Getting Started

Introduction to Scilab:

Scilab is an open source software on the lines of Matlab used extensively in all fields of Engineering. Developed since 1990 by researchers from INRIA (French National Institute for Research in Computer Science and Control) and ENPC (National School of Bridges and Roads), it is now maintained and developed by Scilab Consortium since its creation in May 2003 and integrated into Digiteo Foundation in July 2008. All Scilab libraries are free with accepted and recognized licenses by the FLOSS community (Free, Libre Open Source Software). The Scilab Consortium encourages his team to contribute to third party projects which are used in Scilab. The spoken and video tutorials provided by IIT, Bombay are very informative to first time users of Scilab.

The important features of Scilab are listed below

A number of toolboxes are available with the system:

· 2-D and 3-D graphics

· Scicos : a hybrid dynamic systems modeler and simulator

· Signal processing

· Interface with C/C++

· And a large number of contributions for various domains.

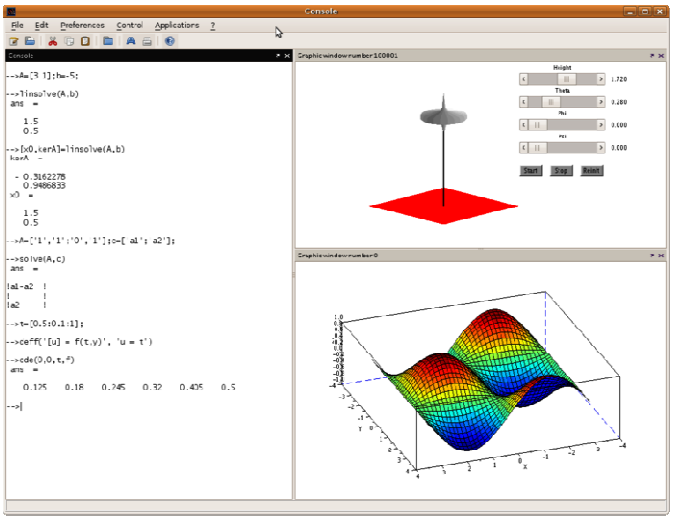


Figure 1: Scilab

Working with SCILAB

Starting Scilab opens the console window which is shown in fig 1.

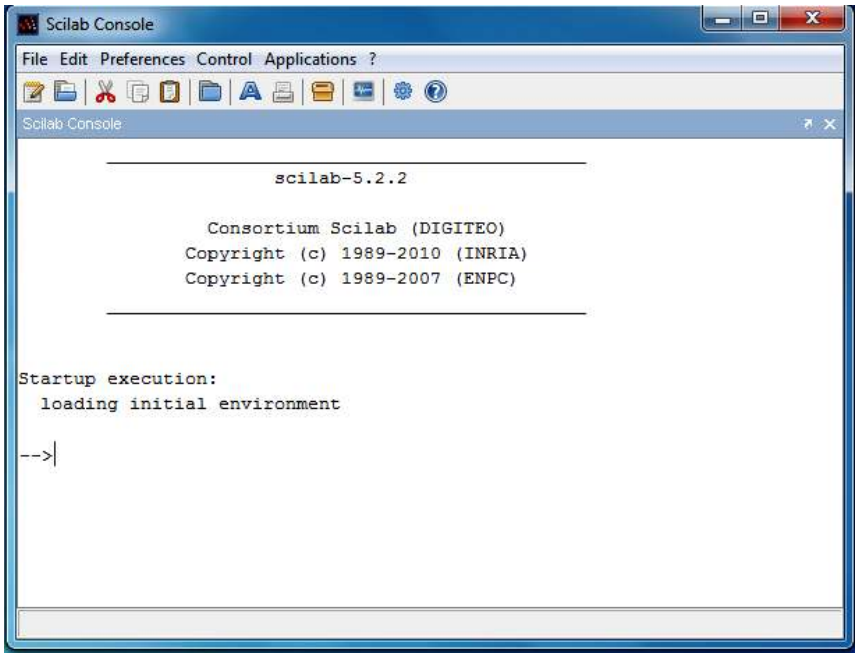


Figure 2: Scilab Console

The console menu has many options like File, Edit, Preferences, Control, Applications and Help.

The Help menu is the most useful of the lot.

HELP Menu:

To go to the help menu, just type “help” in the console window and press enter.

If you need help regarding a particular command type “ help ‘command’ ” in the console window and press enter. The help menu will open the matching results. Help menu gives us all information on the commands and functions supported by Scilab. It has two tabs, contents and search. The ‘Contents’ tab has all the supported functions arranged according to their functionality and the ‘Search’ tab helps us to search for the required function among the list.

