Music 208 Winter 2014
John Ellinger, Carleton College

Introduction to Octave

Octave is a free open source software program for doing math, numerical analysis and Digital Signal Processing (DSP). Octave is mostly compatible with a very expensive commercial program called Matlab. It is also similar to Mathematica but easier to use for digital audio experiments. There are thousands of web pages devoted to learning Octave and Matlab.

Every major audio editing program has DSP units that create and manipulate sounds. Later in this course you'll be using effects like compressors, equalizers, limiters, delay lines, and reverb. You'll have access to different types software synthesizers that use techniques like: additive synthesis, subtractive synthesis, ring modulation, frequency modulation, and physical modeling. All of these effects and synthesis techniques mathematically manipulate the individual samples in a sound file. Audio sound files often consist of several thousand to several million samples. Octave is especially good at this and will help you understand how several effects and synthesis techniques work.

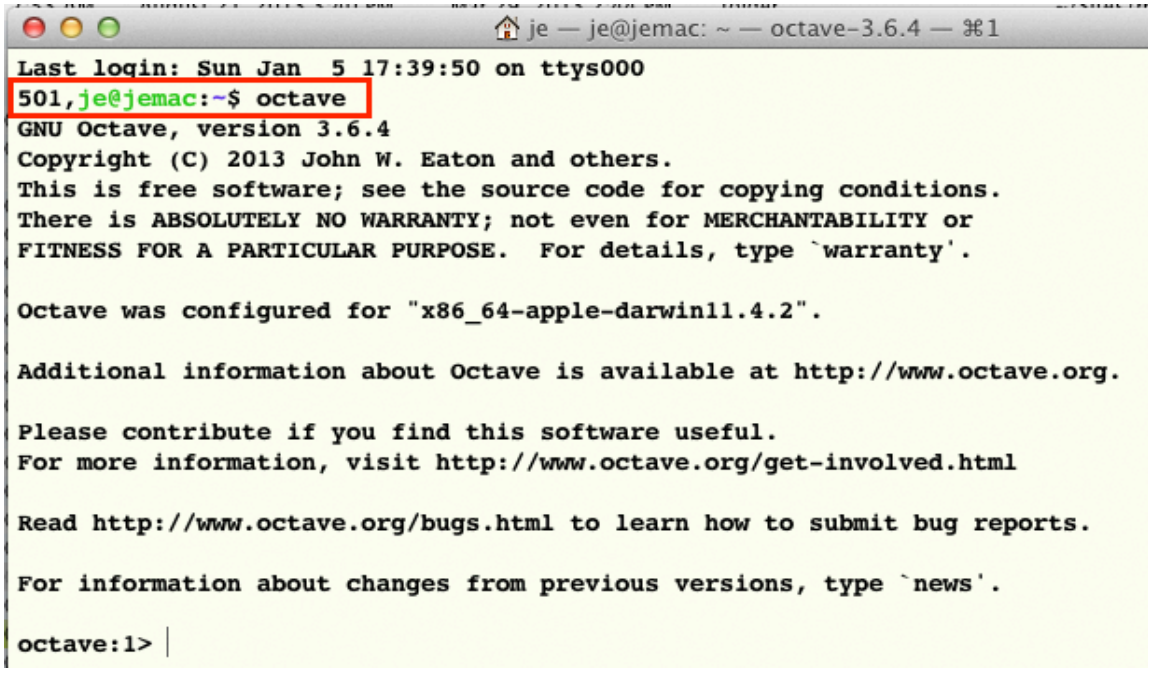
Start Octave

Mac 10.6 Use Octave.app in /Applications

Mac 10.7 Use Octave.app in /Applications or install using 10.8 method

Mac 10.8, 10.9 Start Octave from Terminal

Windows Start Octave from desktop shortcut



```
Last login: Sun Jan  5 17:39:50 on ttys000
501,je@jemac:~$ octave
GNU Octave, version 3.6.4
Copyright (C) 2013 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 "x86_64-apple-darwin11.4.2".

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'.

octave:1>
```

You'll be greeted with octave information and the octave prompt >. Octave runs in a Terminal window as a command line program. You enter commands and press Enter to execute them. Type commands one line at a time. Type Enter to execute the command. When you see this text style type command directly into octave at the octave command prompt > and press enter.

