Introduction to Computer and Programming

Chapter 2: MATLAB scripting

Manuel

Fall 2018

Outline

Running MATLAB

Two modes:

- Desktop: graphical user interface
- Terminal: allows remote access, no mouse support

View in desktop mode:

- Command history
- Workspace
- Command window
- Help

File location: current directory or a directory listed in the path

Basic use

MATLAB as a calculator:

- Operation: `1+2` vs. `1+2;`
- Variables: start with a letter, case sensitive
  e.g. `a=1+2; A=3+2; a123_=4+5;`
- Comments: ignore everything after a `%`
- Separate two commands on a same line: `cmd1, cmd2`
- Split a line over two lines: keep reading on next line after `...`
  e.g. long ...
  line

Simple operations

More MATLAB operations:

- Addition: $+$
- Subtraction: $-$
- Multiplication: $*$
- Power: ^
- (Right) division: $/$
- Left division: $\backslash$
- Order of evaluation: $()$

- pi $= \pi$
- i $= \sqrt{-1}$
- j $= \sqrt{-1}$
- Inf $=$ Infinity
- NaN: Not a Number

Density of the Sun

MATLAB code to input in the workspace window:

```
r=1.496*10^11; c=4.379*10^9; G=6.674*10^-11;
T=365*24*3600;
V=4*pi/3*(c/(2*pi))^3;
M=4*pi^2*r^3/(G*T^2);
M/V
```

Density of the Sun

MATLAB code to input in the workspace window:

```matlab
r=1.496*10^11; c=4.379*10^9; G=6.674*10^-11;
T=365*24*3600;
V=4*pi/3*(c/(2*pi))^3;
M=4*pi^2*r^3/(G*T^2);
M/V
```

Questions.

- How are variables named and used?
- Could the code be shorter?

M-File

MATLAB script:

- Write the code in a file and load it
- Variables are added to the workspace
- To avoid variable conflicts use: `clear`, `clear all`, `clc`
- Add *cell breaks* to debug the code

M-File

MATLAB script:

- Write the code in a file and load it
- Variables are added to the workspace
- To avoid variable conflicts use: `clear`, `clear all`, `clc`
- Add *cell breaks* to debug the code

Exercise.
Write a script which prompts the user for two numbers, stores their sum in a variable, and displays the result.

M-File

MATLAB script:

- Write the code in a file and load it
- Variables are added to the workspace
- To avoid variable conflicts use: `clear`, `clear all`, `clc`
- Add *cell breaks* to debug the code

Exercise.
Write a script which prompts the user for two numbers, stores their sum in a variable, and displays the result.

```
clear all, clc;
number1=input('Input a number: ');
number2=input('Input a number: ');
numbers=number1+number2;
disp(numbers);
```

Arrays and MATLAB

Array

Arrangement of quantities in rows and columns

Arrays and MATLAB

Array
Arrangement of quantities in rows and columns

$$\downarrow$$

Matrix
Two-dimensional numeric array

Arrays and MATLAB

Array
Arrangement of quantities in rows and columns

$$\downarrow$$

Matrix
Two-dimensional numeric array

$$\downarrow$$

MATLAB
MATrix LABoratory

Arrays and MATLAB

Array
Arrangement of quantities in rows and columns

$\downarrow$

Matrix
Two-dimensional numeric array

$\downarrow$

MATLAB
MATrix LABoratory

$\Downarrow$

Arrays are the **most important** concept to understand

Generating arrays and matrices

Creating arrays and matrices:

- Sequence of numbers: `a:b` or `a:b:c`
- Concatenate (join) elements: `[ ]`

Generating arrays and matrices

Creating arrays and matrices:

- Sequence of numbers: `a:b` or `a:b:c`
- Concatenate (join) elements: `[ ]`
- 1-dimensional array: `[a:b]` or `[a:b:c]`
- 2-dimensional array: `[a b c; d e f;]`

Generating arrays and matrices

Creating arrays and matrices:

- Sequence of numbers: `a:b` or `a:b:c`
- Concatenate (join) elements: `[ ]`
- 1-dimensional array: `[a:b]` or `[a:b:c]`
- 2-dimensional array: `[a b c; d e f;]`
- *n* elements from $[a, b]$: `linspace(a, b, n)`
- `zeros(a,b)`
- `ones(a,b)`

Dealing with matrices

Explain the result of the following commands:

```
clear all
a=magic(5)
a=[a;a+2], pause
a(:,3)=[]
a(:,3)=5
a(7,3), pause
whos a
a=reshape(a,5,8)
a', pause
sum(a)
sum(a(:,1))
sum(a(1,:))
```

