MATLAB Strings and Input/Output

This is Chapter 2 (2.6-2.7) and

and Chapter 6 (6.1,6.2.1,6.2.7-6.2.9) and Chapter 8 (8.6.1-8.6.3)

Disclaimer: *These abbreviated notes DO NOT substitute the textbook for this class. They should be used IN CONJUNCTION with the textbook and the material presented in class. If there is a discrepancy between these notes and the textbook, ALWAYS consider the textbook to be correct. Report such a discrepancy to the instructor so that he resolves it. These notes are only distributed to the students taking this class with A. Gerbessiotis in Fall 2014 ; distribution outside this group of students is NOT allowed.*

In MATLAB we are going to use quite frequently strings.

4.1 String. A **string** is a concatenation of zero or more characters. The string with no characters will be called the NULL string or the empty string and for this discussion we shall use the symbol ϵ to represent it.

4.2 Length of a string. The length of a string is the number of characters in it or zero if it is ϵ.

4.3 MATLAB strings. If we want to define a string in MATLAB we use the single-quote character to surround it. Thus `'abcdefgh'` is a string of eight characters, the first eight letters of the alphabet in lower-case. And `''` is the empty string; note that the second quote is immediately after the first with no other space in between. We usually assign a string to a variable to hold it in the memory of a computer and to retrieve it easily by the name of the variable rather than the (quite possibly lengthy) value of it.

Thus `a= 'abcdefgh'` assigns string `'abcdefgh'` to variable `a` in MATLAB. It is easier to say or write variable **a** rather than `'abcdefgh'`. MATLAB has a `function` that gives us the length of a string; this is helpful if we forgot its length, or we have difficulty counting characters in very long strings. The function `length` returns the length of a string. Thus `length(a)` will return 8 and so will `length('abcdefgh')`. Naturally `length('')`, i.e. the length of the empty string is `0`. Given that in MATLAB a string is a row vector of MATLAB `char` data type, it is clear why `length` works in this context. (As a reminder `length` gives the length of a vector or the largest dimension of a matrix in MATLAB.)

4.4 String storage. Note that MATLAB is using UNICODE characters that require two bytes per character. Thus a string of length 10 characters will require a total of 20 bytes when stored in MATLAB. A `char` data type will be used to store a string in a `char` array of geometry/shape 1×10 for this example.

(c) Copyright A. Gerbessiotis. All rights reserved.

```
>> a = 'abcdefgh'   % a string of 8 characters
a=                  % note the indentation; strings is getting printed from the far left corner
abcdefgh
>> b = ''
b=   ''             % there is one exception: the empty string
>> whos
  a         1x8                16  char
  b         0x0                 0  char
>> length(a)
ans = 8
>> length(b)
ans = 0
>> a= [1 2 3; 4 5 6]
a =  1     2     3
     4     5     6
>> mat2str(a)
ans = [1 2 3;4 5 6]
```

4.5 Examples with strings. We define two strings `a, b`. String `a` is as defined above, i.e. `a= 'abcdefgh'`, whereas `b` is the empty string. The two variables used to represent the two strings are stored internally by MATLAB. If we forget their names, we can retrieve all variables defined by us, the user, by typing in MATLAB `whos`. Command (or function) `whos` provides us with information about the variables defined, such as their name, what array dimension they are, their length in characters if they are strings (this is our case), the amount of space used internally by MATLAB for them and some other classification of the variables. For our case `a` is an 1×8 array (i.e. a row vector). We can hypothesize that a (UNICODE) character in MATLAB is stored in two bytes since the total number of bytes used is 16. If the variables defined are no longer needed, we can clean them up by using the function `clear`. After we use the `clear`, the variables previously defined such as `a,b` are no longer available to us.

4.6 mat2str: Print a matrix as a string. The funcion `mat2str(a)` converts a matrix a into a string for printing purposes.

(c) Copyright A. Gerbessiotis. All rights reserved.

4.7 `int2str` **and** `num2str` The former converts an integer into a string of the appropriate length and the latter an arbitrary number (integer or not) into a string. The length of the string will depend on the digits of the number plus a possible sign (for negative numbers) plus a decimal point for real numbers.

