The MATLAB `fprintf` Command

Gerald W. Recktenwald
Department of Mechanical Engineering
Portland State University
gerry@pdx.edu

Text Output with `disp` and `fprintf`

Output to the command window is achieved with either the `disp` function or the `fprintf` function. Output to a file requires the `fprintf` function.

`disp` Simple to use. Provides limited control over appearance of output.

`fprintf` Slightly more complicated than `disp`. Provides total control over appearance of output.

The `disp` function (1)

Syntax:

```
disp(outMatrix)
```

where *`outMatrix`* is *either* a string matrix or a numeric matrix.

Examples: *Numeric output*

```
>> disp(5)
       5

>> x = 1:3;  disp(x)
       1     2     3

>> y = 3-x;  disp([x; y])
       1     2     3
       2     1     0

>> disp([x y])
       1     2     3     2     1     0
```

The `disp` function (1)

Examples: *Numeric output*

```
>> x = 1:3;  y = 3-x;
>> disp([x y])
     1     2     3     2     1     0

>> disp([x' y])
???  All matrices on a row in the bracketed expression
 must have the same number of rows.
```

Note: The last statement shows that the input to `disp` must be a legal matrix.

The `format` function

The `format` function controls the precision of `disp` output.

```
>> format short
>> disp(pi)
      3.1416

>> format long
>> disp(pi)
      3.14159265358979
```

Alternatively, a second parameter can be used to control the precision of the output of `num2str`

```
>> disp([’pi = ’,num2str(pi,2)])
pi = 3.1

>> disp([’pi = ’,num2str(pi,4)])
pi = 3.142

>> disp([’pi = ’,num2str(pi,8)])
pi = 3.1415927
```

The `fprintf` function (1)

Syntax:

```
fprintf(textMessage)
fprintf(formatString,listOfVariables)
fprintf(fileHandle,formatString,listOfVariables)
```

In the first form, *`textMessage`*, is a string that is printed to the command window.

In the second form, *`formatString`* is used to convert the variables in *`listOfVariables`* to a string that is then printed to the command window.

In the third form, *`formatString`* is used to convert the variables in *`listOfVariables`* to a string that is then printed to a file. *Note* that *`fileHandle`* needs to be defined with a `fopen` command before the third form of `fprintf` is used.

Notes to `C` programmers:

1. The MATLAB `fprintf` function uses single quotes to define the format string.
2. The `fprintf` function is vectorized. (See examples on following slides.)

The fprintf function (2)

The fprintf makes it easy to intermingle labeling text with numeric values.

Example:

```
>> x = 3;
>> fprintf('Square root of %g is %8.6f\n',x,sqrt(x));

The square root of 3 is 1.732051
```

The fprintf function (3)

The *outFormat* string specifies how the *outVariables* are converted and displayed. The *outFormat* string can contain any text characters. It also must contain a *conversion code* for each of the *outVariables*. The following table shows the basic conversion codes.

Code	Conversion instruction
%d	format with no fractional part (integer format)
%e	format as a floating-point value in scientific notation
%f	format as a floating-point value
%g	format in the most compact form of either %f or %e
%s	format as a string
\n	insert newline in output string
\t	insert tab in output string

The `fprintf` function (4)

In addition to specifying the type of conversion (e.g. `%d`, `%f`, `%e`) one can also specify the width and precision of the result of the conversion.

Syntax:

```
%wd
%w.pf
%w.pe
```

where *w* is the number of characters in the width of the final result, and *p* is the number of digits to the right of the decimal point to be displayed.

The `fprintf` function (5)

Example: : Specifying field width and precision

Format String	Meaning
`%14.5f`	use floating point format (`%f`) to convert a numerical value to a string 14 characters wide with 5 digits after the decimal point
`%12.3e`	use scientific notation format (`%e`) to convert numerical value to a string 12 characters wide with 3 digits after the decimal point. The 12 characters for the string include the `e+00` or `e-00` (or `e+000` or `e-000` on Windows$^{\text{TM}}$)

The `fprintf` function (6)

More examples of conversion codes

`Value`	`%8.4f`	`%12.3e`	`%10g`	`%8d`
`2`	`2.0000`	`2.000e+00`	`2`	`2`
`sqrt(2)`	`1.4142`	`1.414e+00`	`1.41421`	`1.414214e+00`
`sqrt(2e-11)`	`0.0000`	`4.472e-06`	`4.47214e-06`	`4.472136e-06`
`sqrt(2e11)`	`447213.5955`	`4.472e+05`	`447214`	`4.472136e+05`

The `fprintf` function (7)

The `fprintf` function is vectorized. This enables printing of vectors and matrices with compact expressions. It can also lead to some undesired results.

Examples:

```
>> x = 1:4;  y = sqrt(x);
>> fprintf(’%9.4f\n’,y)
   1.0000
   1.4142
   1.7321
   2.0000
```

The `%9.4f` format string is reused for each element of `y`. The recycling of a format string may not always give the intended result.

```
>> x = 1:4;  y = sqrt(x);
>> fprintf(’y = %9.4f\n’,y)
y =    1.0000
y =    1.4142
y =    1.7321
y =    2.0000
```

The `fprintf` function (8)

Vectorized `fprintf` cycles through the *`outVariables` by columns*. This can also lead to unintended results

```
>> A = [1 2 3; 4 5 6; 7 8 9]
A =
     1     2     3
     4     5     6
     7     8     9

>> fprintf(’%8.2f  %8.2f  %8.2f\n’,A)
    1.00      4.00      7.00
    2.00      5.00      8.00
    3.00      6.00      9.00
```

How to print a table with fprintf (1)

Many times a tabular display of results is desired.

The boxSizeTable function listed on the next slide, shows how the fprintf function creates column labels and formats numeric data into a tidy tabular display. The for loop construct is discussed later in these slides.

How to print a table with fprintf (2)

```
function boxSizeTable
% boxSizeTable  Demonstrate tabular output with fprintf

% --- labels and sizes for shiping containers
label = char('small','medium','large','jumbo');
width = [5; 5; 10; 15];
height = [5; 8; 15; 25];
depth = [15; 15; 20; 35];
vol = width.*height.*depth/10000;   %  volume in cubic meters

fprintf('\nSizes of boxes used by ACME  Delivery Service\n\n');
fprintf('size          width     height    depth    volume\n');
fprintf('               (cm)      (cm)      (cm)     (m^3)\n');
for i=1:length(width)
  fprintf('%-8s  %8d  %8d  %8d   %9.5f\n',...
          label(i,:),width(i),height(i),depth(i),vol(i))
end
```

Note: `length` is a built-in function that returns the number of elements in a vector. `width`, `height`, and `depth` are local variables in the `boxSizeTable` function.

How to print a table with `fprintf` (3)

Example: [Running `boxSizeTable` gives]

```
>> boxSizeTable

Sizes of boxes used by ACME Delivery Service

size          width     height    depth    volume
               (cm)      (cm)      (cm)     (m^3)
small            5         5        15     0.03750
medium           5         8        15     0.06000
large           10        15        20     0.30000
jumbo           15        25        35     1.31250
```

The `fprintf` function (4)

File Output with `fprintf` requires creating a *file handle* with the `fopen` function. All aspects of formatting and vectorization discussed for screen output still apply.

Example: [Writing contents of a vector to a file.]

```
x = ...                            %  content of x
fout = fopen('myfile.dat','wt');   %  open myfile.dat
fprintf(fout,'   k     x(k)\n');
for k=1:length(x)
   fprintf(fout,'%4d     %5.2f\n',k,x(k));
end
fclose(fout)                         % close myfile.dat
```