Array vs. Matrix

Arrays

- Element by element
- .*
- ./
- .\
- .^

Matrices

- Complex conjugate transpose: '
- Nonconjugate transpose: *
- det
- inv
- eig

Basic operations

Explain the result of the following commands:

```
A = [2 7 9 7 ; 3 1 5 6 ; 8 1 2 5]
A(:,[1 4]), pause
A([2 3],[3 1]), pause
reshape(A,2,6), pause
A(:), pause
flipud(A), pause
fliplr(A), pause
[A A(:,end)], pause
A(1:3,:), pause
[A ; A(1:2,:)], pause
sum(A),pause
sum(A'), pause
sum(A,2), pause
[ [ A ; sum(A) ] [ sum(A,2) ; sum(A(:)) ] ], pause
A.'
```

Accessing elements in a matrix

Given a matrix, elements can be accessed by:

- Coordinates: using their (row,column) position
- Indices: using a single number representing their position; the top left element has index 1 and the bottom right "number of elements"

Accessing elements in a matrix

Given a matrix, elements can be accessed by:

- Coordinates: using their (row,column) position
- Indices: using a single number representing their position; the top left element has index 1 and the bottom right “number of elements”

Example.
Explain the following commands:

```
A=magic(5)
A(3,2)
A(6)
numel(A)
```

Outline

The `if` statement

If it rains, then I take an umbrella

The `if` statement

If it rains, then I take an umbrella

Structure in MATLAB:

```
if expression1
  statements1
elseif expression2
  statements2
  else
  statements
end
```

Boolean logic

Boolean logic: introduced by George Boole around mid 1800s

Truth table for the common operations:

A	B	$A \wedge B$	$A \vee B$	$A \oplus B$
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Boolean logic

Boolean logic: introduced by George Boole around mid 1800s

Truth table for the common operations:

A	B	$A \wedge B$	$A \vee B$	$A \oplus B$
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Run instructions based on the truth value of a given expression

Relational operators

Comparative operators:

- $<$ less than
- $<=$ less than or equal to
- $>$ greater than
- $>=$ greater than or equal to
- $==$ equal to
- $\sim=$ not equal to

Relational operators

Comparative operators:

- $<$ less than
- $<=$ less than or equal to
- $>$ greater than
- $>=$ greater than or equal to
- $==$ equal to
- $\sim=$ not equal to

Logical operators:

- & and
- | or
- $\sim$ not
- $\texttt{xor}(\cdot,\cdot)$ exclusive or

Relational operators

Comparative operators:

- $<$ less than
- $<=$ less than or equal to
- $>$ greater than
- $>=$ greater than or equal to
- $==$ equal to
- $\sim=$ not equal to

Logical operators:

- & and
- | or
- $\sim$ not
- $\texttt{xor}(\cdot,\cdot)$ exclusive or

Short-circuit operators:

- `A && B` evaluates expression `B` only if `A` is `True`
- `A || B` evaluates expression `B` only if `A` is `False`

Simple application script

Example.

```
exist('./file') & load('./file')
exist('./file') && load('./file')
k=input('Press a key: ','s');
if k>='0' && k<='9'
  disp('Digit')
else
  disp('Not a digit')
end
```

Simple application script

Example.

```
1 exist('./file') & load('./file')
2 exist('./file') && load('./file')
3 k=input('Press a key: ','s');
4 if k>='0' && k<='9'
5   disp('Digit')
6 else
7   disp('Not a digit')
8 end
```

Questions.

- What are those commands doing?
- How to request some input form the user?
- What is `'s'` on line 1?

The `switch` statement

When it rains, I take an umbrella; When it's sunny I take a hat.

The switch statement

When it rains, I take an umbrella; When it's sunny I take a hat.

Structure in MATLAB:

```
switch variable
  case value1
    statements1
  case value2
    statements2
  otherwise
    statements
end
```

Note: the variable is expected to be a scalar or a string

Example

Write a script which prompts the user for a digit, displays 0 on a 0, < 5 if it is between 1 and 4, and ≥ 5 if it is larger or equal to 5.

```
i=input('Input a digit: ');
switch i
  case 0
    disp('0')
  case {1,2,3,4}
    disp('<5')
  otherwise
    disp('>=5')
end
```

Questions.

- How is the code aligned?
- Why is `input` used without the `'s'` flag?

