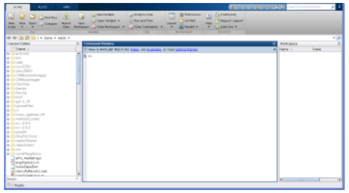
CISC 3250
Systems Neuroscience

Matlab



Professor Daniel Leeds
dleeds@fordham.edu
JMH 332

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 (d now has value 7)*
- `exit` *closes program*

= operation

= assigns value on right to variable on left

- `b=5` **valid** (b has value 5)
- `5 = b` **invalid**

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: `rateV4`≠`RateV4`≠`RATEv4`

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`

5

Example sequence of code

```
EL = -65;
RI = 20;
Vtarget = EL+RI;
```

What value does Vtarget have?

Answer on next slide

6

Example sequence of code (+answer)

```
EL = -65;
RI = 20;
Vtarget = EL+RI;
```

What value does Vtarget have?

Vtarget has value -65+20 = **-45**

7

Incorrect code versions:

Three of these example code are wrong, one is right. Explain why each of the three is wrong.

```
EL = -65;
RI = 20;
EL+RI=Vtarget;
```

```
Vtarget=EL+RI;
RI=-65;
EL=20;
```

```
EL = -65mV;
RI = 20mV;
Vtarget=EL+RI;
```

```
RI=20;
El=-65;
Vtarg=El+RI;
```

Answers on next slide

8

Incorrect code versions:

```
EL = -65;
RI = 20;
EL+RI=Vtarget;
```

Vtarget on wrong side of = (assign operation)

```
Vtarget=EL+RI;
RI=-65;
EL=20;
```

EL, RI not defined before = (assign command) used

```
EL = -65mV;
RI = 20mV;
Vtarget=EL+RI;
```

Matlab confused by mV notation

```
RI=20;
El=-65;
Vtarg=El+RI;
```

Correct!

9

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**

10

Defining/accessing a vector

Vector is a list of numbers

(separated by spaces or by commas)

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

Vector denoted by [ ] braces

Elements separated by commas , or blank spaces

Retrieve ith element of `vec` with `vec(i)`

`b(3)=` `c(end)`

11

Functions

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

Data are analyzed through functions

`function_name(input_variable)`

- `sum(c) -> 5`
- `min(c) ->` **-2**
- `max(c) ->` **4**

- `plot(spike_record)`

13

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)`

14

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]`

15

Accessing vector elements

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

- `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

16

spikeExample

- From our course website
- Contains variable `spikes` – 1 neuron, 500 ms
- 0 if no spike, 1 if spike

- Compute counts for each 100ms window:

```
count(1)=sum(spikes(1:100));
count(2)=sum(spikes(101:200));
count(3)=sum(spikes(201:300));
count(4)=sum(spikes(301:400));
count(5)=sum(spikes(401:500));
rate=count/0.1; % spikes/second
```

18

Loop

Repeating similar action

```
for i = 1:4
  disp(i);
end;
```

Output

```
1
2
3
4
```

Basic syntax

```
for var = VarValues
  actions-to-repeat
end
```


spikeExample – rate loop

- Compute count for each 100ms window:

```
count(1)=sum(spikes(1:100));
count(2)=sum(spikes(101:200));
count(3)=sum(spikes(201:300));
count(4)=sum(spikes(301:400));
count(5)=sum(spikes(401:500));
```

- Compute with for loop:

```
for i=1:5
  count(i)=sum(spikes(100*(i-1)+1:100*i));
end;
rate = count/0.1;
```


Semi-colons

; suppresses output of computation result to screen

```
a=10-8
    a = 2
```

Printed to screen

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


More loop practice: computing compound interest

```
bVec(1)=13;
for t=2:50,
  % 4 percent compound interest
  bVec(t)=bVec*1.04;
end;
plot(bVec)
```

Similar to dv/dt update rule, balance at time t depends on balance at time t-1

More loop practice: implement leaky-integrate-and-fire

```
v(1)=-65; EL=-65;
tau=0.05; step=0.001;
RI=20; % presume constant input
for t=2:1000,
  deltaV=???;
  volt(?)=volt(?)+deltaV*step/tau;
end;
plot(volt)
```

Try replacing the ?? parts and plotting volt!

Does not implement auto-reset

Logic

Conditional behavior based on variable value

```
if x > 5
  y=2;
else
  y=5;
end;
```

Basic syntax

```
if condition
  actions-if-true
else
  actions-if-false
end
```


Logic

Conditional behavior based on variable value

```
if x > 5
  y=2;
else
  y=5;
end;
```

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