SM-2302: Software for Mathematicians

Matlab1: Basic Operations in MATLAB

Dr. Huda Ramli
Mathematical Sciences, Faculty of Science, UBD
huda.ramli@ubd.edu.bn

Semester I, 2023/24

Basic Operations in MATLAB

Simple arithmetic operations can be done in MATLAB.

- Type the addition command: `>> 4+5`
  Press Enter and MATLAB prints: `ans = 9`
- Enter the subtraction command: `>> 5-9`
  Notice that it prints: `ans = -4`
- Multiplication: `>> 5*9`

  `ans = 45`
- Division: `>> 8/9`

  `ans = 0.8889`
- Exponentiation: `>> 5^3`

  `ans = 125`

Variable Assignment

Notice that when the command `4 + 5` is typed above, `MATLAB` responds with `ans = 9` and forgets this calculation when we enter another command. In order to save values, we **assign** the output to a variable.

For example,

command	MATLAB response
`>> x = 18`	`x = 18`
`>> y = x + 2`	`y = 20`

When you assign a variable, the variable name appears in the Workspace viewer. To view the contents of the variable, double-click on the variable in the workspace or enter the variable name on the command line.

Example 1

Predict the responses of the following commands and check your understanding.

command	MATLAB response
`>> a = 3; b = 4; c = 5;`	
`>> a = b* c`	
`>> b/a`	
`>> a = a-18`	
`>> a^b`	

Order of Operations

`MATLAB` uses a fairly standard order of priority for operations:

^	>	*	/	>	+	-

This means addition and subtraction have the same priority, which is below multiplication and division; and powers have the highest priority.

For example, the command `>> 5 - 1/3`
will return an evaluation of 4.6667 or $5 - 1/3$, as division is done before the substraction.

Example 2 (Operations with same precedence)

Investigate the expressions below to see the order in which the operations are performed by MATLAB.

```
>> 5 - 7 - 8
>> 5 / 7 / 8
>> 5 / 56
>> 5 / 7 * 8 * 9
```

Example 3

Evaluate the following expressions. Let `a=4; b=5; c=8;`

(i) $\dfrac{a^b - c/b}{c - a}$ (ii) $\dfrac{a^{c-b}}{c - b}$ (iii) $\dfrac{a^{3/2}}{b}$ (iv) $\dfrac{a - b(c - a)}{c - a}$

Function Evaluation

MATLAB allows extra functionality with its built-in functions:

command	output	remarks
`sqrt(225)`	15	
`pi`	3.1416	π is built-in
`exp(1)`	2.7183	e is not built-in
`sin(pi/2)`	1	
`x = pi/4`	`x = 0.7854`	assigns x
`tan(x)`	1.000	as $x = \pi/4$ by above
`sin(x)^2 + cos(x)^2`	1	again, because $x = \pi/4$

NOTE:

- When you get an error message, check that you have spelled the function name correctly.
- For trigonometric functions, arguments must be specified in radians, not in degrees. Recall, to convert from degrees to radians you multiply by $\pi/180$.

Buit-in mathematical functions

MATLAB notation	Mathematical notation	Operation
`sqrt(x)`	$\sqrt{x}$	square root
`abs(x)`	$\|x\|$	absolute value
`exp(x)`	e^x	exponential function
`log(x)`	$\ln x$	natural logarithm
`log10(x)`	$\log_{10} x$	logarithm base 10
`sin(x)`	$\sin x$	sine
`cos(x)`	$\cos x$	cosine
`tan(x)`	$\tan x$	tangent
`asin(x)`	$\sin^{-1} x$	inverse sine
`acos(x)`	$\cos^{-1} x$	inverse cosine
`atan(x)`	$\tan^{-1} x$	inverse tangent

Vectors

- A **vector** in MATLAB is a variable that contains more than one element.
- Vectors are created in MATLAB using square brackets: `[ ]`
  - To create a row vector:

```
>> a = [1 2 3]   % use percentage symbol for comments
```

  - To create a column vector:

```
>> b = [4;5;6];   % suppress output by adding a semicolon at the end of command line
```

Sequences

- To generate a squence of numbers separated by 1, use the colon symbol as in `a:b`

```
>> c = 1:5                    % c = [1 2 3 4 5]
>> -1:5                       % -1 0 1 2 3 4 5
>> -(1:5)                     % -1 -2 -3 -4 -5
```

- If we want a step size h, we use the syntax `a:h:b`

```
>> x = 0:2:10                 % evens: 0,2,4, …, 10
>> y = 0:2:9                  % evens: 0,2,4, …, 8
>> z = 0:15:60                % 0,15,30,45,60
```

- For evenly-spaced numbers between two points without figuring out the step size, we can use the `linspace` function:

```
>> linspace(0,1)              % 100 numbers between 0 and 1
>> linspace(0,pi,3)           % 3 numbers: [0 pi/2 pi]
```

Vector operations

```
>> a+3          % add scalar to a vector a
>> a+b          % element-by-element addition
>> a-b          % element-by-element subtraction
>> a*3          % scalar-vector multuplication
>> a.*b         % element wise multiplication
>> a*b          % dot product
>> dot(a,b)     % dot product
>> a'           % transpose
>> cross(a,b)   % cross product (only for arrays with 3 elements)
>> a./b         % element wise division
>> a/b          % pseudoinverse: ab^-1
```

Matrices

MATLAB = MATtrix LABoratory
i.e. a matrix is the primary object involved in any MATLAB computation.

