Chapter 2: Data Types and Variables

Basic Operations

Variables:

Variables are defined with the assignment operator, "**=**". MATLAB is dynamically typed, meaning that variables can be assigned without declaring their type, and that their type can change. Values can come from constants, from computation involving values of other variables, or from the output of a function.

Example:

```
>> x = 17
x =
 17
>> x = 'hat'
x =
hat
>> x = [3*4, pi/2]
x =
   12.0000    1.5708
>> y = 3*sin(x)
y =
   -1.6097    3.0000
```

[End of Example]

Note! MATLAB is case sensitive! The variables x and X are not the same.

Note! Unlike many other languages, where the semicolon is used to terminate commands, in MATLAB the **semicolon** serves to suppress the output of the line that it concludes.

```
>> a=5
a =
     5
>> a=6;
>>
```

As you see, when you type a semicolon (;) after the command, MATLAB will not respond. This is very useful because sometimes you want MATLAB to respond, while in other situations that is not necessary.

Built-in constants:

MATLAB has several built-in constants. Some of them are explained here:

Name	Description
i, j	Used for complex numbers, e.g., z=2+4i
pi	π
inf	∞, Infinity
NaN	Not A Number. If you, e.g., divide by zero, you get NaN

Naming a Variable Uniquely:

To avoid choosing a name for a new variable that might conflict with a name already in use, check for any occurrences of the name using the **which** command:

```
which -all variablename
```

Example:

```
>> which -all pi

built-in (C:\Matlab\R2007a\toolbox\matlab\elmat\pi)
```

You may also use the **iskeyword** command. This command causes MATLAB to list all reserved names.

```
>> iskeyword
ans =
    'break'
    'case'
    'catch'
    'classdef'
    'continue'
    'else'
    'elseif'
    'end'
```

```
'for'
'function'
'global'
'if'
'otherwise'
'persistent'
'return'
'switch'
'try'
'while'
```

Note! You cannot assign these reserved names as your variable names.

Note! MATLAB allows you to reassign built-in function names as variable names, but that is not recommended! – so be carefully when you select the name of your variables!

Example:

```
>> sin = 4
sin =
     4

>> sin(3)

??? Index exceeds matrix dimensions.
```

In this example you have defined a variable "sin" – but "sin" is also a built-in function – and this function will no longer work!

If you accidently do so, use the **clear** command to reset it back to normal.

[End of Example]

Task 1: Basic Operations

Type the following in the Command window:

```
>> y = 16;
>> z = 3;
>> y + z
```

Note! When you use a semicolon, no output will be displayed. Try the code above with and without semicolon.

Note! Some functions display output even if you use semicolon, like ***plot***, etc.

Other basic operations are:

```
>>16-3
>>16/3
>>16*3
```

→ Try them.

[End of Task]

Built-in Functions:

Here are some descriptions for the most used basic built-in MATLAB functions.

Function	Description	Example
help	MATLAB displays the help information available	`>>help`
help <function>	Display help about a specific function	`>>help plot`
who, whos	who lists in alphabetical order all variables in the currently active workspace.	`>>who` `>>whos`
clear	Clear variables and functions from memory.	`>>clear` `>>clear x`
size	Size of arrays, matrices	`>>x=[1 2 ; 3 4];` `>>size(A)`
length	Length of a vector	`>>x=[1:1:10];` `>>length(x)`
format	Set output format	
disp	Display text or array	`>>A=[1 2;3 4];` `>>disp(A)`

plot	This function is used to create a plot	`>>x=[1:1:10];`
		`>>plot(x)`
		`>>y=sin(x);`
		`>>plot(x,y)`

clc	Clear the Command window	`>>cls`
rand	Creates a random number, vector or matrix	`>>rand` `>>rand(2,1)`
max	Find the largest number in a vector	`>>x=[1:1:10]` `>>max(x)`
min	Find the smallest number in a vector	`>>x=[1:1:10]` `>>min(x)`
mean	Average or mean value	`>>x=[1:1:10]` `>>mean(x)`
std	Standard deviation	`>>x=[1:1:10]` `>>std(x)`

Before you start, you should use the Help system in MATLAB to read more about these functions. Type "help <functionname>" in the Command window.

Task 2: Statistics functions

Create a random vector with 100 random numbers between 0 and 100. Find the minimum value, the maximum value, the mean and the standard deviation using some of the built-in functions in MATLAB listed above.

[End of Task]

Arrays; Vectors and Matrices

Matrices and vectors (Linear Algebra) are the basic elements in MATLAB and also the basic elements in control design theory. So, it is important you know how to handle vectors and matrices in MATLAB.

A general matrix A may be written like this:

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nm} \end{bmatrix} \in R^{nxm}$$

In MATLAB we type vectors and matrices like this:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

```
>> A = [1 2; 3 4]
A =  1     2
     3     4
```

or:

```
>> A = [1, 2; 3, 4]
A =  1     2
     3     4
```

→ To separate rows, we use a semicolon ";"

→ To separate columns, we use a comma "," or a space " ".

