A brief introduction to SCILAB

SCILAB is a powerful and versatile package for mathematical modelling and an excellent tool for solving a wide range of engineering problems. SCILAB supports simple interactive calculation as well as complex and sophisticated programming. SCILAB offers a free open-source multi-platform alternative to the immensely popular Matlab® package. The software is available for Windows, Linux and Mac-OS X and is actively supported by the community. SCILAB is easy to learn and will help you understand all of the intermediate steps in solving even the most complicated problems. This brief introduction is designed to provide you with a few pointers and examples to help you get started. Further information and full documentation is available from the SCILAB website (www.scilab.org).

Commenting your code

Comments can be very useful and you should get into the habit of using them liberally within your scripts. SCILAB comments are preceded by two forward slash characters (see below). Note that multiline comments are not used (each line of a multiline comment needs to be preceded by //).

```
// Flux produced by an inductor
```

Comments can also be added to the end of a line of script, as follows:

```
vmin = 11.5; // Min supply voltage
vmax = 13.8; // Max supply voltage
```

Note the use of the colon at the end of a line of script. Thus suppresses printing of the line of code when the script is executed.

Script lines

A long line of script can be broken with a line-wrap by using the ellipsis (...) at the end of each line to indicate that the command actually continues on the next line. These have not been used in the examples given in the book. Instead the ; character has been used to indicate the point at which a new line must start.

```
s = 1 -1/2 + 1/3 -1/4 + 1/5 - 1/6 + 1/7 ...
     - 1/8 + 1/9 - 1/10 + 1/11 - 1/12;
```

In the examples shown in the book we have shown long lines wrapped but terminated by ; as shown below:

```
s = 1 -1/2 + 1/3 -1/4 + 1/5 - 1/6 + 1/7 - 1/8 + 1/9 - 1/10 + 1/11 -
1/12;
```

Predefined constants

SCILAB's predefined constants are preceded by the % character as follows:

Constant	*Meaning*	*Value*
%pi	π	3.14…
%i	j	$\sqrt{-1}$
%e	*e*	2.713
%inf	$+\infty$	Infinity
%nan	Not a number	

Operators

SCILAB uses conventional mathematical operators:

Operator	*Meaning*
+	Addition
-	Subtraction
/	Division
*	Multiplication
|	OR
&	AND
=	Equality
(…)	Parenthesis
^	Power
‘	Conjugation

Assigning values to variable

Operators are used to assign values to variables and are follow conventional mathematical rules of precedence:

```
r = 1/8;     // assigns the value 0.125 to the variable, r
d = 2 * %pi   // assigns the value 6.28 to the variable, d
m = (60 * n)/(2 * %pi)  // Parenthesis are evaluated before division
xL = 2 * %pi * f * L;  // Inductive reactance
```

Exponent notation

Exponent notation can be used to indicate very large or very small numbers, as follows:

```
a = 1E-3;    // a = 1 × 10^-3 or 0.001
c = 2 * %pi // 2 // diameter = 2.2 × 10^6 or 200000
```

Complex numbers

Complex numbers are indicated by using %i to represent operator j as shown in the following example:

```
Z1 = 1 - %i;      // 1 - j
Z2 = 3 + (%i*2); // 3 +j2
```

Variables

SCILAB variables can be defined using statements of the form:

```
w = 2; // natural frequency in rad/sec.
df = 0.2; // damping factor
t = 0:1:10; // range of values for t 0 to 10 in steps of 1
```

Variable names can be as long and descriptive as you need but should not exceed 24 characters in length. All ASCII letters upper and lower case letters (a to z and A to Z) and digits (0 to 9) are allowed, with the additional characters %, _, #, !, $, ?. Note that the % characters is reserved for use with predefined constants as mentioned earlier. Note also that SCILAB is case sensitive, which means that upper and lower case letters are considered to be different.

Common functions

SCILAB provides the usual range of common functions including: sin, cos, tan, asin, acos, atan, abs, min, max, sqrt, and sum.

```
i = sin(0.5); // assigns the value 0.4794255 to a
t = max(2, 3, abs(-5), sin(1));  assigns the value 5 to t
```

String variables

Strings variables are delimited by pairs of single or double quotes and may be concatenated (i.e. joined) using the + operator:

```
filename = 'results.dat';
subject = "Diode " + "current";
```

Message boxes

Message boxes are a convenient way of displaying results, for example:

```
messagebox("Flux produced = " + string(phi) + " mWb");
```

Dialogue boxes

SCILAB can display a variety of different types of dialogue box, for example:

```
wait = buttondialog("Please wait","OK");
answ = buttondialog("Output the results?", "Yes|No", "User prompt");
```

User input

User input can be gained in a similar manner. The value(s) entered by a user will be returned in the evstr variable, as shown below:

```
gain = evstr(x_dialog('Required margin? ','0.235'));
```

Note that a default value (0.235) will appear when the dialogue box is displayed. Multiple inputs are also possible (see Example 1 on page 6).

