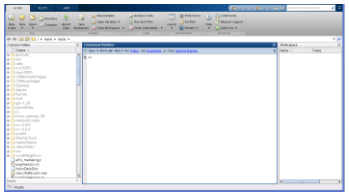
CISC 3250
Systems Neuroscience

Matlab



Professor Daniel Leeds
dleeds@fordham.edu
JMH 328A

Access to Matlab

Laptop/home computer:

- Mathworks link on our course website
- Student license for $49

Lab computer:

- Open terminal
- Type: `matlab`

2

Variables

Variables store information

- Letters

```
neuronType='purkinje';
```

- Single Number

```
numberOfDendrites=1000;
```

- Group of numbers, in [ ] brackets

```
potentials=[-65 -64 -63.9 -62.8 -61.6];
```

3

Commands

Symbols and keywords cause actions

- `b=2` *creates variable b with value 2*
- `d=b+5` *creates variable d with value computed by adding 5 to value of b*
- `exit` *closes program*

4

= operation

= assigns value on right to variable on left

- `b=5` **valid**
- `5 = b` **invalid**

5

Variable names

- A variable name is any valid identifier
  - Starts with a letter, contains letters, digits, and underscores (_) only
  - Cannot begin with a digit
  - Case sensitive: `username≠userName≠UserName`

6

Be careful with variable names

- `NumSpikes=10`

Variables are case-sensitive

- `numspikes-5` **error, did not capitalize N and S**
- `NumSpike-5` **error, forgot letter s at end**

7

Defining a vector

Vector is a list of numbers

- `b=[42,35,68,-3]`
- `c=[-18 12 14]`

Vector denoted by [ ] braces

Elements separated by commas , or blank spaces

8

Plotting data

```
plot(spikeRates)
```

- Multiple plots at once

```
figure              plot(vector1,'b')
plot(vector1)       hold on
figure              plot(vector2,'r')
plot(vector2)
```

9

Semi-colons

- New line or ; establishes end-of-command

```
figure; plot(vector1); figure;
plot(vector2)
```

- ; suppresses output of computation result to screen

```
b=10-8;
```

10

Counting in Matlab

`a:b` creates a vector `[a a+1 … b-1 b]`

- `3:6 -> [3 4 5 6]`

`a:k:b` creates a vector `[a a+k a+2k … b]`

- `3:4:15 -> [3 7 11 15]`

11

Accessing vector elements

```
a=[2.2 1.4 -5 3.5 -7.8];
```

- `name` accesses full vector

```
a
```

- `name(index)` accesses single element

```
a(4)
```
returns *3.5*

- `name(index1:index2)` accesses set of elements

```
a(2:4)
```
returns *[1.4 -5 3.5]*

- `name(end)` accesses final element

12

Vector indexing

Assume we have a recording of spike rates for 100 seconds, recorded over non-overlapping 100 ms windows : vector `SpikeRate`

- `SpikeRate(1)` contains rate from 1-100ms
- `SpikeRate(2)` contains rate from 101-200ms

How do we see rates for 4-6s (4001-6000ms)

- `SpikeRate(401:600)`

13

Standard arithmetic

Operators

- Addition: `5 + 2` evaluates to `7`
- Subtraction: `5 - 2` evaluates to `3`
- Multiplication: `5 * 2` evaluates to `10`
- Division: `4 / 2` evaluates to `2`
- Exponent: `5 ^ 2` evaluates to `25`

14

Data

Data can be read from files

- `load('classExample.mat');`
- `save('classExample2.mat','c','d');`

List the loaded variables

- `who`
- `whos`

Study the variable

- `size(spike_record)`
- `plot(spike_record)`

15

Functions

```
c=[0 3 -2 4];
```

Data are analyzed through functions

`function_name(input_variable)`

- `sum(c) -> 5`
- `min(c) ->`
- `max(c) ->`

- `plot(spike_record)`

16

Matrices: rows and columns

```
B=[2.2 1.4; -5 3.5; -7.8 4.3];
```

$$\begin{bmatrix} 2.2 & 1.4 \\ -5 & 3.5 \\ -7.8 & 4.3 \end{bmatrix}$$

- Spaces/commas separate columns
- Semi-colons (;) separate rows
- `name(row,col)` accesses single element

`B(2,1)` *returns -5*

17

Matrix indexing

Assume we have a 10x500 matrix of spike patterns for 10 neurons `spikeMat`

- `spikeMat(1,:)` contains spikes for neuron 1
- `spikeMat(4,:)` contains spikes for neuron 4
- `spikeMat(:,100)` contains spikes for all neurons at time t=100

In general:

- `name(:,col)` accesses all elements in column

`B(:,2)` `returns [1.4; 3.5; 4.3]`

- `name(:)` vector of all elements in name

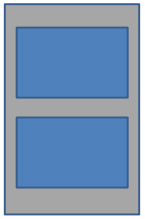
18

Multiple plots

figure -> opens new plotting window

subplot(r,c,i) -> creates grid of plots with

- r rows
- c columns
- fill in position i



```
subplot(2,1,1); plot(vec1);
subplot(2,1,2); plot(vec2);
```

19

Taking repeated action

- Assign `k` each value counting up from `start` value to `finish` value, repeating listed actions for each new value of `k`

```
for k=start:finish,
  action1 to repeat
  action2 to repeat
  action3 to repeat
end;
```

Example from class:

```
for k=1:10,
  subplot(5,2,k), plot(spikeMat(k,:));
end;
```

Finding desired values

`find(vector<number)` `find(c<2)`

Return indices in `vector` that are less than `number`

Comparisons

- `d<2, d>2` strict inequality
- `d<=2, d>=2` semi-inequality
- `d==2` equality

Logic combinations

- `d>5 & d<8` the AND operation
- `d<5 | d>8` the OR operation

21

Saving graphics results

- `print -dpng filename.png`
- `print -djpg filename.jpg`

22

Vector arithmetic

- Vector is list of numbers in between [ ]
- Can replace one of operands with a vector

`2+[3 4 1]` *yields* `[5 6 3]`

- Can place results into new variable

`Variable_Name=number*vector;`

- Both operands can be vectors, but special rules apply

23