Familiarization with the Octave Programming Language: COSC360

Dr. Jordan Hanson - Whittier College Dept. of Physics and Astronomy

December 31, 2021

Abstract

The Octave programming language is an open-source software package designed for scientific computational work. The syntax is compatible in most respects with the proprietary package MATLAB. Both Octave and MATLAB are built from toolkits that cover a wide variety of applications within science and engineering. First, installation of Octave on a variety of systems is presented. Next, basic syntax and common techniques are presented. For more information, see `https://www.gnu.org/software/octave/index` (general information), `https://wiki.octave.org/GNU_Octave_Wiki` (Wiki), and `https://wiki.octave.org/FAQ` (FAQ).

1 Installation

Installation techniques for Windows, Mac OS, and Linux are covered below.

1.1 Installation for Windows

For complete details, see `https://wiki.octave.org/FAQ#Installation`. The most common installation method for current Windows operating systems is to download and run the binary installer (.exe file), following the prompts. Navigate to the following site:

```
https://www.gnu.org/software/octave/download.html
```

Scroll to the Microsoft Windows section, and under Windows-64 (recommended), run the link entitled `octave-6.4.0-w64-ins`
Most systems acquired in the last few years have 64-bit architecture. Following the prompts should complete the installation. Run GNU Octave and type the following commands in the command line:

```
pkg list
pkg install -forge control
pkg install -forge signal
pkg install -forge io
pkg install -forge statistics
```

The first command should state that no new packages are installed. The last four commands will each take a few moments to run. These commands cause the system to download and install toolkits from a site called SourceForge.

1.2 Installation for Mac OS

For complete details, see `https://wiki.octave.org/FAQ#Installation`. The most common installation method for Mac OS installation is to navigate to the following site:

```
https://wiki.octave.org/Octave_for_macOS
```

This is part of the GNU Octave Wiki. Find the link entitled "Download installer from GitHub." This link leads to a repository on GitHub, and under the section 6.2.0, we see a DMG file. This is the drag-and-drop installer for Mac OS. Download this file and double-click on it. When the install is finished, run GNU Octave and type the following commands in the command line:

```
pkg list
pkg install -forge control
pkg install -forge signal
pkg install -forge io
pkg install -forge statistics
```

The first command should state that no new packages are installed. The last four commands will each take a few moments to run. These commands cause the system to download and install toolkits from a site called SourceForge.

1.3 Installation for Linux

For complete details, see `https://wiki.octave.org/FAQ#Installation`. The most common installation method for Linux distributions is via the command line. On Ubuntu-based systems, for example, open the terminal and run the following command:

```
sudo apt install octave
```

Type your administrator password and the process should begin. To run GNU Octave, run the following commands from the command line:

- `octave-gui &`
- `octave-cli`

The first one opens the standard graphical user interface, and the second one opens octave in a command line interface via the shell. This is similar to running an "interactive session" in Python. Using either method, run the following commands to install some useful packages:

```
pkg list
pkg install -forge control
pkg install -forge signal
pkg install -forge io
pkg install -forge statistics
```

2 Basic Commands

Octave is a high-level *scripting* programming language. Although it is possible to write packages and compile code in octave, the most common application is executing a script that performs some analysis on digital data.

```
a = 1+1i;
b = conj(a);
a * b
```

The result of this code should be 2.0. Why? We are defining a complex number in the first line, computing the complex conjugate, and multiplying them. Octave naturally handles vectors of numbers and matrices. Let's define a vector of complex numbers.

```
a = [1 2 3 5 7 11];
size(a)
ans = 1   6
a = a';
size(a)
```

The code in the fourth line *transposes* the vector. This means trading the rows for the columns of the vector. What begins as a 1×6 vector (one row by six columns) ends as a 6×1 vector (six rows by one column). Operations are as expected, but we need special notation for vectoral calculations:

```
a = 2.0;
b = 4.0;
b/a
ans = 2
b = [4.0 4.0 4.0]
b/a
ans =
2   2   2
a./b
ans =
0.5   0.5   0.5
```

Placing a dot (.) before a standard operation indicates that the operation is to be carried out in a vectoral-sense.

```
t1 = [1 2 3 4 5 6 7 8 9 10];
t2 = t1+1;
t1.*t2
ans =
2   6   12   20   30   42   56   72   90   110
```

The colon operator (:) represents iteration in octave. Consider three cases:

```
fs = 1000.0;
t = [0.0:1/fs:10.0];
plot(t,sin(2.0 * pi * 3.0 .* t));
```

Octave should produce a plot of a sine wave with a frequency of 3.0 Hz (if the time is in seconds). How can you tell? Count the number of complete oscillations in 2.0 seconds. Do you see 6.0 oscillations? What is the significance of f_s in the code?

```
fs = 1000.0;
t = [0.0:1/fs:10.0];
plot(t,sin(2.0 * pi * 3.0 .* t));
```

The `axis` command is useful for controlling the plotted region:

```
axis([0 2 -2 2]);
```

The colon operator (:) also refer to elements in a vector.

```
t(1)
ans =
0
t(2:5)
ans =
0.001   0.002   0.003   0.004
t(:)
ans =
(should be the whole vector)
t(1:end)
ans =
(should be the whole vector)
```

Now you know how to create vectors and plot them in octave. Let's solve a few programming problems. First, let's introduce the `help` command.

```
help <command>
```

Type the following commands

1. help plot
2. help :

What is the purpose of the *clear* and *home* commands?
The key here is that if you are confused, the help command can boost your understanding of the correct usage.

Speaking of the colon command, how do we create matrices? Create a matrix in octave in one of three ways:

1. **Concatenate row vectors**
2. Concatenate colomn vectors
3. Write it straight away

First, concatenation of row vectors:

```
a = [1:10];
b = [11:20];
A = [a; b];
display(A)
```

The semi-colon also tells the [] operators to concatenate vertically. What is the size of the matrix? Create a matrix in octave in one of three ways:

1. Concatenate row vectors
2. **Concatenate colomn vectors**
3. Write it straight away

First, concatenation of row vectors:

```
a = [1:10]';
b = [11:20]';
A = [a b];
display(A)
```

The lack of the semi-colon tells the [] operators to concatenate horizontally. What is the size of the matrix? Create a matrix in octave in one of three ways:

1. Concatenate row vectors
2. Concatenate colomn vectors
3. **Write it straight away**

First, concatenation of row vectors:

```
A = [[1 2]; [3 4];];
display(A)
```

In this case, you can actually omit the internal concatenation operators. Can you guess how to take the transpose of the matrix? Create a matrix with complex numbers. First, concatenation of row vectors, and take the transpose. Do you notice something?

```
A = [[1 2i]; [3i 4];];
display(A)
```

Try using the *help* command on the transpose operator to sort out the issue. Now look up the *conj* command, and use it to take *just the transpose* of the matrix, without conjugation. Two additional commands that are helpful for creating square ($N \times N$) matrices:

```
ones(N,M)
zeros(N,M)
```

Create a 2×2 matrix and multiply it with a 2×1 vector. *Careful with the order; the matrix has to be on the left.* You can imagine a matrix as an **operator** that acts on a vector. Example: two-dimensional rotation matrix (observe on board).

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{1}$$

Code the rotation matrix (Eq. 1) for $\theta = \pi/4$, and also create three 2×1 column vectors representing x and y coordinates. Multiply them and display the results. How are the resulting vectors related to the original vectors? The answer is that they have been *rotated* by 45 degrees.