# Octave Tutorial 1: How to Get Started

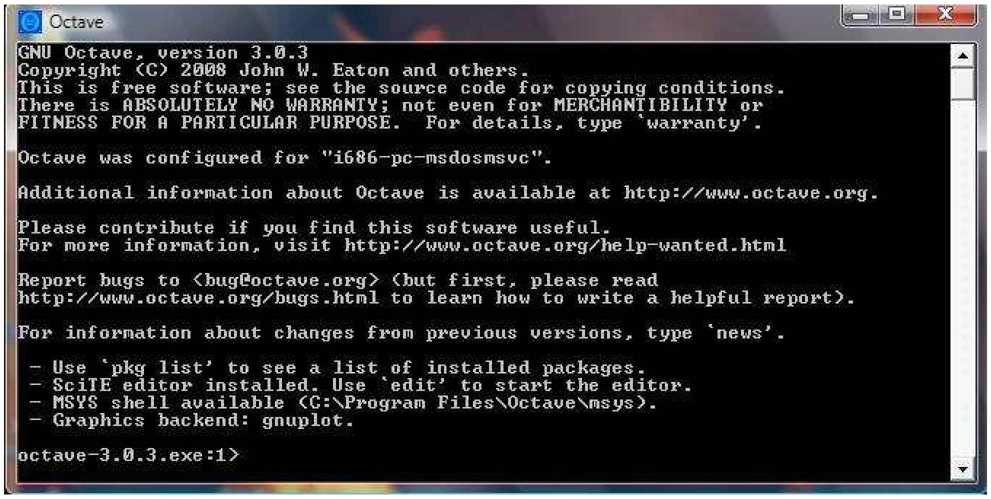
Extracts from *Introduction to Octave*, by P.J.G. Long

## 1 What is Octave?

Octave is an open-source interactive software system for numerical computations and graphics. It is specially designed for solving mathematical problems involving matrix computations: solving simultaneous equations, computing eigenvectors and eigenvalues and so on. C++ and other industry-standard programming languages do not natively support many of these mathematical concepts, or displaying graphics, and often require more time to program. So Octave is the ideal tool for this kind of problems, and more specifically for a linear algebra course.

## 2 Starting Octave

If not already running, start Octave. If you are a Windows user, select **Start→ Programs→ GNU Octave**. If you are a Linux user, type `octave` in a xterm window. Wait for the window to display the Octave header. You should see something like this:



```
GNU Octave, version 3.0.3
Copyright (C) 2008 John W. Eaton and others.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTIBILITY or
FITNESS FOR A PARTICULAR PURPOSE.  For details, type `warranty'.

Octave was configured for "i686-pc-msdosmsvc".

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/help-wanted.html

Report bugs to <bug@octave.org> (but first, please read
http://www.octave.org/bugs.html to learn how to write a helpful report).

For information about changes from previous versions, type `news'.

 - Use `pkg list' to see a list of installed packages.
 - SciTE editor installed. Use `edit' to start the editor.
 - MSYS shell available (C:\Program Files\Octave\msys).
 - Graphics backend: gnuplot.

octave-3.0.3.exe:1>
```

Now you are in the Octave environment. The `octave-3.0.3.exe:1>` is the Octave prompt. This may look slightly different depending on what version of Octave you have installed. For simplicity, in this and other tutorials we will refer to

`octave#:#>`

as the Octave prompt.

If you want to leave Octave at any point, type `quit` (or `exit`) at the prompt and press enter.

# 3 The Octave environment

As we will see from the examples below, Octave has a *command-line* interface - commands are typed in one at a time at the prompt, each followed by a carriage return. This creates output or plots.

## 3.1 Simple calculations: Octave as a calculator

The simplest way to use Octave is just to type mathematical commands at the prompt, like a normal calculator. All of the usual arithmetic expressions are recognised. For example, type

```
octave#:#> 2+2
```

at the prompt and press return. You should see

```
ans = 4
```

The basic arithmetic operators are `+` (addition), `-` (subtraction), $*$ (multiplication), `/` (division), and ˆ is used to mean "to the power of" (e.g. `2^3` = `8`). Brackets `( )` can also be used. The order precedence is the same usual, i.e. brackets are evaluated first, then powers, then multiplication and division, and finally addition and subtraction. Try a few examples.

## 3.2 Numbers and formatting

Octave usually displays numbers to five significant figures. However, it stores them internally, and in variables, to a much higher precision, so the answer given as output is the more accurate one. For example,

```
octave#:#> 3*0.76619
ans = 2.2986
```

this is not the answer that would be expected from simply performing $3 \times 0.76619$, but it is the more accurate to *five* significant figures.

To get Octave to display numbers to fifteen significant figures, type

```
octave#:#> format long
```

and press enter. Now repeat the multiplication above and see how the displayed answer changed.

