Chapter 1: An Introduction to Scilab

Scilab is open source software and can be downloaded for installation from the web page of its developer organization viz.: https://scilab.org. The developer organization is presently owned by ESI group. Scilab is supported on UNIX, Macintosh, and Windows environments.

Scilab is an interactive software system for developed for numerical computations and graphics. It is especially designed for matrix computations: solving systems of linear equations, performing matrix transformations, factoring matrices, and so forth. The developers of Scilab have created libraries of a large number of inbuilt mathematical functions. Over that, developers have supplemented Scilab with a wide range of packages of inbuilt programs, called *toolboxes*. These toolboxes, which are collections of inbuilt programs, have been developed for solving different problems of specific areas of practical applications by following specific methods or algorithms.

Further, Scilab is developed to work as a programming language also, in a sense that the users can code their programs also just like any other programming language like C, C++ etc. Also, the inbuilt functions can be used into the users' programs. In addition, it has a variety of graphical capabilities, and can be extended through programs written in its own programming language. This feature of Scilab makes it user friendly interactive software.

Just like its commercial counterpart MATLAB, in Scilab also all the types of variables namely, real, complex, Boolean, integer, string and polynomial variables, are considered as *matrices*. Another salient feature of Scilab is that it understands the difference between real numbers and purely real complex numbers.

Scilab Advantages

- Ø It simplifies the analysis of mathematical models
- Ø It frees you from coding in lower-level languages (saves a lot of time but with some computational speed penalties)
- Ø Provides an extensible programming/visualization environment
- Ø Provides professional looking graphs

Scilab Disadvantages

The only disadvantage of Scilab over the lower level computational programming languages is that it being an interpreted (i.e., not a pre-compiled) language can turn out as slow during large scale computations.

Remark: The choice of preferring Scilab over lower level computational programming languages depends upon the requirement of its additional graphics features and availability of inbuilt programs over the scale of the data of the problem to be solved.

1.1 Getting Started

Here, we will learn about the installation and use of the software in Windows operating system. Its use is similar in other operating systems also. Installation of the software is easy like any other software. The software gets ready after its installation into a computer system. Just like any other software, Scilab can be opened by Double clicking on the Scilab icon in desktop or by clicking on the icon appearing after the entering the name Scilab in the Windows search bar. The Scilab window should come up on your screen. It looks like this:

This window is the default layout of the Scilab desktop. It is a set of tools for managing files, variables, and applications associated with Scilab.

1. The Console is a command window used for entering Scilab functions and other commands at the command line prompt appearing as -->
2. The Command History Window is used to view or execute previously run functions.
3. The Current Directory/Workspace Window lists the folders/files in the Current Directory (where you are working) or the values and attributes of the variables you have defined.
4. The Current Directory line at the top tells you where Scilab thinks your files are located. This should always point to the folder that you are working in so that your files are saved in your own directory. An example would be to enter the pathname

C:\Users\Maths50\Scilab\yourname

or use the ... button to browse for a folder.

This should always be done at the start of a new session. When you open a Scilab document, it opens in the associated tool. If the tool is not already open, it opens when you open the document and appears in the position it occupied when last used. Figures open undocked, regardless of the last position occupied.

5. Scilab provides a variable browser, which displays the list of variables currently used in the environment.

6. The Editor, named as SciNotes, is used to access and edit Scilab program files. The Scilab program files are called script files. The editor can be accessed from the menu of the console, under the Applications > SciNotes menu, or from the console, as presented in the following session.

```
--> editor()
```

7. Script Files: Script files are normal ASCII (text) files that contain Scilab commands. There are two types of script files in Scilab, namely, programs and functions. It is essential to suffix an appropriate extension name after these files. Extension name for program files is ".sce" (e.g., scriptname.sce) and for functions is ".sci" (e.g., functionscript.sci).