Thus `num2str(-1.234)` will be converted into the MATLAB string '-1.234' of 6 characters stored into a `char array` of 12 bytes. Note that `-1.234` can be stored into a `double` in arithmetic form that requires only 8 bytes!

4.8 `double(x)` **and** `single(x)` `char(x)`. We can explicitly convert a matrix object from one data type to another. One needs to be careful in interpreting the result. Yet for the example below the weakly-typed nature of MATLAB is exemplified. In the first case a is a logical variable, in the second case it becomes a double, and then a character occupying 1, 8 and 2 bytes respectively! Function `single(x)` converts x into a 4-byte floating-point number as opposed to `double(x)` that converts x into an 8-byte floating-point number.

```
>> a=int8(123)     % A one-bytes integer 123 (int8 data type)
>> whos a
a  1x1   1  int8
>> b=int2str(a)    % Convert 123 into a string and store the string into b
b =
123
>> whos b          % b is '123' i.e. 6B (2 per character 1 2 3)
b  1x3   6  char
>>c=str2num('123')          % 123 will be converted into a number (i.e. double)!!
>> whos c
c   1x1   8  double
```

(c) Copyright A. Gerbessiotis. All rights reserved.

4.9 Default format. The default value of the `format` command is `short`. By default numbers in MATLAB are printed to 4 decimal digits (on the right of the decimal point) rounded up (thus a 3.14157 will be printed as 3.1416). If the number is too small or too large, the number will be converted into scientific notation and will be displayed with the use of `e` to denote 10^.
The first three examples below show the possible output for a real number.

```
>> a=123.45                    |>> format bank
a = 123.4500                   |>> b
>> b=1234.5                    |b = 1234.50
b = 1.2345e+03                 |>> format rat
>> c=0.00012345                |>> a
c = 1.2345e-04                 |a = 2469/20
>> format short e              |>> b
>> a                           |b = 2469/2
a = 1.2345e+02                 |>> c
>> format short g              |c = 9/72904
>> a                           |
a = 123.45                     |
```

4.10 User specified format. If a default format is not desired, a user can change the default behavior.

```
1.   format short     :  default to 4 digits             ||    int8()  : convert to int 8-bit
2.   format long      :  14 digits after decimal         ||    int16() :               16-bit
3.   format short e   :  x.abcd [1+4digits] plus exponent||    int32() :
4.   format short g   :  best of 1 and 3                 ||    int64() :
5.   format long  e   :         [15digits]  plus exponent||   uint8()  : convert to unsigned 8-bit int
6.   format long  g   :  best of 2 and 5                 ||   uint16() :
7.   format bank      :  "dollar and cents" notation     ||   uint32() :
8.   format hex       :  hexadecimal                     ||   uint64() :
9.   format rat       :  rational approximation          ||   get(0,'Format');
10.  format compact   :  suppress line feeds             ||   format('short','g')
11.  format loose     :  restore  line feeds             ||
12.  format +         :  only signs are printed          ||   intmax,intmin,realmax,realmin
13.  format short eng :  exponent is multiple of 3
14.  format long  eng :  exponent is multiple of 3
```

(c) Copyright A. Gerbessiotis. All rights reserved.

```
>> format hex
>> 14
ans= 402c000000000000             % 14 is not integer 14 but double(14)
>> whos
  ans   1x1   8  double array
>>int8(14)
ans = 0e

>>whos
  ans   1x1   1  int8 array

>>int16(14)
ans = 000e
>>whos
  ans   1x1   1  int16 array
>>uint8(256)
ans= ff
>>uint8(1000)
ans= ff
```

4.11 Default representation of a number is `double` even if it looks to be an integer. Note that `format hex` will print in hexadecimal form the values of variables usually stored in double. If you want to print the corresponding integer you need to explicitly convert the double say 14 in the example above to an integer 14 using 8 or 16 bits. Then that integer will be printed as a hexadecimal 8-bit or 16-bit integer i.e. using 2 or 4 hexadecimal digits. Note that integer conversions truncate large numbers. Thus 256, and 1000 are represented in 8-bit as `ff` which is 255 in unsigned 8-bit format. If `int8(256)` were to be attempted the largest 8-bit integer is 127 and thus `7f` would be printed instead!

