exp, log, log10, log2, sqrt, cbrt
```

There is an extension of some of these functions for square matrices that is not given element-wise, and these end with the letter “m”.

```
expm, logm, sqrtm
```

Here are some of the predefined mathematical constants. (See Chapter 17 in the documentation.) The Euler number (the base of the natural exponential, i.e. `exp(1)`) is `e`. However, if you desire the natural exponential function, you get more accuracy using `exp(x)` rather than `(e).^x` so please use the given operation. The geometric constant we usually call π , is given by `pi` and the complex imaginary $\sqrt{-1}$ is given by `I` or `i` although it will output as `0 + 1i` in the console.

3 Monte Carlo Methods

We do not cover this chapter, and so I might do this later. I would point out that you can get the Matlab codes for the textbook examples (including the card game simulations) at one of the authors websites.

<http://academics.davidson.edu/math/chartier/Numerical/>

4 Solution of a Single Nonlinear Equation in One Unknown

4.1 Bisection