Fundamentals of MATLAB Programming

1.1 Variables and constants in MATLAB

MATLAB variable names consist of a letter, followed by any number of letters, digits, or underscores.

Reserved names for the constants.
eps – error tolerance for flating-point operation

- i and j – if i or j is not overwritten, they both represent $\sqrt{-1}$
- Inf – the MATLAB representation of infinity quantity $+\infty$
- NaN – not a number, which is often returned by the operations 0/0, Inf/Inf and others.
- pi – double-precision representation of the circumference ratio π
- lasterr – returns the error message received last time.
- lastwarn – returns the last obtained warning message.

1.2 Data structure

Double-precision data type

To ensure high-precision computations, double-precision floating-point data type is used, which is 8 bytes(64 bits). It is composed of 11 exponential bits, 53 number bits and a sign bit, representing the data range of $\pm \times 10^{308}$.

Symbolic data type

Before finding analytical solutions, the related variables should be declared as **symbolic**, with the **syms** statement `syms var_list var_props`, where *var_list* is the list of variables to be declared, separated by spaces. If necessary, the types of the properties of the varibles can be assinged by *var_props*, such as **real** or **positive**.

The varible precision arithmetic function vpa() can be used to display the symbolic variables in any precision. The syntax of the function is `vpa(A, n)` or `vpa(A)`, where A is the variable to be displayed, and n is the number of digits expected, with the default value of 32 decimal digits.

Example 1.1 Display the first 300 digits of π.

Solution

```
>> vpa(pi, 300)
```

Other data types

(i) **Strings** String variables are used to store messages.

(ii) **Multi-dimensional arrays**

(iii) **Cell arrays** Cells are extension of matrices, whose elements are no longer values. The element, referred as *cells*, of cell arrays can be of any data type.

(iv) **Classes and objects**

1.3 Basic structure of MATLAB

Two types of MATLAB statements can be used:

(i) **Direct assignment** The basic structure of this type of statement is

```
variable = expresion
```

and expression can be evalued and assigned to the variable defined in the left-hand-side, and established in MATLAB worksapce. The semicolon can be used to suppress the display of intermediate results, and the reserved variable **ans** always stores the latest statement without a left-hand-side variable.

Example 1.2 Specify the matrix $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{bmatrix}$ into MATLAB workspace.

Solution

```
>> A = [1,2,3; 4,5,6; 7,8,0]
```

The size of a matrix can be expanded or reduced dynamically, with the following statemensts.

```
>> A = [1,2,3; 4,5,6; 7,8,0];      % assignment is made, however no display
   A = [[A; [1 2 3]], [1;2;3;4]]  % dynamically define the size of matrix
```

Example 1.3 Enter complex matrix $B = \begin{bmatrix} 1+j9 & 2+j8 & 3+j7 \\ 4+j6 & 5+j5 & 6+j4 \\ 7+j3 & 8+j2 & 0+j1 \end{bmatrix}$ into MATLAB.

Solution The notations i and j can be used to describe the imaginary unit.

```
B=[1+9j,2+8j,3+7j; 4+6j 5+5j,6+4j; 7+3j,8+2j 0+1j]
```

(ii) **Function call statement** The basic statement structure of function call is
$[returned\_arguments] = function\_name(input\_arguments)$
Generally the function names are the file names in the MATLAB path.

1.4 Colon expressions and sub-matrices extration

Colon expression is an effective way in defining **row vectors**. The typical form of colon expression is `v = s1:s2:s3`. Thus a row vector **v** can be established in MATLAB workspace, with the initial value *s1*, the increment *s2* and the final value *s3*. The default value for increment is 1.

Example 1.4 For different increments, establish vectors for $t \in [0, \pi]$
Solution

```
>> v1 = 0 : 0.2 : pi    % row vector from 0 to 3 with step of 0.2

>> v2 = 0 : -0.1 : pi   % negative step, no vector generated
   v3 = 0 : pi          % with the default step of 1
   v4 = pi : -1 : 0     % a vector in the reverse order of v1
```