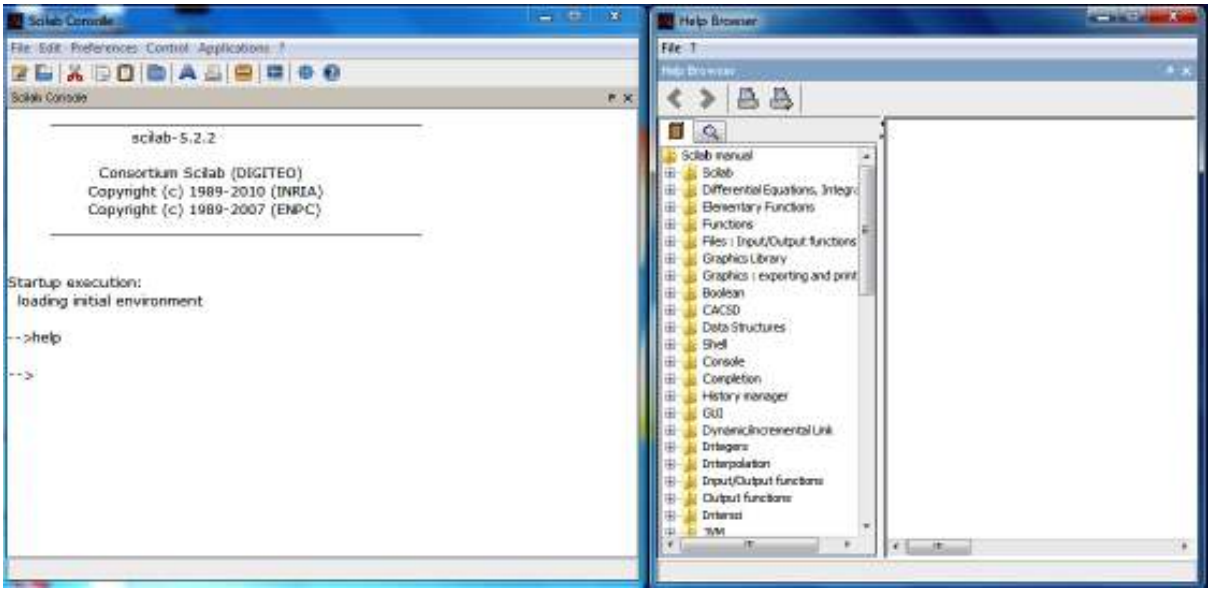


Figure 3: Help Browser

APPLICATIONS Menu:

The Applications menu gives us four options

1. Editor

2. Xcos

3. Matlab to Scilab translator

4. Module Manager – ATOMS

Figure 2 gives a pictorial representation of the applications menu.

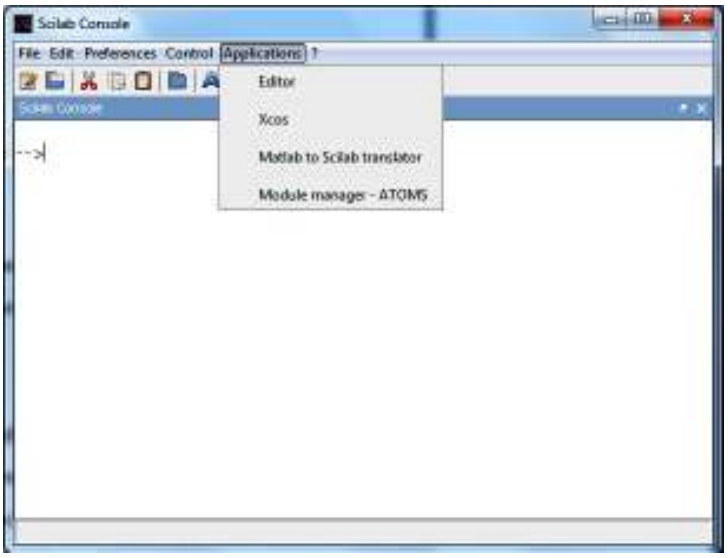


Figure 4: Applications Menu

Let us look into each one of them in detail.

EDITOR:

The editor in Scilab is used to write scripts and execute them in the Scilab console. It is opened through the applications menu of the console window i.e. Applications->Editor. On opening the Editor we go into the Scilab text editor (Figure 5).