8. Executing a Script file

A script file can be executed using exec command, for example:

```
-->exec('D:\Puducherry\Class_2021_10_12_for_Loop.sce', -1)
```

1.2 Using Scilab as a calculator

The basic arithmetic operators are + for addition, - for subtraction, * for multiplication, / for division, ^ for exponentiation, and these are used in conjunction with braces or commonly called round brackets ( ). The symbol ^ is used to get exponents (powers): 2^4 = 16. An alternative symbol used for the same purpose is **.

Example:

```
--> 2+3/4*5 ans =

         5.7500
```

Note that in this calculation the result was 2+(3/4)*5 and not 2+3/(4*5) because Scilab works according to the priorities of operations given in the following order.

1. quantities in brackets ( )
2. powers or exponent ^ or **
3. multiplication *, left division /, and right division \, working left to right
4. addition + and subtraction -, working left to right

1.3 Basic Elements of Scilab as a Programming Language

=

=

```
-->x=1   x

=

        1.
```

There are rules of defining variable names for their use in Scilab. These are given below.

Variable Names

Variable names may be as long as the user wants, but only the first 24 characters are taken into account in Scilab. For consistency, we should consider only variable names which are not made of more than 24 characters. All ASCII letters from "a" to "z", from "A" to "Z" and from "0" to "9" are allowed, with the additional letters "%", "_", "#", "!", "$", "?".

Caution: Variable names for which the first letter is "%" have a special meaning in Scilab. These represent the mathematical pre-defined variables, which are discussed in a later section.

Variable names which are allowed and not allowed in Scilab are illustrated below.

Allowed: NetCost, Left2Pay, x3, X3, z25c5

Not allowed: Net-Cost, 2pay, %x, @sign

Pre-defined mathematical variables

Variable Name	**Description**
%pi	the mathematical constant π
%e	Euler’s constant e
%i	the imaginary number i

Input and output of Mathematical values in Scilab

Apart from the real numbers (expressed in the natural way), complex numbers and Booleans are provided as inputs to Scilab in the formats given below.

Complex numbers
Example: The complex number $2 + 3i$ is input in Scilab as: 2 + 3 * %i .

Booleans
The truth value “True” is input in Scilab by using %t or %T, whereas the truth value “False” is input using %f or %F.

Both these are discussed in detail in later sections.

Strings

String is a sequence of characters. All the characters including all letter in both cases (i.e., from a to z and from A to Z), digits from 0 to 9 can be used in a string. Strings can be defined and then stored to any variable names by delimiting them in double quotes “ ” . Let us learn how to define strings through examples given below.

```
-->A = "Scilab"

    A   =

Scilab

-->B = "Software"

    B   =

Software
```

Remarks:

- Strings have no direct mathematical use.
- They are used mainly for displaying a lingual message as a part of the output upon the execution of a program.

The concatenation (*i.e.*, join) of two strings can be done by using the concatenation operator (+). The use of concatenation operator is demonstrated below through a Scilab session.

```
-->"Scilab" + "Software"  ans  =

ScilabSoftware

-->A+B  ans  =

ScilabSoftware

-->A + " " + B  ans

=

 Scilab Software
```

It is to be observed that only a string can be concatenated with a string. The concatenation of a string with a number is not possible. For this purpose, a function “string” is available in Scilab which converts a number to the string corresponding to that number. This function can be used appropriately in a situation when there is a requirement of displaying an output message including the output value of some computation also. For example,

```
-->a = 1  a  =

         1.

-->b = 2  b  =

   2.

-->c = a + b  c  =

         3.

-->"The sum of " + string(a) + " and " + string(b) + " is " + string(c) + "."  ans  =

  The sum of 1 and 2 is 3.
```

Suppressing the display of output

The output of any command can be suppressed by ending the command that particular command with semicolon ";". For example, the Scilab command in console

```
-->x = 2
```

is returned with a display of the action performed as following

```
x  =

         2.
```

Whereas, the Scilab command

```
-->x = 2;
```

does not display the action performed but will keep output in the computer's temporary memory being used, in exactly same way as was done in the previous

command. The value of this output for the variable x will remain there in the temporary memory
(RAM) for further use, unless it changes by reassigning some other value to this variable or the Scilab session is closed. Irrespective of the display of the output, the value assigned to the variable x can accessed any time from the variable browser in both the cases discussed above.