(c) Copyright A. Gerbessiotis. All rights reserved.

```
>> a=num2str(12345)
a =
12345
>> disp(a)  % a is a string containing characters 1 2 3 4 5
12345
>> a-0      % 1 2 3 4 5 in Unicode is  49, 50, 51, 52, 53 (and so are in ASCII btw)
ans =
      49             50             51             52             53
>> whos a
  Name      Size                     Bytes  Class
  a         1x5                         10  char array
>> disp(a,a,a)                    % Only one argument to disp!
Error using disp
Too many input arguments
>> disp([a a a ])
123 123 123
>> a= 123.45                      % a now is a number
  123.4500
>> disp([ a a a ])
 123.45000 123.4500 123.4500
>> fprintf('a=  %e a= %f a=%g',a,a,a)      % fprintf allows for more options!
a=  1.234500e+02 a= 123.450000 a=123.45>>
```

4.12 `disp`. Function `num2str` converts a number into a string of characters, as noted earlier.Then that array can be displayed with the use of function `disp`. Function `disp` displays strings or numbers in a more compact form that printing the variable's name and expecting its value to be displayed over possibly multiple lines. Command `disp` takes only one argument. If you need to print more than one variable or values you need to pack all othem in an vector/array/matrix.

(c) Copyright A. Gerbessiotis. All rights reserved.

4.13 C language `fprintf`. We mentioned earlier that MATLAB has been implemented in *C*. Thus certain language features of the C programming language are available to MATLAB. Function `fprintf` is one of them. The format of `fprintf` is `fprintf(format,data)`. The `format` is a format string that contains placeholders (i.e. **conversion characters**) of where variables are to be displayed and in what format. The variable names is the list `data`. The variable names in data are separated by commas, and how many variables there are in the `data` are determined by the number of placeholders in `format`. This is a variable-number-of-argument function. For those of you proficient in C, note that the double-quotes `"` of C have become single-quotes `'` in MATLAB's `format` string.

```
fprintf('%d %f %s %8d %5.2f \n',1,2.3,'abc',4,pi);
```

In the `format` of the example above,

- `format` is `'%d %f %s %8d %5.3f \n'`
- `data` is `1,2.3,'abc',4,pi`.
- a `%d` indicates that a variable will be displayed as integer base-10,
- a `%f` indicates that a variable will be displayed as a real in floating-point format,
- a `%s` indicates that a string variable will be displayed.
- a `%8d` indicates that a variable will be displayed as an 8-digit integer base-10.
- a `%5.2f` indicates that a variable will be didsplayed as a real in floating-point format: exactly two decimal digits will be printed, the decimal point, and at least one integer digit (or as many as needed no matter what). The total length will be 5 including the 2 decimal digits and the period. If the number only requires 4 digits (and period) then a space will precede it to make 5.
- A one-to-one correspondence between placeholders and data will print `1` as `%d` , `2.3` as `%f`, `'abc'` as `%s`, 4 as `%8d`, and `pi` as `%5.2f`.
- `\n` is one character. It denote a newline; the current line will be completed and the printing cursor will move to the first character position of the next line.

(c) Copyright A. Gerbessiotis. All rights reserved.

Thus a **conversion character** such as the `d` in `%d` indicates an integer. An integer number can be inserted between the percent and the conversion character to indicate **length**. Another character can be inserted between the two such as `h, l, b` to indicate `subtype`. Instead of a single integer one can have `x.y` two integers separated by a period and the integer to the right of the period indicates the precision. By default justification is right; a $-$ after the percent but before x would denote left justification. Padded zeroes can be introduced with a 0 after percent, and a sign will be printed with a + (for negative numbers a sign is always printed anyway).

```
Conversion Characters (following the %)
***************************************
  %d   %i      :  integer base-10                 Example:  %d %i %10d  %8i
  %ld  %li     :  64-bit integer base-10           Example:  %10ld %8li
  %hd  %hi     :  16-bit integer base-10           Example:  %10hd %10hi
  %hu  %u  %lu : 16-bit 32-bit 64-bit unsigned integer base-10
  %ho  %o  %lo : 16-bit 32-bit 64-bit unsigned integer base-8  (octal)
  %hx  %x  %lx : 16-bit 32-bit 64-bit unsigned integer base-16 (hexadecimal a-f)
  %hX  %X  %lX : 16-bit 32-bit 64-bit unsigned integer base-16 (hexadecimal A-F)
  %f           : fixed point real number
  %e           : exponential notation           i.e. 3.1415e+00
  %E           : same as %e but with capital E  i.e. 3.1315E+00
  %g %G        : the more compact of %e and %f and %E and %f respectively
  %bx %bX      : double-precision hexadecimal
  %bo %bu      : double-precision octal and unsigned
  %c           : single character
  %s           : string of characters
Subtypes
********
  l,h,b   in %hd etc , %lu etc, %bo etc.
Width/Justification
*******************
 none : right justify   - : left justify   0 : padded zeroes  + : print sign
Examples: %10.7f  ,  %-10.7f ,  %010.7f  , %+10.7f
```