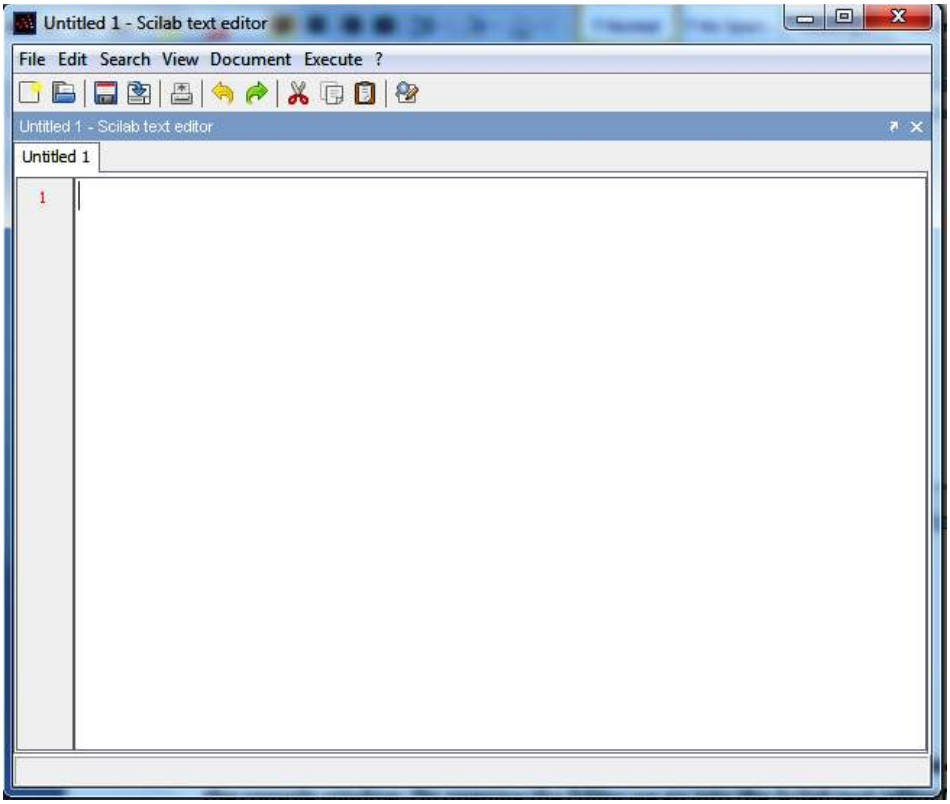


Figure 5: Scilab Editor

The Editor window has an Execute menu which helps us to run the scripts written in Scilab.

It has three options which are explained below.

1. Load into Scilab: Allows to execute the statements in the current file, as if we did a copy and paste. This implies that the statements which do not end with a ";" character will produce an output in the console.

2. Evaluate Selection: Allows to execute the statements which are currently selected.

3. Execute File Into Scilab: Allows to execute the file, as if we used the exec function. The results which are produced in the console are only those which are associated with printing functions, such as disp for example.

Programming in Scilab

Scripts in Scilab are nothing but a combination of the inbuilt functions provided by Scilab to achieve the required result. Scripts in Scilab may range from simple matrix computations to complex Signal Processing algorithms.

Some basic operators used for programming are provided below.

Symbol	Symbol Operation
[ ]	Matrix definition
;	Statement separator
'	Transpose
+	Addition
–	Subtraction
*	Multiplication
==	Equal to
~=	Not equal to
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
~	Negation
&	Element wise AND
|	Element wise OR

We next look into the tutorials where each feature of Scilab is explained with the help of examples and exercises.

TUTORIAL 1

MATRIX COMPUTATIONS

Matrix operations built into Scilab include addition, subtraction, multiplication, transpose, inversion and many more.

We will first learn to create matrices

Row matrix: A = [1 2 3]

Column matrix: A = [1; 2; 3]