Outline

The `while` loop

Loops in MATLAB:

- Definition: group of statements repeatedly executed as long as a given conditional expression evaluates to `True`
- Types: `while`, `for`, and vectorizing

The `while` loop

Loops in MATLAB:

- Definition: group of statements repeatedly executed as long as a given conditional expression evaluates to `True`
- Types: `while`, `for`, and vectorizing

Structure in MATLAB:

```
while expression
  statements
end
```

Example.

```
i=0
while true
  i=i+1
end
```

Example

```
o=input('Input a basic arithmetic operation: ','s');
i=1;
while (o(i) >= '0' && o(i) <= '9')
  i = i+1;
end
n1=str2num(o(1:i-1));
n=o(i);
n2=str2num(o(i+1:end));
switch n
  case '+'
    n1+n2
  case '-'
    n1-n2
  case '*'
    n1*n2
  case '/'
    n1/n2
  otherwise
    disp('Not a basic arithmetic operation')
end
```

Questions

In the previous code:

- How is the code formatted?
- What is the user expected to input?
- What is the purpose of the `while` loop?
- How is switch used?
- What is happening if something else that an integer is input?

The for loop

Structure in MATLAB:

```
for i=start:increment:end
  statements
end
```

The for loop

Structure in MATLAB:

```
for i=start:increment:end
  statements
end
```

Example.

```
a=[]
for i=0:2:100
  a=[a i]
end
```

Questions.

- How is the code indented?
- What is this code doing?

Vectorizing loop

MATLAB: array/matrix language

$$\downarrow$$

Convert `for`/`while` loops into vector/matrix operations

Vectorizing loop

MATLAB: array/matrix language

$$\downarrow$$

Convert `for`/`while` loops into vector/matrix operations

Example.

```
a=zeros(1,100000000); i=1;
tic; while i<=100000000; a(i)=2*(i-1); i=i+1; end; toc;
a=zeros(1,100000000);
tic; for i=1:100000000; a(i)=2*(i-1); end; toc;
tic; [0:2:199999999]; toc;
```

Questions.

- Reformat and indent the code with one instruction per line
- What is this code doing?

The `continue` and `break` commands

More advanced loop commands:

- `continue`: directly jump to the next iteration
- `break`: exit the loop early

The continue and break commands

More advanced loop commands:

- continue: directly jump to the next iteration
- break: exit the loop early

Example.

```
d={'1','2','3','4','5','6','7','8','9','0'}; cnt=0;
w=input('Input a word: ','s');
for i=1:length(w);
  switch w(i);
    case d;
      continue;
    case ' ';
      break;
    otherwise
      cnt=cnt+1;
  end,
end
cnt
```

Questions

In the previous code:

- What is this code doing?
- How is the code indented?
- What is the variable `d`?
- How are `continue` and `break` used?

Efficiency

Arrays are stored linearly in memory:

- Row first: elements are read by row
- Column first: elements are read by column
- MATLAB uses the *column-major order*
- Column should be in the outer loop

Efficiency

Arrays are stored linearly in memory:

- Row first: elements are read by row
- Column first: elements are read by column
- MATLAB uses the *column-major order*
- Column should be in the outer loop

Example.

To store the matrix $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$ in memory is MATLAB using 1 2 3 4 5 6 or 1 4 2 5 3 6?

Example

```
N = 10000; a = zeros(N);
tic;
  for j = 1:N
    for i=1:N
      a(j,i) = 1;
    end
  end
toc;
```

Example

```
1 N = 10000; a = zeros(N);
2 tic;
3   for j = 1:N
4     for i=1:N
5       a(j,i) = 1;
6     end
7   end
8 toc;
```

Questions.

- What is this code doing?
- Is *j* representing the rows of the columns, what about *i*?
- What is happening if *i* and *j* are switched on line 5?

Accessing specific elements in a matrix

Access elements depending on a *logical mask:*

1. Generate an logical array depending on some condition
2. Apply a transformation only on a 1 in the logical array

Accessing specific elements in a matrix

Access elements depending on a *logical mask:*

1. Generate an logical array depending on some condition
2. Apply a transformation only on a 1 in the logical array

Example.

- For a matrix A set all its elements larger than 10 to 0
- Given a vector square all its even values and cube the others

```
A=magic(5); B=A >10;A(B)=0
a=input('Vector: ')
b=(mod(a,2)==0);
c=a.^2;
c(~b)=a(~b).^3
```

Questions

In the previous code:

- What is the result of `whos B`?
- What does $B = A > 10$ mean?
- What is the goal of line 3?
- After line 4 what is in c?
- Why is ~b used?

Key points

- How to write simple scripts in MATLAB?
- What is the difference between an array and a matrix?
- What is a conditional statements?
- What loop types exist in MATLAB, which one is best used?
- What is a logical mask?

Thank you!