To get a specific part of a matrix, we can type like this:

```
>> A(2,1)
ans =
    3
```

or:

```
>> A(:,1)
ans =
```

```
     1
     3
```

or:

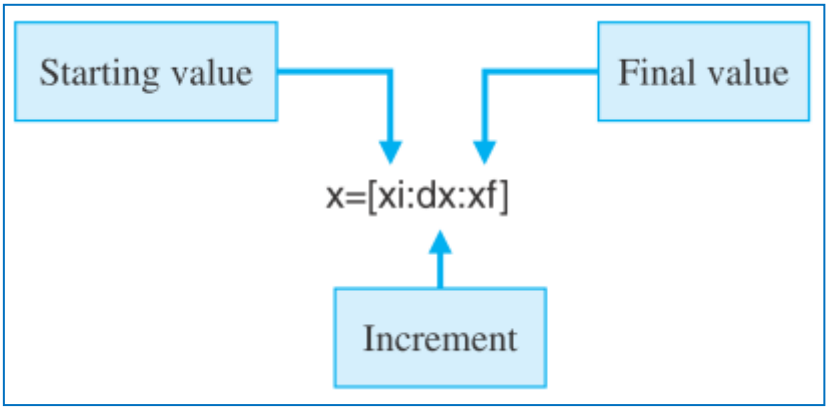
```
>> A(2,:)
ans =
     3     4
```

From 2 vectors x and y we can create a matrix like this:

```
>> x = [1; 2; 3];
>> y = [4; 5; 6];
>> B = [x y]
B =  1     4
     2     5
     3     6
```

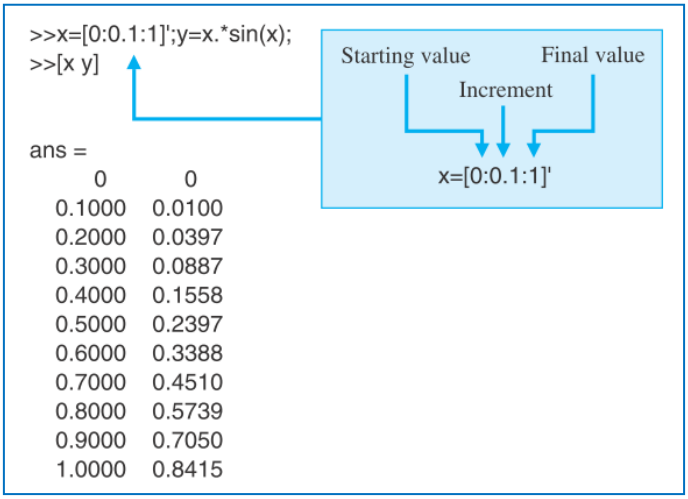
Colon Notation

The "**colon notation**" is very useful for creating vectors:



Example:

This example shows how to use the colon notation creating a vector and do some calculations.



[End of Example]

Task 3: Vectors and Matrices

Type the following vector in the Command window:

$$x = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

Type the following matrix in the Command window:

$$A = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix}$$

Type the following matrix in the Command window:

$$C = \begin{bmatrix} -1 & 2 & 0 \\ 4 & 10 & -2 \\ 1 & 0 & 6 \end{bmatrix}$$

→ Use Use MATLAB to find the value in the second row and the third column of matrix C.

→ Use MATLAB to find the second row of matrix C.

→ Use MATLAB to find the third column of matrix C.

[End of Task]

Deleting Rows and Columns:

You can delete rows and columns from a matrix using just a pair of square brackets [].

Example:

Given:

$$A = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix}$$

To delete the second column of a matrix A, use:

```
>>A=[0 1; -2 -3];
>>A(:,2) = []
A =
    0
   -2
```

[End of Example]

Tips and Tricks

Naming conversions:

When creating variables and constants, make sure you create a name that does not already exist in MATLAB. Note also that MATLAB is case sensitive! The variables x and X are not the same.

Use the which command to check if the name already exists:

```
>> which -all <your name>
```

Example:

```
>> which -all sin
built-in (C:\Matlab\R2007a\toolbox\matlab\elfun\@double\sin)  % double method
built-in (C:\Matlab\R2007a\toolbox\matlab\elfun\@single\sin)  % single method
```

[End of Example]

Large or small numbers:

If you need to write large or small numbers, like $2\,x\,10^5$, $7.5\,x\,10^{-8}$you can use the "e" notation, e.g.:

```
>> 2e5
ans =
     200000
```

```
>> 7.5e-8
ans =
  7.5000e-008
```

Line Continuation:

For large arrays, it may be difficult to fit one row on one command line. We may then split the row across several command lines by using the line continuation operator "…".

Example:

```
>> x = [1 2 3 4 5 ...
6 7 8 9 10]
x =
    1     2     3     4     5     6     7     8     9    10
```

[End of Example]

Multiple commands on same line:

It is possible to type several commands on the same line. In some cases, this is a good idea to save space.

Example:

```
>> x=1, y=2, z=3
x =
     1
y =
     2
z =
     3
```

[End of Example]

Array Operations

We have the following basic matrix operations:

+	Addition
–	Subtraction
*	Multiplication
/	Division
^	Power

The basic matrix operations can be modified for element-by-element operations by preceding the operator with a period. The modified operations are known as **array operations**.

Given

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

Then

$$A.* B = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} \\ a_{21}b_{21} & a_{22}b_{22} \end{bmatrix}$$

The elements of A.*B are the products of the corresponding elements of A and B.

We have the following array operators:

+	Addition
−	Subtraction
.*	Multiplication
./	Division
.^	Power

Example:

```
>> A = [1; 2; 3]
A =
     1
     2
     3
>> B = [-6; 7; 10]
B =
    -6
     7
    10
>> A*B
??? Error using ==> mtimes
Inner matrix dimensions must agree.

>> A.*B
ans =
```

```
-6
14
30
```

[End of Example]