To return Octave to its normal display accuracy (five significant figures) type

```
octave#:#> format short
```

and press enter.

In its normal display accuracy, Octave displays very large or very small numbers using exponential notation. For example, $13142.6 = 1.31426 \times 10^4$ is displayed by Octave as `1.3143e+004`.

There are also some other kinds of numbers recognised, and calculated, by Octave:

**Complex numbers**, e.g. `3+4i`, are fully understood by Octave.

**Infinity** (`Inf`), this is the result of dividing by zero.

**Not a Number** (`NaN`), this is the result of zero divided by zero, and also of some other operations which generate undefined results.

## 3.3 Built-in functions

As well as the basic operators, Octave provides all of the usual mathematical functions, a selection of these is listed below.

| | |
|---|---|
| `cos` | Cosine of an angle (in radians) |
| `sin` | Sine of an angle (in radians) |
| `tan` | Tangent of an angle (in radians) |
| `exp` | Exponential function ($e^x$) |
| `log` | Natural logarithm ($\ln x$) |
| `atan` | Inverse tangent |
| `sqrt` | Square root |
| `abs` | Absolute value |

These functions are invoked by typing the name of the function and then the function argument (or arguments) in ordinary brackets `( )`. For example,

```
octave#:#> exp(1)
ans = 2.7183
```

## 3.4 Named variables

In any significant calculation you are going to want to store your answers, or reuse values, just like using the memory on a calculator. Octave allows you to define and use named variables, but does not require any type of declaration or dimension statement (unlike most high level language, e.g. C++). For example, if you type

```
octave#:#> num = 25
```

and hit return, Octave creates a variable named `num` and stores the value 25 in it.

If the variable `num` already exists, Octave changes its previous value to 25. Once defined, variables can be used as function arguments. Thus

```
octave#:#> a = sqrt(num)
a = 5
```

Octave is case sensitive: it distinguishes between uppercase and lowercase letters. `A` and `a` are *not* the same variable for Octave.

Every time you type in an expression which is not assigned to a variable, Octave assigns the answer to a variable called `ans`. This can be used in exactly the same way as any other variable, however its value changes whenever the output of a new calculation is not stored in a user-named variable. For example, if you enter

```
octave#:#> exp(1)
ans = 2.7183
```

and then

```
octave#:#> log(ans)
ans = 1
```

the value of `ans` changed from 2.7183 to 1. So be careful when you use `ans` as a variable!

If you want to see the value of a variable at any point, simply type the variable name and press return. For example, if you entered all the examples above in the same order as they were given, the value stored in the variable `a` should be 5. Check it. Type `a` and press enter. Octave will display the current value of `a`.

There are also some variable names that have been pre-defined in Octave and can be readily used in calculations. For example, the number $\pi$ is stored (to 15 decimal places) in the variable `pi`. So, if you want to calculate the cosine of $\pi$, you'll type

```
octave#:#> cos(pi)
```

Octave will return

```
ans = -1
```

Other example is the imaginary number $i$. Unless otherwise defined, Octave recognizes `i` as $\sqrt{-1}$.

## 4 Other useful commands

### 4.1 Repeating previous commands

Octave keeps a record of all the commands you have typed during a session and you can use the cursor keys ↑ and ↓ to review the previous commands (with the most recent first). Try it!

Once a command has been recalled, you can edit it before running it again. You can use ← and → to move the cursor through the line. This is particularly useful if you have just typed a long line and then Octave finds an error in it. Using the arrow keys you can recall the line, correct the error, and try it again!

### 4.2 Getting help

If you are not sure what a particular Octave command does, or want to find a particular function, Octave has an integrated help system. For example, if you don't remember what the function `abs` does, type

```
octave#:#> help abs
```

and then press enter. Octave will return information on the function `abs`, including commands on how to navigate through the help window.

In general, the basic form of using help is to type

```
octave#:#> help commandname
```

### 4.3 Cancelling a command

If you find yourself having typed a command which is taking a long time to execute, it is very useful to know how to stop it! You can cancel the current command by pressing `Ctrl -C`.

## 4.4 Semicolons and hiding answers

The semicolon `;` serves a useful purpose in Octave. If you type a command at the prompt, as we have been doing so far, Octave always displays the result of the expression. However, if you finish the line with a semicolon, the semicolon will inctruct Octave not to display the result. Try it. Type

```
octave#:#> b=sin(pi);
```

and hit return. Octave will perform the calculation but will suppress the output. To convince you, now type `b` (without semicolon) at the prompt and press enter. Octave will now display the output.

This use of the semicolon is particularly useful if you don't need to know the result there and then, or the result would otherwise be an enormous list of numbers.