Octave

Thanh Ha Do, VNU University of Science

SEAMS SCHOOL ON

MATHEMATICAL MODELLING
IN BIOLOGY

March 08-15, 2017

Contents

- Overview
- Start, quit, getting help
- Variables and data types
- Matrices
- Plotting

Contents

Overview

Start, quit, getting help

Variables and data types

Matrices

Plotting

Overview

Octave is the "open-source **Matlab**"

- It is for free (GPL license)
  - www.octave.org
  - www.mathworks.com
- There are minor differences in syntax

Octave and **Matlab** are high-level languages and mathematical programming environments for

- Visualization
- Programming, algorithm development, etc.
- Scientific computing: linear algebra, optimization, statistic, signal processing, etc.

Contents

Overview

Start, quit, getting help

Variables and data types

Matrices

Plotting

Start, Quick, Getting Help

Start Octave: type the shell command octave or whatever your OS needs

Interrupt Octave: by typing `Ctrl-C`

Quit Octave: type `quit` or `exit`

Get **help**: type `help` or `doc`

- Get **help** on a **specific command**: type `help command`
  - `help size, help plot, help figure, help inv, ...`
- To get **help** on the **help system**, type `help help`
- Type `q` to **exit help** mode

Contents

Overview

Start, quit, getting help

Variables and data types

Matrices

Plotting

Variables and data types

In Octave/Matlab almost **everything** is a **matrix**

Main matrix classes

- **Strings**: matrices of characters
- **Structures**: matrices of named fields for data of varying types and sizes
- **Logical**: matrices of boolean 0/1-values

Not treated in this tutorial

- Cells (like structures)
- Function handles (pointer to functions)

Variable and data types

vector or **arrays**?

A matrix with one column or row

Scalars?

A matrix of dimension 1×1

Intergers?

A double

Characters

A string of size 1

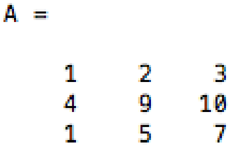
Variable and data types (con't)

Creating a Matrix

Simply type:

```
» A = [1, 2, 3; 4, 9, 10; 1, 5, 7]
```

Octave will respond with a matrix in pretty-print:



```
A =

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

More on matrices will introduce further down this tutorial

Variable and data types (con’t)

Creating a Character String

Simply type: `» str = ’Hello World’`
Opposed to Matlab, Octave can also deal with double quotes.
For compatibility reasons: always use **single quotes**

Creating a Structure

Type for instance

```
» data.id = 3
» data.timestamp = 1256.235
» data.name = ’School’
```

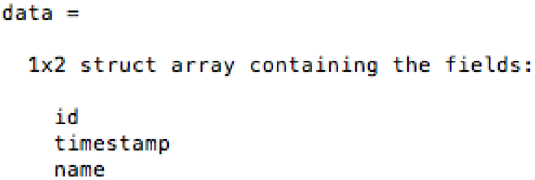
Variable and data types (con't)

Creating a Vector of Structures

A new measurement has arrived. Extend struct by:

```
» data(2).id = 4
» data(2).timestamp = 1268.45879
» data(2).name = 'Department'
```

Octave will respond with ....



```
data =

  1x2 struct array containing the fields:

    id
    timestamp
    name
```

Variable and data types (con't)

Display Variables

Simply type its name