Specifying Mac and Windows Pathnames in Octave

The Mac uses forward slashes to separate directories. The top level of the hard drive is the first /. This example indicates that a folder named Volumes is at the top level of the hard drive. Inside Volumes is a folder named MC16 with another folder inside MC16 named m208.

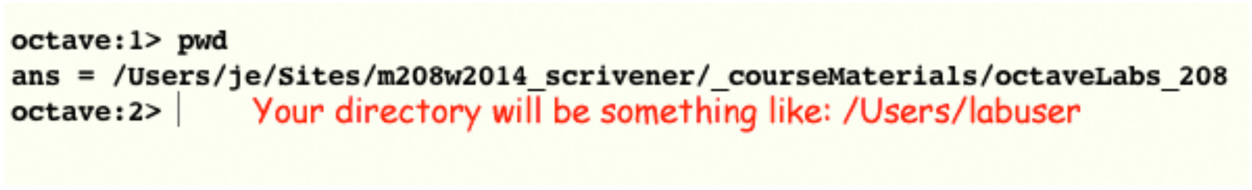
```
/Volumes/MC16/m208
```

Windows uses \ to separate directories. In the Octave Terminal on windows a single \ is interpreted as an "escape" character so you need to separate directories with a double backslash \\. Also the top level hard drive is usually C:\ which becomes C:\\. The example given above when translated to the Octave windows terminal becomes:

```
C"\\Volumes\\MC16\\m208
```

pwd

Octave should respond with your home directory.



```
octave:1> pwd
ans = /Users/je/Sites/m208w2014_scrivener/_courseMaterials/octaveLabs_208
octave:2> |
```

Your directory will be something like: /Users/labuser

VERY IMPORTANT: Always double check your working directory with pwd. That is the directory Octave will use to read and write files. It simplifies things if you keep all your Octave files in one directory on your USB flash drive.

Octave Prompt

The octave prompt > is prefixed by "octave:1." The prefix displays the application name and the command number after the colon.

Comments

Comments in octave are prefixed by a % or #.

```
% this is a comment
# this is also a comment
```

Math Operators

All the usual math operators and functions are available.

Plus	+		3 + 4
Minus	-		3 - 4
Multiply	*	Shift-8	3 * 4
Divide	/		3 / 4
Exponent Power	^	Shift-6	3 ^ 4
Grouping	( )		(3 + 4) * 3

Addition, Subtraction, Multiplication, Division, Power

To add 1 plus 4, type 1+4 and press Enter. Spaces between the math operators are optional.

```
% addition, subtraction, multiplication, division, power
1+4
16-7.52
3*4
12/5
2^3
```

A Semicolon Suppresses Output

Sometimes you want to see a result immediately, sometimes you don't. If the command line ends with a semicolon, Octave will not display the result.

```
octave-3.4.0:88> % a semicolon at the end of a line supreses output
octave-3.4.0:88> 7+3*10;
octave-3.4.0:89> % is the answer 100 or 37
octave-3.4.0:89> (7+3)*10
ans =  100
octave-3.4.0:90> 7+(3*10)
ans =  37
octave-3.4.0:91> 7+3*10
ans =  37
```

Octave calculates 7+3*10=37 because multiplication has a higher precedence that addition. When you are not sure about the order Octave will evaluate an expression you can use parentheses to make it clear.

Named Variables

You can assign a variable to an expression: <variableName> = <expression> Enter these commands one at a time. Terminate each line with a semicolon except for line 3, a3 = 3 * 4.

Spaces between operators (+ - * /) are ok.

```
% named variables
a1 = 1 + 4;
a2 = 16-7.52;
a3 = 3 * 4
a4 = 100/37;
a5 = 2^3;
```

Variable Values Are Saved And Can Be Displayed Anytime

Enter these variable names one at a time. Do not end with a semicolon.

```
octave-3.4.0:75> a1
a1 =  5
octave-3.4.0:76> a2
a2 =  8.4800
octave-3.4.0:77> a3
a3 =  12
octave-3.4.0:78> a4
a4 =  2.7027
octave-3.4.0:79> a5
a5 =  8
octave-3.4.0:80> ((a1+a2)*(a3-a4))/a5
ans =  15.666
```