Matrices

SCILAB is a powerful tool for manipulating matrices. There are several ways of entering a matrix but the most simple is as follows:

1. Separate each element in a row by a blank space or a comma
2. Separate each row of elements with a semi-colon
3. Enclose the entire list of elements in a pair of square brackets.

For example, to enter a 3 x 3 magic square and assign to the variable M:

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

The 3 x 3 matrix, M, is then:

8. 1. 6.

3. 5. 7.

4. 9. 2.

Flow control

SCILAB supports conventional program flow control using if, elseif, and end statements. The following logical operators may be used:

Operator	*Meaning*
==	Equal
~=	Not equal
>=	Greater than or equal
<=	Less than or equal
>	Greater than
<	Less than
~	Not

If a logical expression is true, it returns a Boolean variable T (true), otherwise F (false).

The if statement takes the form:

```
if condition
   action
end
```

The body will be executed only when the condition statement is true. The if statement can provide further conditional actions, as shown below:

```
if condition_1
   action_1
elseif condition_2
   action_2
elseif condition_3
   action_3
elseif ...
   ...
end
```

The three liberally commented examples that follow are designed to provide you with a quick overview of the capabilities of SCILAB. They do, however, only barely scratch the surface – SCILAB is an extremely powerful tool!

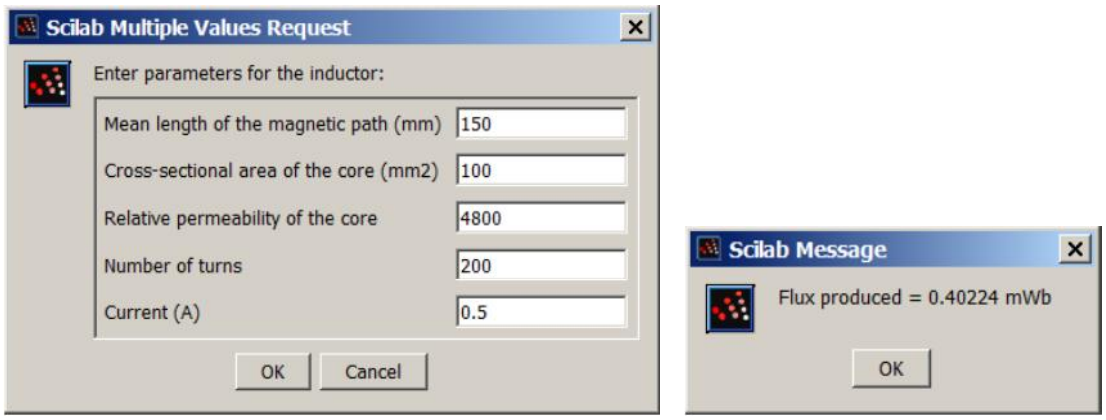
Mike Tooley

January 2012

Example 1 A simple interactive calculation

The application generates a dialogue box (SCILAB calls this a 'multiple values request box') where the user enters the five input variables. An initial default value is displayed for each variable. After the result is calculated its values is displayed in a message box.

```
// Flux produced by an inductor
// First get input data from a dialogue box
txt = ['Mean length of the magnetic path (mm)';'Cross-sectional area of the
core (mm2)';'Relative permeability of the core';'Number of turns';'Current
(A)']; // A single line of code has been wrapped here!
var = x_mdialog('Enter parameters for the
inductor:',txt,['150';'100';'4800';'200';'0.5']); // As above!
l = 1E-3*evstr(var(1));
a = 1E-6*evstr(var(2));
p = evstr(var(3));
n = evstr(var(4));
i = evstr(var(5));
// Calculate reluctance of the magnetic path
s = l/(12.57*1E-7*p*a);
// Calculate flux produced
phi =1E+3*(n * i)/s;
// Display result
messagebox("Flux produced = " + string(phi) + " mWb")
```

