

Creative Computing II

Introduction to *Octave*

10th November 2009

This lab sheet serves as an introduction to running and interacting with *Octave*.

1. First, we need to check that *Octave*, and the particular functionality we'll be using, actually runs on whatever machine you're using. The lab machines should have *Octave* installed and working, with an obvious icon on the desktop.

If you are using Mac OS X, you may not have it installed; binary downloads are available from <http://www.gnu.org/software/octave/download.html>. You will also need some packages from <http://octave.sourceforge.net/packages.html>; download at least the 'Audio', 'Image' and 'Signal' packages. To install them, you will need to start *Octave*, then at the prompt type `pkg install filename`.

If you are using Debian GNU/Linux or one of its derivatives (such as Ubuntu), simply install the `octave`, `octave-audio`, `octave-image` and `octave-signal` packages, using your favourite interface to the APT package manager. Other distributions are likely to have similar packages.

Start *Octave* and type in the following commands, hitting Return after each line:

- (a) `t = [0:1/8000:1];`
- (b) `x = sin(2*pi*440*t);`
- (c) `stem(t(1:40), x(1:40), '*');`
- (d) `sound(x,8000);`

The command in 1c should produce graphical output, and the command in 1d should produce a one-second sound. If either of these commands fails to do so, there is something wrong with your installation (but check your volume control first!)

2. As you should have seen in part 1, the *Octave* environment can be used interactively; you do not need to have a complete program, but can effectively treat the prompt as a souped-up calculator. This part demonstrates a little bit of the syntax for doing arithmetic on scalars and using scalar variables.

- (a) To investigate *Octave*'s capabilities for arithmetic, including the four basic operators, exponentiation and square roots, type the following at the prompt, each time followed by Return:

- i. `12`
- ii. `4 + 8`
- iii. `4 + 8;`
- iv. `198 - 186`
- v. `3 * 4`
- vi. `84 / 7`
- vii. `sqrt(144)`

- viii. $3 + 3 * 3$
- ix. $(3 + 1) * 3$
- x. $3 * 2 ^ 2$

Check that you understand what the interpreter prints for each of these cases. (To explain part 2(a)iii, you may want to read ‘Simple Examples’, section 1.2 of the GNU *Octave* manual.)

- (b) You can store the result of a calculation in a variable; variables are named with textual identifiers:

- i. `a = 3`
- ii. `b = 4;`
- iii. `a * b`
- iv. `cos(pi)`

As demonstrated in 2(b)iv, some variables have predefined values: `pi` is one, and others are `e` and `i`: you may get odd results if you attempt to use these as variable names in your programmes.

3. Vectors and Matrices

- (a) Vectors are sequences of scalars; they can be constructed in a variety of ways:

- by explicitly specifying the contents: `[0 1 2 3 4 5]`
- by specifying start, step and end: `[0:1:5]`
- by specifying start and end (step defaults to 1): `[0:5]`
- using the `linspace` operator: `linspace(0, 5, 6)`

Type each of these expressions into the *Octave* prompt; check that the same vector is returned. Use each of these construction methods to build the vector starting at -3 and ending at 3, with a step of 1.

- (b) Many arithmetic operations work in a similar way on vectors as they do on scalars, operating element-by-element:

- addition and subtraction by a scalar: `1 + [0:5]` and `[0:5] - 2`
- multiplication and division by a scalar: `[0:5] * 3` and `[0:5] / 3`
- addition and subtraction of two vectors: `[0:5] + [0:5]` and `[0:5] - [0:5]`

Type each expression into the prompt and check that you understand the vectors that are returned.

- (c) There are two different kinds of multiplication of vectors: element-by-element (known as the Hadamard product) and the matrix product (treating a vector as a $1 \times n$ or $n \times 1$ matrix). These are respectively denoted by `.*` and `*` – in other words, the normal multiplication in *Octave* is the matrix multiplication; to get element-by-element multiplication, you need to prepend a dot.

- Hadamard multiplication: `[0:5] .* [5:-1:0]`
- `[0:5] .* [1:5]` is an error, because the dimensions don't match
- matrix multiplication: `[0:5] * [5:-1:0]'` (' indicates the transpose of a vector)

- $[0:5]'$ * $[5:-1:0]$ is also a valid matrix multiplication, but gives a different answer from above; why?
- $[0:5]$ * $[5:-1:0]$ is an error, because the dimensions are nonconformant

Again, type these expressions in and check that you understand the results.

- (d) Chapter 3, volume I of the CC227 subject guide has a number of learning activities related to *Octave*. Work through the activities and the worked examples in section 3.2 (pp. 15–29)

Other resources:

- the online help in *Octave*: type `help help` at the prompt, and read what is printed. Type `help name` to get help on the `name` operator
- Amuasi, H. *Octave Tutorial*. Available at <http://www.aims.ac.za/resources/tutorials/octave/>
- Eaton, J. W. *The GNU Octave Manual*. Available at <http://www.gnu.org/software/octave/doc/interpreter/>