Text Strings

A sequence of text characters is called a string. You can assign a string to a variable.

```
% variables can contain strings (a series of characters)
str = "This is a message.";

octave-3.4.0:82> str
str = This is a message.
```

printf

The printf command originated in the C programming language and has been adopted by many languages others including Octave. The basic format of a printf command is:

printf("string in quotes with format codes %d or %s or %f \n", variable1, variable2, variable3);

- items prefixed by % are called format codes
- %d indicates a decimal integer
- %f indicates a float or a number with decimal places
- %s indicates a string of characters
- %x indicates to display a number in hexadecimal format (%X for uppercase ABCDEF)

- the % format codes are enclosed in quotes, a comma follows the ending quote and variables follow the comma
- the number of the % format codes inside the quotes must match the number of variables listed after the quotes
- the order the % format codes appear corresponds to the order of the variables
- \n (backslash n) is a line break character
- \t (backslash t) is a tab character

Here's an example that uses all of the above format codes. The printf command will display output even when terminated by a semicolon.

```
octave-3.4.0:92> % variables can be formatted using the printf command
octave-3.4.0:92> str = "The value of a3 is:";
octave-3.4.0:93> printf("%s\n\tinteger: %d\n\tfloat: %f\n\thex: %X\n", str,
a3, a3, a3);
The value of a3 is:
      integer: 12
      float: 12.000000
      hex: C
```

Cancel A Command

Type Control-c.

Math Functions Useful In Digital Sound Processing (DSP)

```
% constants
pi;
2*pi;
e;
i;
% square root
sqrt(2);
% absolute value
abs(-53.2);
```

Trig Functions

```
% trigonometry
% angle in radians
sin(pi/8);
cos(3*pi/8);
% angle in degrees
sind(22.5);
cosd(90-22.5);
% arctangent, the angle in radians whose tangent is pi
4*atan(1); % equals pi
```

Powers And Logarithms

```
% powers and logarithms
% base 10
10^3; % ten to the third power
log10(1000);
% base 2
2^16 ;
log2(65536);
% natural
e^2;
log(7.3891);
```

Approximations

```
% round
round(1.5);
round(1.49999);
% use floor and ceiling (ceil) to round up or down
% round down
floor(1.5);
floor(1.49999);
% round up
ceil(1.5);
ceil(1.49999);
```

Complex Numbers

```
% complex numbers
z = 3+4i;

octave-3.4.0:95> magnitude = abs(z)
magnitude =  5
octave-3.4.0:96> phase = arg(z)
phase =  0.92730
```

The Octave Help System

Help

Execute help <command> to display help for that command.

```
octave-3.4.0:97> help abs
`abs' is a built-in function

 -- Mapping Function:  abs (Z)
     Compute the magnitude of Z, defined as |Z| = `sqrt (x^2 + y^2)'.
```

```
octave-3.4.0:98> help arg
`arg' is a built-in function

 -- Mapping Function:  arg (Z)
 -- Mapping Function:  angle (Z)
     Compute the argument of Z, defined as, THETA = `atan2 (Y, X)', in
     radians.
```

Online Help

http://www.gnu.org/software/octave/doc/interpreter/

http://www.gnu.org/software/octave/doc/interpreter/Function-Index.html#Function-Index

Format

The format command controls the number of decimal places.

```
octave-3.4.0:109> % format
octave-3.4.0:109> format short
octave-3.4.0:110> pi
ans =  3.1416
octave-3.4.0:111> format long
octave-3.4.0:112> pi
ans =  3.14159265358979
octave-3.4.0:113> % reset
octave-3.4.0:113> format short
octave-3.4.0:114>
```

Recall History

You can use the up and down arrow keys at the octave prompt to recall previous commands. Try it.

You can also execute the command **history**. Directions at bottom of screen f, b, q. Try it.

Arrays, Lists, Vectors

The samples in a sound file stored sequentially as a long list of numbers. Some programming languages call them arrays, others call them lists, Octave calls them vectors. Octave has functions for these common operations:

- creation
- deletion
- inserting
- retrieving
- removing
- replacing
- resizing
- querying current size
- reading and writing
- copying

Vectors

Octave vectors are one dimensional rows or columns of numbers.

You can create a vector three ways:

1. enclose numbers in brackets
2. as two numbers separated by a colon
3. as three numbers separated by colons

Each case will be discussed below.