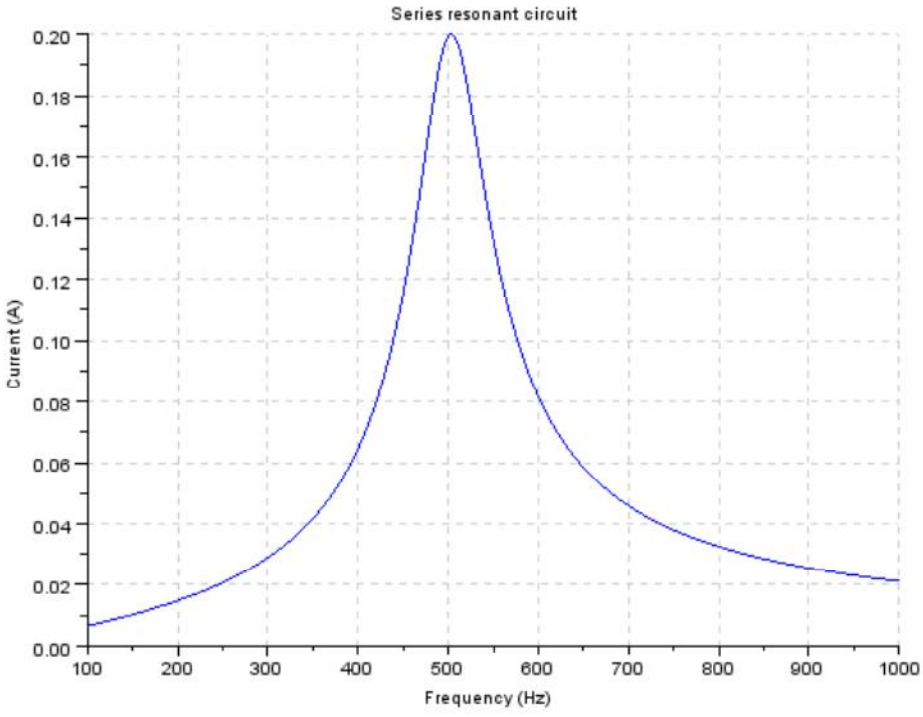


Example 2 A simple calculation with graphical output

The application uses the SCILAB plot2d function to output a graph. Note how the range of frequency values is specified in the sixth line of the script.

```
// Plot of current in a series resonant circuit
// This script plots a graph if i against f over
// the range 100 Hz to 1 kHz.
//
// First set up the table of values for frequency
f = [100:1:1000]';
// Next enter values for each of the circuit variables
```

```
// (note the use of exponent notation where appropriate)
//
C = 1E-6; // Capacitance = 1 uF
L = 100E-3 // Inductance = 1 mH
r = 50; // Resistance = 50 ohm
v = 10; // Supply = 10V
//
// Calculate the inductive reactance
xL = 2 * %pi * f * L;
//
// Calculate the capacitive reactance
xC = (2 * %pi * f * C)^-1;
//
// Calculate the impedance
z = sqrt(r^2 + (xL - xC)^2);
//
// Calculate the current
i = v * z^-1;
//
// Now plot the graph of current against frequency
// but first get rid of any earlier plot
clf();
plot2d(f,i,style=[2]);  // plot line in blue
//
xgrid(color('lightgrey'));  // place a grey grid on the plot
xtitle(['Series resonant circuit'],'Frequency (Hz)','Current (A)');
plot(f,i);  // and finally plot the curve
//
// Experiment with the effect of different values for
// r on the shape (i.e. bandwidth) of the curve. Try
// values between 10 and 200 ohm
```



Example 3 Fast Fourier analysis

This final application shows how the Fast Fourier transform, y = fft(s), can be used to display the frequency spectrum of a complex waveform:

```
//Fast Fourier analysis of a complex waveform
// Build a test signal sampled at 1000 Hz which contains to two pure
sinusoidal
// components at 50 and 70 Hz
sample_rate=1000;
t = 0:1/sample_rate:0.6;
N=size(t,'*'); // number of samples
s= 1*sin(2*%pi*50*t)+ 1/3 * sin(3*%pi*100*t+%pi/2);
// Obtain the Fourier transform
y=fft(s);
// As the fft response is symmetric only retain the first N/2 points
f=sample_rate*(0:(N/2))/N; //associated frequency vector
n=size(f,'*');
// Clear the graphic display and then plot the frequency spectrum
clf()
plot2d(f,abs(y(1:n)));
```