Matlab

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

\[ = \text{ operation} \]
- \( = \) assigns value on right to variable on left

- \( b = 5 \) valid
- \( 5 = b \) invalid

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
Standard arithmetic

Operators

• Addition: $5 + 2$ evaluates to 7
• Subtraction: $5 - 2$ evaluates to 3
• Multiplication: $5 \times 2$ evaluates to 10
• Division: $4 \div 2$ evaluates to 2
• Exponent: $5^2$ evaluates to 25

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

Logic

Conditional behavior based on variable value

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

Basic syntax

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

Comparisons

• $d<2$, $d>2$  strict inequality
• $d\leq2$, $d\geq2$  semi-inequality
• $d==2$  equality

Logic combinations

• $d>5$ & $d<8$  the AND operation
• $d<5$ | $d>8$  the OR operation
Loop

Repeating similar action
for \( i = 1:4 \)
\[
\begin{align*}
& \text{disp}(i); \\
& \text{end;}
\end{align*}
\]

Basic syntax
for \( \text{var} = \text{VarValues} \)
\[
\begin{align*}
& \text{actions-to-repeat} \\
& \text{end}
\end{align*}
\]

Output
1
2
3
4

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

Counting in Matlab

\( a:b \) creates a vector \([ a \ a+1 \ b \] \)
- \( 3:6 \) \( \rightarrow [3 \ 4 \ 5 \ 6] \)

\( a:k:b \) creates a vector \([ a \ a+k \ a+2k \ b \] \)
- \( 3:4:15 \) \( \rightarrow [3 \ 7 \ 11 \ 15] \)

Accessing vector elements

\( a = [2.2 \ 1.4 \ -5 \ 3.5 \ -7.8]; \)
- \( \text{name(index)} \) accesses single element
  \( a(4) \) returns 3.5
- \( \text{name(index1:index2)} \) accesses set of elements
  \( a(2:4) \) returns \([1.4 \ -5 \ 3.5]\)
- \( \text{name(end)} \) accesses final element
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

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

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)
\texttt{SpikeRate(41:60)}

Data

Data can be read from files
- \texttt{load('classExample.mat');}
- \texttt{save('classExample2.mat','c','d');}

List the loaded variables
- \texttt{who}
- \texttt{whos}

Study the variable
- \texttt{size(spike_record)}
- \texttt{plot(spike_record)}

Semi-colons

; suppresses output of computation result to screen
\texttt{a=10-8}
\hspace{1cm} a = 2 \hspace{1cm} \text{Printed to screen}
\texttt{b=10-8;}
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)`

spikeExample

```
• From our course website
• Contains variable spikes – 10 neurons, 500 ms
• 0 if no spike, 1 if spike
```

- Compute rates for each 100ms window:
  - `rate(1)=sum(spikes(6,1:100));`
  - `rate(2)=sum(spikes(6,101:200));`
  - `rate(3)=sum(spikes(6,201:300));`
  - `rate(4)=sum(spikes(6,301:400));`
  - `rate(5)=sum(spikes(6,401:500));`

spikeExample – rate loop

```
• Compute rates for each 100ms window:
  rate(1)=sum(spikes(6,1:100));
  rate(2)=sum(spikes(6,101:200));
  rate(3)=sum(spikes(6,201:300));
  rate(4)=sum(spikes(6,301:400));
  rate(5)=sum(spikes(6,401:500));
```

```
• Compute with for loop:
  for i=1:5
      rate(i)=sum(spikes(6,100*(i-1)+(1:100)));
  end;
```

Plotting data

```
plot([4,5,-2,8])
```

```
• From course site:
  spikePlot(spikes)
```
Matrices: rows and columns

\[ B = \begin{bmatrix} 2.2 & 1.4; & -5 & 3.5; & -7.8 & 4.3; \end{bmatrix}; \]

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

\[ B(2,1) \quad \text{returns} \quad -5 \]