Dynamic nature of variable in Scilab

As Scilab is an *interpreted language*, therefore it allows dealing with variables in a dynamic way. The dynamic nature being referred here is in the sense that a variable set to a value of one particular type can later be used to reassign a value of different type also. The same is illustrated below in a Scilab session in Console.

```
-->x = 2 + 3 * %i  x  =

   2. + 3.i   -->y =

4*x  y  =

    8. + 12.i   -->x =

5  x  =

   5.   -->y =

2*x  y  =

         10.
```

Comments and continuation lines

Any text that follows // in a line is ignored by the compiler. The main purpose of this facility is to enable inserting comments in the script files. Inserting comments in a script file helps the programmer to read the code of a program with the help of summary messages about the commands. This provision of putting any line as a comment can be exploited even for editing or debugging script files also.

Commands which are too long to be typed in a single line can be continued in multiple subsequent lines by putting *two dots* at the end of each previous line. In Scilab, any line which ends with two dots is considered to be the start of a new continuation line.

In the following Scilab session in Console, we give examples of Scilab comments and continuation lines.

```
-->// This is my comment .

-->x =1..

-->+2..

-->+3..

-->+4  x

=

        10.
```

Remark: At this stage of introduction of Scilab, the use of comments and continuation lines is demonstrated here through Console, but they are more justifiably used in script file also.

1.4 Elementary mathematical functions

Scilab has in-built elementary mathematical functions for their direct use into computations. Most of these functions take one input argument and return one output argument. These functions are vectorized in the sense that their input and output arguments are matrices. This allows computing data with higher performance, without any loop. The list of elementary mathematical functions is present in following tables.[1]

Function Name	Syntax	Type of variable as input argument	Description
exp	exp(X)	scalar, vector, or vector (real or complex entries)	exp(X) is the (element-wise) exponential of the entries of X.
expm	expm(X)	a square matrix with real or complex entries.	
log	log(x)	scalar, vector or matrix	log(x) is the "element-wise"

[1] A detailed description of these functions can be accessed from the help document of Scilab, which is available within the software and on the website of the Scilab developers as well.

A detailed description of these functions can be accessed from the help document of Scilab, which is available within the software and on the website of the Scilab developers as well.

log10	log10(x)	scalar,	base 10 logarithm
log1p	log1p(x)	scalar,	
log2	log2(x)	scalar,	

logm	logm(x)		logm(x) is the matrix logarithm of x. The result is complex if x is not positive or definite positive. If x is a symmetric matrix, then calculation is made by Schur form. Otherwise, x is assumed diagonalizable. One has expm(logm(x))=x.
max	max(A)		maximum value of matrix A
min	min(A)		minimum value of matrix A
modulo	modulo(n,m)	integers	remainder of n divided by m (n and m integers)
pmodulo	pmodulo(n,m)	integers	positive arithmetic remainder of n divided by m (n and m integers)
sign	sign(A)	real complex or matrix	returns the matrix made of the signs of A(i,j). For complex A, sign(A) = A./abs(A).
signm	signm(A)	real complex or matrix	for square and Hermitian matrices X=signm(A) is matrix signum function.
sqrt	sqrt(x)	real or complex scalar or vector	returns the vector of the square root of the elements. Result is complex if x is negative.
sqrtm	sqrtm(x)	real or complex square matrix	the matrix square root of the x x matrix (x=y^2) Result may not be accurate
			if x is not symmetric

Remark: All the elementary mathematical functions listed above are discussed as real functions or real vector valued functions. Those mathematical functions which are extended from real to complex functions (for example, trigonometric, exponential and logarithmic functions) have same function names. This means that, if their input argument is a complex number, the same function returns the output as a complex number, behaving as a complex function. This feature of Scilab is phrased as "elementary functions in Scilab are overloaded for complex numbers". The following Scilab session illustrates this feature.

```
-->y = sin(%pi/2)  y  =

         1.

 -->w = sin(2 + 3 * %i)  w  =

         9.1544991 - 4.168907i
```

Some other functions provided in Scilab which help managing complex numbers are provided below.