```
» a = 4
```

Suppress Output

Add a semicolon

```
» a;
» sin(pi)
```

Applies also to function calls

Variable and data types (con't)

Variable have **no permanent type.** Octave/Matlab are weakly typed languages

`s = 3` followed by `s = ”octave"` is fine

Use `help` or `who` (or the more detailed `whos`) to **list** the **currently defined variables.** Example:

Variable and data types (con't)

Numerical Precision

Variables are stored as double precision numbers in IEEE floating point format

- `realmin`: Smallest positive floating point number: 2.23e-308
- `realmax`: Largest positive floating point number: 1.80e+308
- `eps`: Relative precision: 2.22e-16

These keywords are **reserved** and can be used in your code

Variable and data types (con't)

Control Display of Float Variables

`format short`	Fixed point format with 5 digits
`format long`	Fixed point format with 15 digits
`format short e`	Floating point format, 5 digits
`format long e`	Floating point format, 15 digits
`format short g`	Best of fixed or floating point with 5 digits
`format long g`	Best of fixed or floating point with 15 digits

See `help format` for more information

Variable and data types (con't)

Talking about Float Variables...

`ceil(x)`	Round to smallest interger not less than x
`floor(x)`	Round to largest integer not greater than x
`round(x)`	Round towards nearest integer
`fix(x)`	Round towards zero

If x is a matrix **matrix**, the functions are applied to **each element** of x

Contents

Overview

Start, quit, getting help

Variables and data types

Matrices

Plotting

Matrices

Creating a Matrix

Simply type:

```
» A = [1, 2, 3; 4, 9, 10; 1, 5, 7]
```

To delimit **columns**, use comma or space

To delimit **rows**, use semicolon

The following expressions are **equivalent**

```
» A = [1 2 3; 4 9 10; 1 5 7]
» A = [1, 2, 3; 4, 9, 10; 1, 5, 7]
```

Matrices

Creating a Matrix

Alternative Example:

```
» phi = pi/3
» [cos(phi) -sin(phi); sin(phi) cos(phi)]
```

Matrices

Creating a Matrix from Matrices

```
» A = [1 2 3; 4 9 10; 1 5 7] ; B = [33; 33; 33]
```

Column-wise

```
» C = [A B]
```

Row-wise

```
» D = [A; [33 33 33]]
```

Matrices

Indexing

Always "row before column"!

`aij = A(i, j)`	get an element
`r = A(i, :)`	get a row
`c = A(:,j)`	get a column
`B = A(i:k, j:l)`	get a sub-matrix

Useful indexing command end:

```
» A = [1 2 3; 4 9 10; 1 5 7]
» v = A(2:end; 2:end)
```

Matrices

The two meaning of colon ':'

» `A(3,:), B(:,1)`

- **Wildcard** to select entire matrix **row** or **column**
- **Defines a range** in expression like

```
indices = 1:5    Returns row vector 1, 2, 3, 4, 5
steps = 1:3:61   Returns row vector 1, 4, 7, ..., 61
t = 0:0.01:1     Returns vector 0, 0.01,0.02, ..., 1
```

- **Useful command** to define ranges: `linspace`

Matrices

Assigning a Row/Column

All referenced elements are set to the scalar value

```
» A = [1 2 3; 4 9 10; 1 5 7]
» A(3,:) = -2
```

Adding a Row/Column

If the referenced row/columns does not exist, it's added

```
» A(5,:) = -2
```

Result ??

Matrices

Deleting a Row/Column

Assigning an empty matrix[] deletes the referenced rows or columns

Examples:

```
» A(3,:) = []
» A(1:1:3,:) = []
```

Matrices

Get Size

`nr = size(A, 1)`	Get number of rows of A
`nc = size(A, 2)`	Get number of columns of A
`[nr nc] = size(A)`	Get both (remember order)
`l = length(A)`	Get whatever is bigger
`numel(A)`	Get number of elements
`isempty(A)`	Check if A is empty matrix[]

Octave only:

`nr = rows(A)`	Get number of rows of A
`nc = columns(A)`	Get number of columns of A

Matrices

Matrix Operations With x being a column vector

`B = 3*A`	Multiply by scalar
`C = A*B + X - D`	Add and multiply
`B = A'`	Transpose A
`B = inv(A)`	Invert A
`s = v'*Q*v`	Mix vectors and matrices
`d = det(A)`	Determinant of A
`[v lambda] = eig(A)`	Eigenvalue decomposition
`[U S V] = svd(A)`	Singular value decomposition

Matrices

Vector Operations
With x being a column vector

`s = x'*x`	Inner product, result is a scalar
`X = x*x'`	Outer product, result is a matrix
`e = x*x`	Gives an error

Element-Wise Operations With x being a column vector

`s = x.+x`	Element-wise addition
`p = x.*x'`	Element-wise multiplication
`q = x./x`	Element-wise division
`e = x.^3`	Element-wise power operator

Matrices

Useful Vector Functions

`sum(v)`	Compute sum of elements of v
`cumsum(v)`	Compute cumulative sums of elements of v (returns a vector)
`prod(v)`	Compute product of elements of v
`cumprod(v)`	Compute cumulative products of elements of v (returns a vector)
`diff(v)`	Compute difference of subsequent elements [v(2) - v(1)v(3)-v(2)...]
`mean(v)`	Mean value of elements in v
`std(v)`	Standard deviation of elements

Matrices

Useful Vector Functions

`min(v)`	Return smallest element in v
`max(v)`	Return largest element in v
`sort(v,’ascend’)`	Sort in ascending order
`sort(v, ’descend’)`	Sort in descending order
`find(v)`	Find indices of non-zero elements Great in combination with vectorization Example: `ivec = find(datavec == 5)`

Matrices

Special Matrices

`A = zeros(m, n)`	Zero matrix of size $m \times n$ (Often used for preallocation)
`B = ones(m, n)`	Matrix of size $m \times n$ with all 1's
`I = eye(n)`	Identity matrix of size n
`D = diag([a b c])`	Diagonal matrix of size 3×3 with a, b, c in the main diagonal

Matrices

Random Matrices and Vectors

`R = rand(m,n)`	Matrix with $m \times n$ uniformly distributed random numbers from interval [0..1]
`N = randn(m,n)`	Matrix with $m \times n$ normally distributed random numbers with zero mean, unit variance
`v = randperm(n)`	Row vector with a random permutation of the numbers 1 to n

Matrices

Multi-Dimensional Matrices Matrices can have more than two dimensions

Create a 3-dimensional matrix: e.g.,

```
    » A = ones(2, 5, 2)
» A(:,:,1) ?
```

Matrices

Multi-Dimensional Matrices

All operations to create, index, add, assign, delete and get size apply in the same fashion

Examples:

```
» [m n l] = size(A)
» A = ones(m, n, l)
» m = min(min(min(A)))
» aijk = A(i, j, k)
» A(:, :, 2) = -3
```

Matrices

Matrices Massage

Matrix operations that have no mathematical meaning. Useful for manipulating data with is organized in matrices

`reshape(A, m,n)`

Change size of matrix A to have dimension $m \times n$. An error results of A does not have $m \times n$ elements

`circshift(A,[m n]]`

Shift elements of A m times in row dimension and m times in column dimension. Has no mathematical meaning

`shiftdim(A, n)`

Shift the dimension of A by n. **Generalizes transpose** for multi-dimensional matrices

Matrices

Matrices Massage

Matrix operations that have no mathematical meaning. Useful for manipulating data with is organized in matrices

`fliplr(A)` **Reverses the order** of columns of matrix A in left/right-direction. Rows are not changed

`flipud(A)` **Reverses the order** of rows of matrix A in up/down-direction. Columns are not changed

`flipdim(A, dim)` **Flip** matrix A along **dimension dim.** Typically. for multi-dimensional matrices

`rot90(A)` **90 degree counterclockwise rotation** of matrix A. This is **not** the transpose of A

Matrices

Matrices Massage Example

Let P = $[x1; y1; x2; y2; \ldots]$ be a $2n \times 1$ column vector of n pairs (x, y). Make it a column vector of (x, y, theta) tuples with all theta being pi/2

Make P it a $2 \times n$ matrix