- In general, we create matrices with more than one row and coloumn.

```
>> A = [1 2 3 4
   5 6 7 8
   9 10 11 13 ]
```

```
>> A = [1 2 3 4; 5 6 7 8; 9 10 11 12]
% generates the same matrix
```

- Functions to create matrices:

```
>> zeros(5,5)      % all zeros
>> ones(5,5)       % all ones
>> I=eye(5)        % unit matrix
>> rand(5,5)       % unifomly distributed random elements, between 0 and 1
>> randn(5,5)      % normally distributed random elements, mean 0 and variance 1
```

Matrix arithmetic

Matrices can be combined using the operations `+`, `-`, `*` to form new matrices - provided that the matrices have the same dimensions. For example, if

$$\mathbf{A} = \begin{pmatrix} -3 & -1 \\ 3 & 2 \\ 3 & 0 \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} 7 & 3 \\ -2 & 2 \\ 3 & -1 \end{pmatrix}, \text{ and } \quad \mathbf{C} = \begin{pmatrix} -3 & -2 & -2 \\ 5 & 8 & -2 \end{pmatrix}$$

then the MATLAB commands gives

```
>> D = A+B
```

% $\mathbf{D} = \begin{pmatrix} 4 & 2 \\ 1 & 4 \\ 6 & -1 \end{pmatrix}$

```
>> E = B-A
```

% $\mathbf{E} = \begin{pmatrix} 10 & 4 \\ -5 & 0 \\ 0 & -1 \end{pmatrix}$

```
>> F = C+A
```

% an error message because dimensions do not match

Matrix operations

```
>> A+3          % matrix A plus a scalar
>> A*3          % matrix A multiplied by a scalar
>> A*a          % matrix A multiplies a vector a
>> sin(A)       % element-wise sine of a matrix
>> exp(A)       % element-wise exponential of a matrix
>> A+B          % matrix addition
>> A-B          % matrix subtraction
>> A*B          % matrix multiplication
>> A.*B         % element-wise multiplication
>> A.^3         % element-wise power
>> A'           % transpose or complex conjugate transpose of a matrix
>> inv(A)       % inverse of a matrix
>> A./B         % element-wise division
>> A/B          % equivalent to A*inv(B)
>> A\B          % backslash operator: inv(A)*B
>> det(A)       % determinant of a matrix
```

Matrix indexing

- Index starts from 1 (not from 0)
- Column-major convention

```
>> A(3,2)          % the element of 3rd row and 2nd column
>> A(:,1)          % the 1st column
>> A(2,2:3)        % through 2nd to 3rd elements of 2nd row
>> sum(A(2,:))     % sum of all elements of 2nd row
>> max(A(3,:))     % maximum element of the 3rd row
```

Example 4

(a) Create length 3 vectors and 3×3 matrices.

(b) Practice the vector operations listed above.

(c) Practice the matrix operations listed above.

Cell array

Objects of different types (e.g. data, text, color) in MATLAB can be represented as matrices in a cell array.

Create a cell array:

- `>> cell(size1,size 2,...)`
  % creates a multidimensional cell array. Refer to `help cell` for more details
- `>> myCell = {2, [7 8 9], [1 2 3; 4 5 6]; 'text', rand(5,5),{11; 22; 33}}`
  % initialize a 2-by-3 cell array of different types of elements

Example 5

Determine what is produced by the following MATLAB statements.

(a) `>> myCell{1,3};`

(b) `>> myCell{2,1:2};`

(c) `>> myCell{1,:};`

(d) `>> iscell(myCell)`

Creating plots

The basic idea is to first plot several points (x_i, y_i) and then to connect them together using lines using the command `plot(x,y)`.

Example 6 (Plotting $y = x^2 - \sqrt{x+3} + \cos 5x$)

```
>> n = 81;
>> x = linspace(-3, 5, n);
>> y = zeros(1,n);
>> y = x.^2 - sqrt(x+3) + cos(5*x);
>> plot(x,y)
```

Plot of sine and cosine functions

```
x = 0:pi/100:4*pi;  % create a vector for x-coordinates
y = sin(x);  % vector for sin x
y2 = cos(x);  % vector for cos x
plot(x,y,'k')  % plot sin x in black
hold on  % hold allows to overlay several plots on the same set of axes
plot(x,y2,'--r','linewidth',2)  % plot cos x in thick red dashed line
xlabel('x')  % add graph labels
ylabel('y')
axis([0  4*pi  -1  1])  % axis([xmin xmax ymin ymax])
title('Plot of sine and cosine functions','FontSize',12)
legend('sin(x)','cos(x)')
```

See `help plot` documentation for more plotting capabilities.

Example 7 (Chebyshev polynomials)

Chebyshev polynomials are used in a variety of engineering applications. The jth Chebyshev polynomial $T_j(x)$ is defined by

$$T_j(x) = \cos(j \arccos(x)), \quad -1 \leq x \leq 1.$$

(a) Explain what happens when the following MATLAB code is executed.

```
x = linspace(-1,1,201);
T1 = cos(acos(x));
T3 = cos(3*acos(x));
T5 = cos(5*acos(x));
T7 = cos(7*acos(x));
```

```
subplot(2,2,1), plot(x, T1)
subplot(2,2,2), plot(x, T3)
subplot(2,2,3), plot(x, T5)
subplot(2,2,4), plot(x, T7)
```

(b) Compare this to the following `plot` code:

```
subplot(1,1,1)
plot(x, T1, 'b')
hold on
plot(x, T3, 'r')
plot(x, T5, 'g')
plot(x, T7, 'c')
```