APPENDIX C

Formatted Output in MATLAB

The `disp` and `format` commands provide simple ways to control the screen output. However, some users might require more control over the screen display. In addition, some users might want to write formatted output to a data file. The `fprintf` function provides this capability. Its syntax is `count = fprintf(fid,format,A,...)`, which formats the data in the real part of matrix `A` (and in any additional matrix arguments) under control of the specified format string `format`, and writes it to the file associated with file identifier `fid`. A count of the number of bytes written is returned in the variable `count`. The argument `fid` is an integer file identifier obtained from `fopen`. (It may also be 1 for standard output—the screen—or 2 for standard error. See `fopen` for more information.) Omitting `fid` from the argument list causes output to appear on the screen, and is the same as writing to standard output (`fid` = 1). The string `format` specifies notation, alignment, significant digits, field width, and other aspects of output format. It can contain ordinary alphanumeric characters, along with escape characters, conversion specifiers, and other characters, organized as shown in the following examples. Table C.1 summarizes the `basic` syntax of `fprintf`. Consult MATLAB help for more details.

Suppose the variable `Speed` has the value 63.2. To display its value using three digits with one digit to the right of the decimal point, along with a message, the session is

```
>>fprintf('The speed is: %3.1f\n',Speed)
The speed is:  63.2
```

Here the "field width" is 3, because there are three digits in 63.2. You may want to specify a wide enough field to provide blank spaces or to accommodate an unexpectedly large numerical value. The `%` sign tells MATLAB to interpret the

Table C.1 Display formats with the `fprintf` function

Syntax	Description
`fprintf('format',A, ...)`	Displays the elements of the array `A`, and any additional array arguments, according to the format specified in the string `'format'`.
`'format'` structure	`%[-][number1.number2]C`, where `number1` specifies the minimum field width, `number2` specifies the number of digits to the right of the decimal point, and `C` contains control codes and format codes. Items in brackets are optional. `[-]` specifies left justified.

Control codes		Format codes	
Code	**Description**	**Code**	**Description**
`\n`	Start new line.	`%e`	Scientific format with lowercase e.
`\r`	Beginning of new line.	`%E`	Scientific format with uppercase E.
`\b`	Backspace.	`%f`	Decimal format.
`\t`	Tab.	`%g`	`%e` or `%f`, whichever is shorter.
`''`	Apostrophe.		
`\\`	Backslash.		

following text as codes. The code `\n` tells MATLAB to start a new line after displaying the number.

The output can have more than one column, and each column can have its own format. For example,

```
>>r = [2.25:20:42.25];
>>circum = 2*pi*r;
>>y = [r;circum];
>>fprintf('%5.2f %11.5g\n',y)
  2.25       14.137
 22.25        139.8
 42.25       265.46
```

Note that the `fprintf` function displays the *transpose* of the matrix `y`.

Format code can be placed within text. For example, note how the period after the code `%6.3f` appears in the output at the end of the displayed text.

```
>>fprintf('The first circumference is %6.3f.\n',circum(1))
The first circumference is 14.137
```

An apostrophe in displayed text requires two single quotes. For example:

```
>>fprintf('The second circle''s radius %15.3e is large.\n',r(2))
The second circle's radius      2.225e+001 is large.
```

A minus sign in the format code causes the output to be left justified within its field. Compare the following output with the preceding example:

```
>>fprintf('The second circle''s radius %-15.3e is large.\n',r(2))
The second circle's radius 2.225e+001      is large.
```

Control codes can be placed within the format string. The following example uses the tab code (\t).

```
>>fprintf('The radii are:%4.2f \t %4.2f \t %4.2f\n',r)
The radii are:   2.25     22.25   42.25
```

The disp function sometimes displays more digits than necessary. We can improve the display by using the fprintf function instead of disp. Consider the program:

```
p = 8.85; A = 20/100^2;
d = 4/1000; n = [2:5];
C = ((n - 1).*p*A/d);
table (:,1) = n';
table (:,2) = C';
disp (table)
```

The disp function displays the number of decimal places specified by the format command (4 is the default value).

If we replace the line disp(table) with the following three lines,

```
E='';
fprintf('No.Plates Capacitance (F) X e12 %s\n',E)
fprintf('%2.0f \t \t \t %4.2f\n',table')
```

we obtain the following display:

```
2          4.42
3          8.85
4          13.27
5          17.70
```

The empty matrix E is used because the syntax of the fprintf statement requires that a variable be specified. Because the first fprintf is needed to display the table title only, we need to fool MATLAB by supplying it with a variable whose value will not display.

Note that the fprintf command truncates the results, instead of rounding them. Note also that we must use the transpose operation to interchange the rows and columns of the table matrix in order to display it properly.

Only the real part of complex numbers will be displayed with the fprintf command. For example:

```
>>z = -4+9i;
>>fprintf('Complex number:    %2.2f \n',z)
Complex number:   -4.00
```

Instead you can display a complex number as a row vector. For example, if $w = -4 + 9i$:

```
>>w = [-4,9];
>>fprintf('Real part is %2.0f. Imaginary part is %2.0f. \n',w)
Real part is -4. Imaginary part is 9.
```