```
» P = reshape(P, 2, numel(P)/2)
```

Add a third row, assign pi/2

```
» P(3,:) = pi/2
```

Reshape it to be a $3n \times 1$ column vector

```
» P = reshape(P, numel(P),1)
```

String

Most Often Used Commands

`strcat`	Concatenate strings
`int2str`	Convert integer to a string
`num2str`	Convert floating point numbers to a string
`sprintf`	Write formatted data to a string Same as C/C++ fprintf for strings

Example

```
» s = strcat('At step ',int2str(k),', p = ',num2str(p,4))
```

Given that strings are matrices of characters, this is equivalent to

```
» s = ['At step ' int2str(k) ', p = ' num2str(p,4)]
```

String

Octave/Matlab has virtually all common string and sparsing functions

You can encouraged to browse through the list of commands or simply `help command`

Some commands:

```
strcmp, strncmp strmatch, char, ischar,
findstr, strfind, str2double, str2num,
num2str, strvcat, strtrim, strtok, upper,
lower, ...
```

Contents

Overview

Start, quit, getting help

Variables and data types

Matrices

Plotting

Plotting

Plotting in 2D

- Display x, y plot : `plot(x, cos(x))`
  - Creates automatically a figure window. **Octave uses gnuplot to handle graphics.**
- Create figure window ’n’: `figure(n)`
  - If the figure window **already exists**, brings it into the forground (=makes it the current figure)
- Create new figure window with identifier incremented by 1: `figure`

Plotting

Several Plots

Series of x,y-pairs: `plot(x1, y1, x2, y2, ...)`

e.g. `plot(x, cos(x), x, sin(x))`

Add **legend** to plot: `legend`

```
legend (’cos(x)’, ’sin(x)’)
```

Alternatively, `hold on` does the same job:

```
» hold on; plot(x, cos(x));
» plot(x, sin(x));
» plot(x, x.ˆ2);
```

Plotting

Frequent Commands

`clf`	Create figure
`hold on`	Hold axes. Do not replace plot with new plot, superimpose plots
`grid on`	Add grid lines
`grid off`	Remove grid lines
`title('My Plot')`	Set title of figure window
`xlabel('time')`	Set label of x-axis
`ylabel('prob')`	Set label of y-axis

Plotting

Controlling Axes

`axis equal`	Set equal scales for x-/y-axes
`axis square`	Force a square aspect ratio
`axis tight`	Set axes to the limits of the data
`a = axis`	Return current axis limits [xmin xmax ymin ymax]
`axis([-1 1 2.5 5])`	Set axis limits (freeze axes)
`axis off`	Turn off tic marks
`box on`	Adds a box to the current axes
`box off`	Removes box

Plotting

Controlling Plot Styles

In `plot(x, cos(x),’r+’ )` the format expression `’r+’` means **red cross**

There are a number of line styles and colors, see `help plot`

Example:

```
» x = linspace(0,2*pi,100);
» plot(x,cos(x),’r+’,x,sin(x),’bx’);
```

more on **plotExample.m**

Plotting

Exporting Figures

`print -deps picBW.eps`	Export B/W .eps file
`print -depsc picC.eps`	Export color .eps file
`print -djpeg -r80 myPic.jpg`	Export .png in 80 ppi
`print -dpng -r100 myPic.png`	Export .png in 100 ppi

See `help print` for more devices including specialized ones for Latex

`print` can also be **called as a function**

Then it takes arguments and options as a comma-separated list

```
print('-dpng', '-r100', 'myPic.png')
```

Plotting

This tutorial cannot cover the large variety of graphics commands in Octave/Matlab

You are encouraged to browse through the list of commands or simply type `help command`

Some commands:

```
hist, bar, pie, area, fill, contour,
quiver, scatter, compass, rose, semilogx,
loglog, stem, stairs, image, images ...
```

Plotting

Plotting in 3D

`plot3`	Plot lines and points in 3D
`mesh`	3D mesh surface plot
`surf`	3D colored surface plot

Most 2D plot commands have a **3D sibling.** Check out, for example,
`bar3, pie3, fill3, contour3, quiver3, scatter3, stem3`
let see some **examples...**