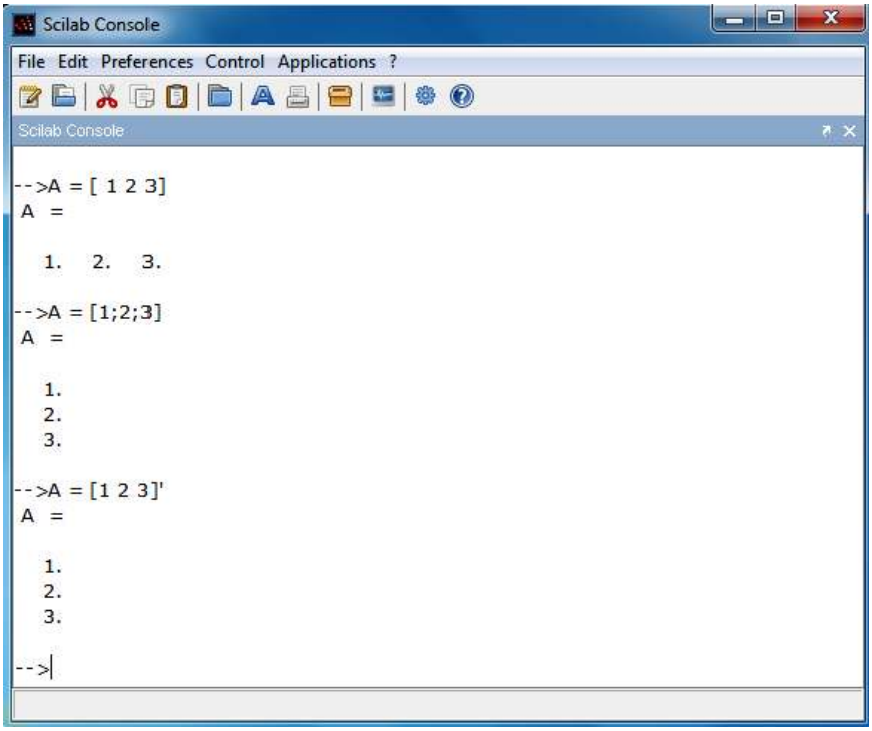
A column matrix can also be created by using the transpose operator i.e. A = [1 2 3]'

Addition, subtraction and multiplication are some of the basic functions provided by Scilab.

Multiplication can also be carried out element wise using the dot operator.

Eg: A =[1 2 3] B =[2 4 6]

A.*B = [2 8 18]



Size operator:

One of the most common errors encountered during running a script is the array size mismatch. Size operator outputs the size of the array.

Eg: A = [1 2 3]

The command size(A) gives the result 1. 3. which implies the matrix A is a row vector with 1 row and 3 columns.

Exercise:

1. Create a 3X2 matrix.

2. Try out addition, subtraction and multiplication of two matrices as an exercise.

3. Find the inverse of the matrix A = [1 0 0;1 1 1;1 1 0]. Let B= A^{-1}. Now multiply each element of B with the corresponding element of A and print the element present in the second row and third column.

TUTORIAL 2

INPUT & OUTPUT OPERATIONS

The most important feature in a script is its ability to accept input from the user and display the output on the Graphic window. Below are some most commonly input output commands.

1. **input:** The input command is used to accept an input through the keyboard. It is similar to the input command of Matlab.

Eg: i) number = input (“Enter a number”)

ii) name = input(“What is your name?”,”string”)

Note: While taking character set as inputs, we need to use the string in the input command(Eg ii) for the Editor to understand that the data being requested is a string.

2. **disp:** The disp command displays the requested data in the Scilab console window.

Eg: disp(“name”)

Exercise:

Write a Scilab program to enter a string through the keyboard and display it on the console window.

TUTORIAL 3

PLOTTING GRAPHS

The disp command has a limitation that it can only print characters or numbers. A graphical representation is thereby given by plot commands which plots the graph as a function of the independent variable.

Eg: Plotting a sine wave

```
pi = 3.14;

x= 0:0.1:2*pi

y = sin(x)

plot(y)

plot2d(y)

plot2d1(y)

plot2d2(y)

plot2d3(y)

plot2d4(y)
```

Run the above code in Scilab and see the difference in plots.

Use the help command to understand the differences.

TUTORIAL 4

CONTROL STATEMENTS

Scilab provides a series of inbuilt commands which help us in writing scripts for complex algorithms. The watch() command gives a list of all the inbuilt functions provided by Scilab. The most important of them, being the loops and the conditional statements.

Loops:

1. **for loop:** The syntax for a "for" loop is same as the one used in Matlab.

Syntax:

```
for "variable"= x:y:z
        "actions which need to be performed "
end
```

where x is the initial value, z is the final value and y is the increment size(y is optional i.e. y=1 if not mentioned) .

Example:

```
 for i=1:1:10
    disp(i);
 end;
```

2. **while loop:** While loop is used in cases where we do not know the exact number of iterations to be used. The exit from the loop is based on a condition. The syntax for the same is given below.

Syntax:

```
while "condition"
    "actions to be performed"
end
```

Example:

```
i = 4;
j=0;
while i>j
    j=j+1;
   disp(j);
end
```

Conditional statements:

Scilab supports the conditional statements like if, else, elseif.

The syntaxes for the same are given below

```
If “condition”  then
   “action”
elseif ”condition” then
    “action”
else
   “action”
end
```

Example:

```
x=1;
if x<0 then
   disp('Negative')
elseif x==0 then
    disp('Zero')
else
    disp('Positive')
end
```

Exercise:

Write a Scilab program to enter the names and marks of 4 students in an examination and display their result as either “Pass” or “Fail”.