1. Create a Vector by enclosin numbers separated by spaces and enclosed in brackets [ ].

```
octave-3.4.0:114> % you can create lists of numbers by specifying the numbers
(vectors)
octave-3.4.0:114> xRowV = [1 3 5.67 4 2];
octave-3.4.0:115> % a row vector
octave-3.4.0:115> xRowV
xRowV =

   1.0000   3.0000   5.6700   4.0000   2.0000

octave-3.4.0:116>
```

Column Vector

By typing a semicolon after each element you'll create a column vector.

```
% column vector
xColV = [1; 3; 5.67; 4; 2];
xColV =

   1.0000
   3.0000
   5.6700
   4.0000
   2.0000
```

An apostrophe turns a row vector into a column vector and vice versa.

```
octave-3.4.0:121> xRowV'
ans =

   1.0000
   3.0000
   5.6700
   4.0000
   2.0000

octave-3.4.0:122> xColV'
ans =

   1.0000   3.0000   5.6700   4.0000   2.0000
```

2. Create a vector with two numbers separated by a colon

The array begins with the first number, increments by one, and ends with the last number

```
octave-3.4.0:123> xRowV = 1:10
xRowV =

    1    2    3    4    5    6    7    8    9   10

octave-3.4.0:125> xColV = [1:10]'
xColV =

    1
    2
    3
    4
    5
    6
    7
    8
    9
   10
```

3. Create a vector using three numbers separated by colons

The array begins with the first number, increments by the second number, and ends with the last number.

```
octave-3.4.0:126> xRowV = 1:2:20
xRowV =

    1    3    5    7    9   11   13   15   17   19

octave-3.4.0:127> xRowV = 2:2:20
xRowV =

    2    4    6    8   10   12   14   16   18   20

octave-3.4.0:128> xRowV = 10:-1:1
xRowV =

   10    9    8    7    6    5    4    3    2    1

octave-3.4.0:130> xRowV = 0:.075:1
xRowV =

 Columns 1 through 8:

    0.00000    0.07500    0.15000    0.22500    0.30000    0.37500    0.45000
0.52500

 Columns 9 through 14:

    0.60000    0.67500    0.75000    0.82500    0.90000    0.9750

octave-3.4.0:131> xRowV = 0:1/7:1
xRowV =

    0.00000    0.14286    0.28571    0.42857    0.57143    0.71429    0.85714
1.00000
```

Find The Number Of Elements In A Vector

```
octave-3.4.0:132> length(xRowV)
ans =  8
```

Copy A Vector

```
octave-3.4.0:135> xRowV = [ 1 3 2 4 5 ];
octave-3.4.0:136> yColV = xRowV';
octave-3.4.0:137> yColV
yColV =

   1
   3
   2
   4
   5
```

Add A Constant To Every Element

```
octave-3.4.0:138> % add a number to each element in the vector
octave-3.4.0:138> yRowV = xRowV;
octave-3.4.0:139> yRowV = xRowV + 3
yRowV =

   4   6   5   7   8

octave-3.4.0:140> % equivalent result using the += operator
octave-3.4.0:140> yRowV = xRowV;
octave-3.4.0:141> yRowV += 3
yRowV =

   4   6   5   7   8
```

Multiply Every Element By A Constant

```
octave-3.4.0:142> % multiply every element by a constant
octave-3.4.0:142> yRowV = xRowV;
octave-3.4.0:143> yRowV = xRowV * 3
yRowV =

    3    9    6   12   15

octave-3.4.0:144> % equivalent result using the *= operator
```

```
octave-3.4.0:144> yRowV = xRowV;
octave-3.4.0:145> yRowV *= 3
yRowV =

    3    9    6   12   15
```

Find The Sum, Product, And Average Of Every Element In A Vector

Because sum and prod are built in functions they cannot be used as variable names.

```
octave-3.4.0:146> summ = sum(xRowV);
octave-3.4.0:147> prd = prod(xRowV);
octave-3.4.0:148> average = summ / length(xRowV);
octave-3.4.0:149> printf("sum = %f, product = %f, average = %f\n", summ, prd,
average);
sum = 15.000000, product = 120.000000, average = 3.000000
```

Add Or Subtract Two Vectors Element By Element

```
octave-3.4.0:150> x = [1 2 3];
octave-3.4.0:151> y = [4 5 6];
octave-3.4.0:152> x+y
ans =

   5   7   9
```