Functions to manage complex numbers

Another salient feature of Scilab is that it understands the difference between real numbers and purely real complex numbers.

Function Name	**Description**
real	gives the real part of complex number
imag	gives the imaginary part of complex number
imult	performs multiplication of number in input with *i*
isreal	returns true if the variable has no complex entry

Following Scilab session demonstrates the use of these functions.

```
--> z = 2 + 3 * %i
```

```
 z  =

          2. + 3.i

     2. + 3.i

 -->x = real(z)  x  =

2.         -->y = imag(z)

y  =

3.        -->z1 = imult(z)

z1  =

  - 3. + 2.i

-->isreal(z)  ans

=

  F   -->isreal(2)  ans  =

  T
```

Remark: Scilab distinguishes between real and purely real complex numbers. This feature can be verified by appropriately using the isreal function as demonstrated below.

```
-->z2 = 2 + 0 * %i  z2  =

          2.

-->isreal(z2)  ans

=         F

-->isreal(2)  ans  =

   T
```

Booleans

Some comparison operators and logical connective operators available in Scilab are described given below.

Comparison operators

Operator	Use	Description
==	a==b	returns truth value true if expression a is equal to b; otherwise false
~= or <>	a~=b or a<>b	returns truth value true if expression a is not equal to b; otherwise false
<	a<b	returns truth value true if a real expression a is less than b; otherwise false
>	a>b	returns truth value true if a real expression a is greater than b; otherwise false
<=	a<=b	returns truth value true if a real expression a is less than or equal to b; otherwise false
>=	a>=b	returns truth value true if a expression a is greater than or equal to b; otherwise false

Logical connective operators

Operator	Use	Description
&	A&B	logical **AND** operator: returns truth value true only for the case when both the logical expressions A and B have truth value true
|	A|B	logical **OR** operator: returns truth value true for when at least one of the logical expressions A and B have truth value true
~	~A	logical **NOT** operator: returns truth value true if the logical expressions A has truth value false and vice-versa

The use of all the operators is demonstrated through a Scilab session in Console, as given below.

```
-->x = 1;

-->y = 2; -->z = x +

y; z  =

          3.

-->z==3 ans

=        T

-->x<y  ans

=   T

-->x<=y  ans

=   T

-->x>z  ans

=

  F   -->x>=1

ans  =   T

-->x~=1  ans

=   F

-->x<>1  ans

=

  F

-->z = (x==1) & (y==2)  z  =
  T

-->z1 = (x==1) | (y==3)  z1  =

T   -->z2 = ~z1  z2  =       F
```

Strings

String is an array of characters. To represent a string in Scilab, a set of characters is enclosed within double quotes (for example, “A string in Scilab”).

- Strings can be stored in variables by using assignment operator, just as other values of real or complex numbers can be stored.
- Concatenation (i.e., joining) of strings is done by using the concatenation operator +. The following Scilab session demonstrates the concatenation of two strings.

```
-->x = "String "  x  =

String

-->y = "Concatenation"  y  =

Concatenation

-->z = x + y  z  =

String Concatenation
```

- The Scilab in-built function string gives the output as a string of numeric characters corresponding to any number supplied as input. Its use is demonstrated through a Scilab session in console.

```
-->m = 2;

-->n = 4;
```

```
-->string(m) + " divides " + string(n) ans  =

2 divides 4
```

Exercise 1 In the console:

1. Use assignment operator for creating and setting (assigning) variables to float value, string, Boolean values,
2. Use of comment and continuation line,
3. Use of inbuilt Mathematical function and operators,
4. Use of pre-defined Mathematical variables,
5. Use of Booleans and comparison operators,
6. Use of complex numbers, and operations on them,
7. Use of strings and concatenation operator, and use of comparison operator ‘==’ for comparison of two strings for checking their equality.
8. Swap the values assigned to two variables without using third variable.