(c) Copyright A. Gerbessiotis. All rights reserved.

```
Special Characters (partial list)
*********************************
''  : single quote           \a  : alarm (bell)   \t  : tab
%%  : single percent         \b  : backspace      \xN : hexadecimal ASCII/UNICODE char
\\  : single backslash       \n  : newline        \N  : octal       ASCII/UNICODE char

Examples. fprintf('\40') prints an octal, fprintf('\x20') prints hexadecimal
```

4.14 Special characters. Special characters can also be included in the format string: `\n` prints a new line; a `\t` prints a tab character; \\ prints a single backslash, `%%` prints a single percent (rather than being interpreted as a placeholder). A `\xN` prints the ASCII/UNICODE character with hexadecimal value N, and `\N` prints the char for that octal number.

4.15 Width and justification. If a number appears between the percent and the placeholder character, it indicates minimum number of digits printed. For example `%8d` means at least 8 digits will be printed The more complex `%5.3f` means that at least 5 digits (including the decimal point) will be printed of which exactly 3 to the right of the decimal; this leaves room for a single integer digit. If the number to be displayed is 10.123 definitely we need a width of 6 characters not 5! And of course -10.123 would require 7 characters total. Thus `5.3` provides a suggestion rather than a hard limit. A `%-5.3f` left justifies (default: right), a `%+5.3f` prints a sign, and a `%05.3f` pads with zeroes.

(c) Copyright A. Gerbessiotis. All rights reserved.

4.16.1 tic, toc One may time a surround a piece of MATLAB code with a sequence of tic and toc commands. The pair will time the execution from the tic to the toc and report elapsed time.

4.16.2 cputime Reports the cputime used since the start of the current MATLAB session.

4.16.3 clock, date, datestr(now), weekday(now) Reports the current date and time as a date vector. It is preferable to use date and datestr(now) to get the date and date and time respectively in printable/viewable form. Function call weekday(now) reports the day of the week as an integer (starting from Sunday).

4.16.4 calendar Displays a calendar for the current month.

```
% Examples
>> tic;  k(10000,10000)=10; toc
>> clock
    ans = 1.0e+03 *
        2.0140    0.0090    0.0040    0.0180    0.0030    0.0179
>> date
    ans = 04-Sep-2014
>> datestr(now)
    ans = 04-Sep-2014 18:03:22
>> weekday(now)
    ans = 5
```

(c) Copyright A. Gerbessiotis. All rights reserved.

4.17 input. In its simplest form `a=input('prompt text ')` function input prompts a user in the command window by printing the text `prompt text` of the argument string and waits for a user's response. It is the evaluated and returned into variable `a`. If however one uses `a=input('prompt text ','s')` instead, the evaluation of the user's input is not performed. The latter is an easy way to assign to a a string that say contains quotation marks. If the `'s'` IS NOT used, every quotation mark needs to be typed twice; its inclusion in the input as a second argument eliminates this need.

```
>> a=input('enter text ')
enter text  'text'
 a= text
>> a=input('enter text ','s')
enter text  'text'
 a= 'text'
```

One may also create menus using the command menu. Its first argument is a prompts with the remaining arguments strings representing possible options. The string chosen by the user is returned by the menu function call as an integer (starting with 1 for the first choice).

(c) Copyright A. Gerbessiotis. All rights reserved.