If you try to add a row vector to a column vector you'll get an error.

```
octave-3.4.0:153> x = [1 2 3];
octave-3.4.0:154> y = [4; 5; 6];
octave-3.4.0:155> x+y
error: operator +: nonconformant arguments (op1 is 1x3, op2 is 3x1)
```

This can be fixed using the transpose operator ' (single quote)

```
octave-3.4.0:155> x+y'
ans =

   5   7   9

octave-3.4.0:156> x'+y
ans =

   5
   7
   9
```

The two vectors must also be the same size.

```
octave-3.4.0:157> x = [1 2 3];
octave-3.4.0:158> y = [4 5 6 7];
octave-3.4.0:159> x+y
error: operator +: nonconformant arguments (op1 is 1x3, op2 is 1x4)
```

The Obvious Way to Multiply Two Vectors May Not Be What You Expect

If you try to multiply two vectors you'll get an error.

```
octave-3.4.0:159> x = [1 2 3];
octave-3.4.0:160> y = [4 5 6];
octave-3.4.0:161> x*y
error: operator *: nonconformant arguments (op1 is 1x3, op2 is 1x3)
```

If you transpose the one of the vectors you'll get an an answer, but it not be what you were expecting.

```
octave-3.4.0:161> x*y'
ans =  32
```

You got the dot product of the two vectors. The dot product is the sum of element by element multiplication and is rarely used in digital audio.

```
octave-3.4.0:166> x*y'
ans =  32
octave-3.4.0:167> dot(x,y)
ans =  32
octave-3.4.0:168> x(1)*y(1)+x(2)*y(2)+x(3)*y(3)
ans =  32
```

Multiply Two Vectors Element By Element

In audio calculations you normally want element by element multiplication of two vectors. These are rules you need to be aware of:

1. Both vectors must be row vectors or both vectors must be column vectors.
2. The two vectors have to be the same size.
3. Use the .* operator to multiply element by element.

```
octave-3.4.0:169> x .* y
ans =

    4   10   18
```

Row .* Column doesn't work

```
octave-3.4.0:177> xRowV = [1 2 3]
xRowV =

   1   2   3

octave-3.4.0:178> yColV = [4 5 6]'
yColV =

   4
   5
   6

octave-3.4.0:179> xRowV .* yColV
error: product: nonconformant arguments (op1 is 1x3, op2 is 3x1)
```

Element by element operations also work with these operators:

```
./

.^
```

You Can Access Any Element Of A Vector By Enclosing Its Index Position In Parentheses

Important: The first element of the list is index 1.

```
octave-3.4.0:179> x = [6 5 4 3 2 1];
octave-3.4.0:180> x(1) * x(4)
ans =  18
```

You Can Access A Range Of Elements Using The Colon Operator

```
octave-3.4.0:182> x(4:6) * pi
ans =

   9.4248   6.2832   3.1416
```

If You Try To Access An Element Outside The Vector's Range You'll Get An Error

```
octave-3.4.0:183> x(0)
error: subscript indices must be either positive integers or logicals
octave-3.4.0:183> x(7)
```

```
error: A(I): index out of bounds; value 7 out of bound 6
```

Array Indexes Must Be Integers

```
octave-3.4.0:183> x(4.2)
error: subscript indices must be either positive integers or logicals
```

Find The Minimum And Maximum Values In A Vector

```
octave-3.4.0:183> x = [ 4 57 3 -5.23 4 -6.1 17 36 -6 ];
octave-3.4.0:184> mx = max(x)
mx =  57
octave-3.4.0:185> mi = min(x)
mi = -6.1000
```

Find The Minimum And Maximum Values In A Vector And Their Index Position

```
octave-3.4.0:186> [mx,ix] = max(x)
mx =  57
ix =  2
octave-3.4.0:187> x(ix)
ans =  57
octave-3.4.0:188> [mi,ii] = min(x)
mi = -6.1000
ii =  6
octave-3.4.0:189> x(ii)
ans = -6.1000
```

Merge Two Vectors And Sort Them

```
octave-3.4.0:193> x = 10:-2:2;
octave-3.4.0:194> y = [ 3:2:11 -1 -3 ];
octave-3.4.0:195> z = [ x y ]
z =

   10    8    6    4    2    3    5    7    9   11   -1   -3

octave-3.4.0:196> sorted = sort(z)
sorted =

   -3   -1    2    3    4    5    6    7    8    9   10   11
```

Powers Of Two

Here are three ways to display the first 16 powers of 2.

```
% another way way
% for loop
for n=0:16
      2^n;
endfor
% nicely formatted way
for n=0:16
      printf( "2^%d = %d\n", n, 2^n );
endfor
```

Method 1 - uses the .^ operator for element by element raising to a power.

```
% powers of 2
n = [0:16];
% note the use of .^ for element by element power of 2
p = 2 .^ n;
p'
```

Method 2 - uses the power function **power(X,Y)** returns X raised to the Y power.

```
power(2,n)
```

Method 3 - uses a for loop to iterate over the second line 16 times.

```
% for loop
for n=0:16; 2^n endfor
```

Method 3 - uses printf for better formatting.

```
octave-3.4.0:210> for n=0:16; printf( "2^%d = %d\n", n, 2^n ); endfor
2^0 = 1
2^1 = 2
2^2 = 4
2^3 = 8
2^4 = 16
2^5 = 32
2^6 = 64
2^7 = 128
2^8 = 256
2^9 = 512
2^10 = 1024
2^11 = 2048
```

```
2^12 = 4096
2^13 = 8192
2^14 = 16384
2^15 = 32768
2^16 = 65536
```

Digital Audio Times And The Format Command

Create the sample times at the CD audio rate.

```
octave-3.4.0:225> % digital audio example
octave-3.4.0:225> % Sample Rate
octave-3.4.0:225> SR = 44100;
octave-3.4.0:226> % Sample Period
octave-3.4.0:226> T = 1/SR;
octave-3.4.0:227> % duration of 2.5 seconds
octave-3.4.0:227> dur = 2.5 * SR;
octave-3.4.0:228> % create the sample indexes
octave-3.4.0:228> n = 0:T:dur;
error: invalid range
octave-3.4.0:228> % display the sample times for the first two tenths of a
millisecond  octave-3.4.0:228> sampleTimes = [0:T:0.0002]
sampleTimes =

 Columns 1 through 6:

   0.0000e+00   2.2676e-05   4.5351e-05   6.8027e-05   9.0703e-05
1.1338e-04

 Columns 7 through 9:

   1.3605e-04   1.5873e-04   1.8141e-04
```

Times are displayed in scientific format, 2.267e-05 means $2.267 * 10^{-5}$.

Octave has a format command. Type help format to see the options. Here's two examples.

format short g

```
octave-3.4.0:233> format short g
octave-3.4.0:234> sampleTimes = [0:T:0.0003]
sampleTimes =

 Columns 1 through 6:

           0  2.2676e-05  4.5351e-05  6.8027e-05  9.0703e-05  0.00011338
 Columns 7 through 12:

  0.00013605  0.00015873  0.00018141  0.00020408  0.00022676  0.00024943

 Columns 13 and 14:

  0.00027211  0.00029478
```

format compact

```
octave-3.4.0:235> format compact
octave-3.4.0:236> sampleTimes = [0:T:0.0003]
sampleTimes =

 Columns 1 through 6:
           0  2.2676e-05  4.5351e-05  6.8027e-05  9.0703e-05  0.00011338
 Columns 7 through 12:
  0.00013605  0.00015873  0.00018141  0.00020408  0.00022676  0.00024943
 Columns 13 and 14:
  0.00027211  0.00029478
```

Type **format** without any parameters to return to the default setting.

Recalling The Past

The history command shows you what you've been doing. In this example 10 means display the 10 most recent lines. If I had not included the 10, history would have displayed 1351 previous commands.

```
octave-3.4.0:237> history 10
 1342 sampleTimes = [0:T:0.0002]
 1343 format short g
 1344 sampleTimes = [0:T:0.0002]
 1345 sampleTimes = [0:T:0.001]
 1346 sampleTimes = [0:T:0.0003]
 1347 format short g
 1348 sampleTimes = [0:T:0.0003]
 1349 format compact
 1350 sampleTimes = [0:T:0.0003]
 1351 history 10
```

Saving Your Work

Octave has a feature for saving your work to a file using the diary command.

diary 'filename' (toggles diary on and off)

Turn the diary on and save to a file m208diary.txt in the current working directory.

```
octave-3.4.0:238> diary 'm208diary.txt';
```