# Some Useful Public domain Softwares

Key words: Public domain softwares, Unix, Gromacs, Gamess, installation of Scilab, Help menu, roots of equations, interpolation, matrix operations, diagonalization, differential equations, curve plotting, optimization, curve fitting

## 8.1 Introduction

In the present chapter, you will be introduced to some very useful public domain software which helps in carrying out the common numerical tasks and also for plotting the experimental data. One is SCILAB which helps in numerical techniques and has a vast help menu. This can be downloaded in UNIX as well as windows environments. Another is the software Xmgrace which is excellent for plotting data and is presently available in UNIX. We have already used the software Graph 4.3 in Chapter 4 wherein we studied interpolation. Avogadro is useful software for plotting and viewing molecular clusters as well as large molecules. Gamess and Gromacs are very powerful software that are useful for doing ***ab initio*** calculations and bimolecular simulations respectively. This list of public domain software will keep growing with time and you can search on the web for additional software. A useful website to know about the basic microscopic properties of a few molecules (such as energy, dipole moment, polarizability and so on) is http://www.chemeddl.org/collections/molecules.

## 8.2 SCILAB Introduction

**Scilab is freely downloadable from the link http://www.scilab.org/ Download scilab to your computer and install it.**

When we click on the Scilab icon, we can see the window as shown in the following figure (Fig. 1). We have to enter the Scilab command at the prompt symbol (-->) which is shown in the Fig. 1. Many of commands are available at the menu bar (top of the window). The most important command for the beginner is the help command which is located at the menu bar as question mark symbol (?). By clicking on the help menu, you can see the Scilab manual with lot of commands and examples for each command which is shown in fig.2.

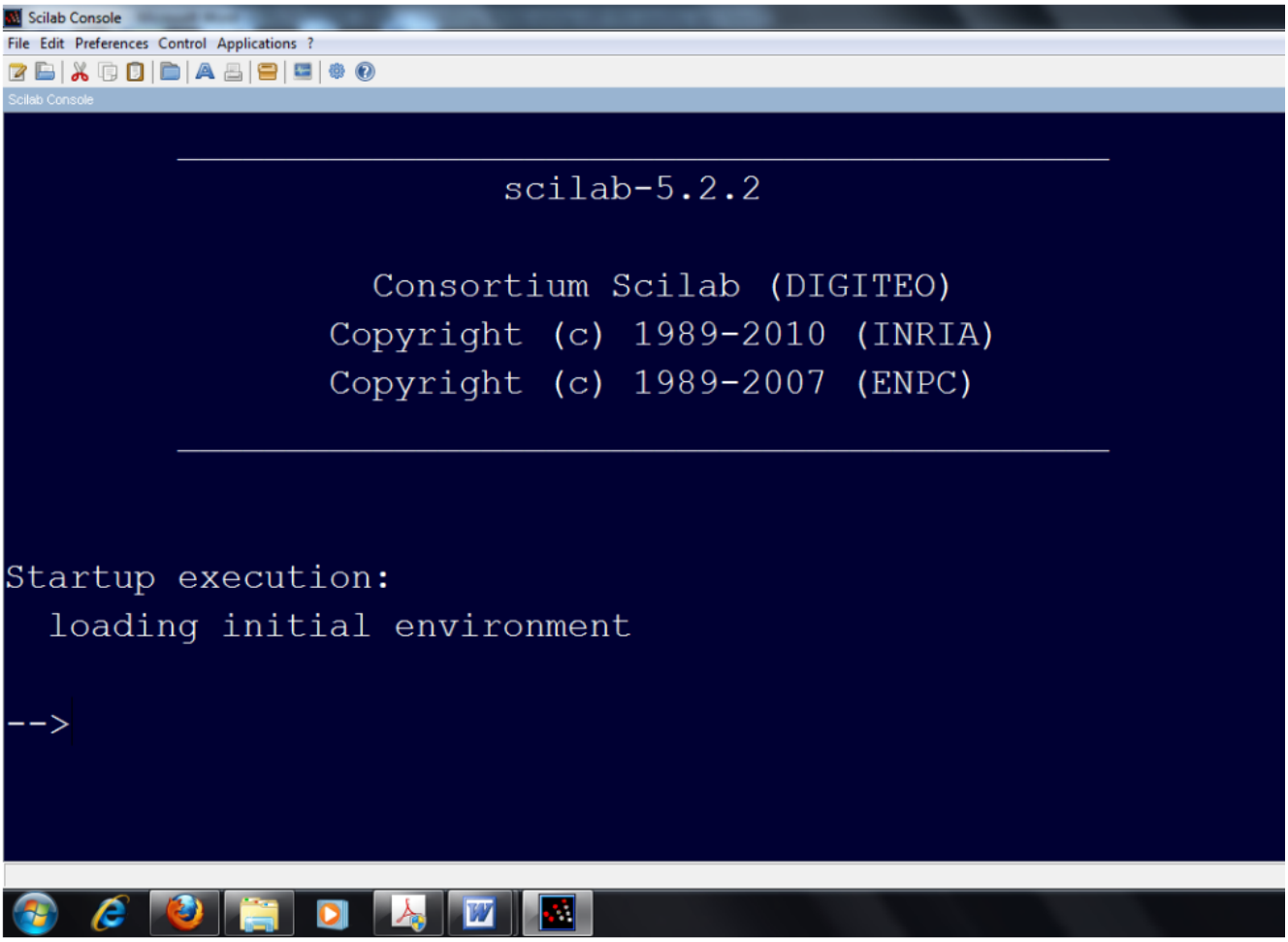



Fig. 1. Scilab window.

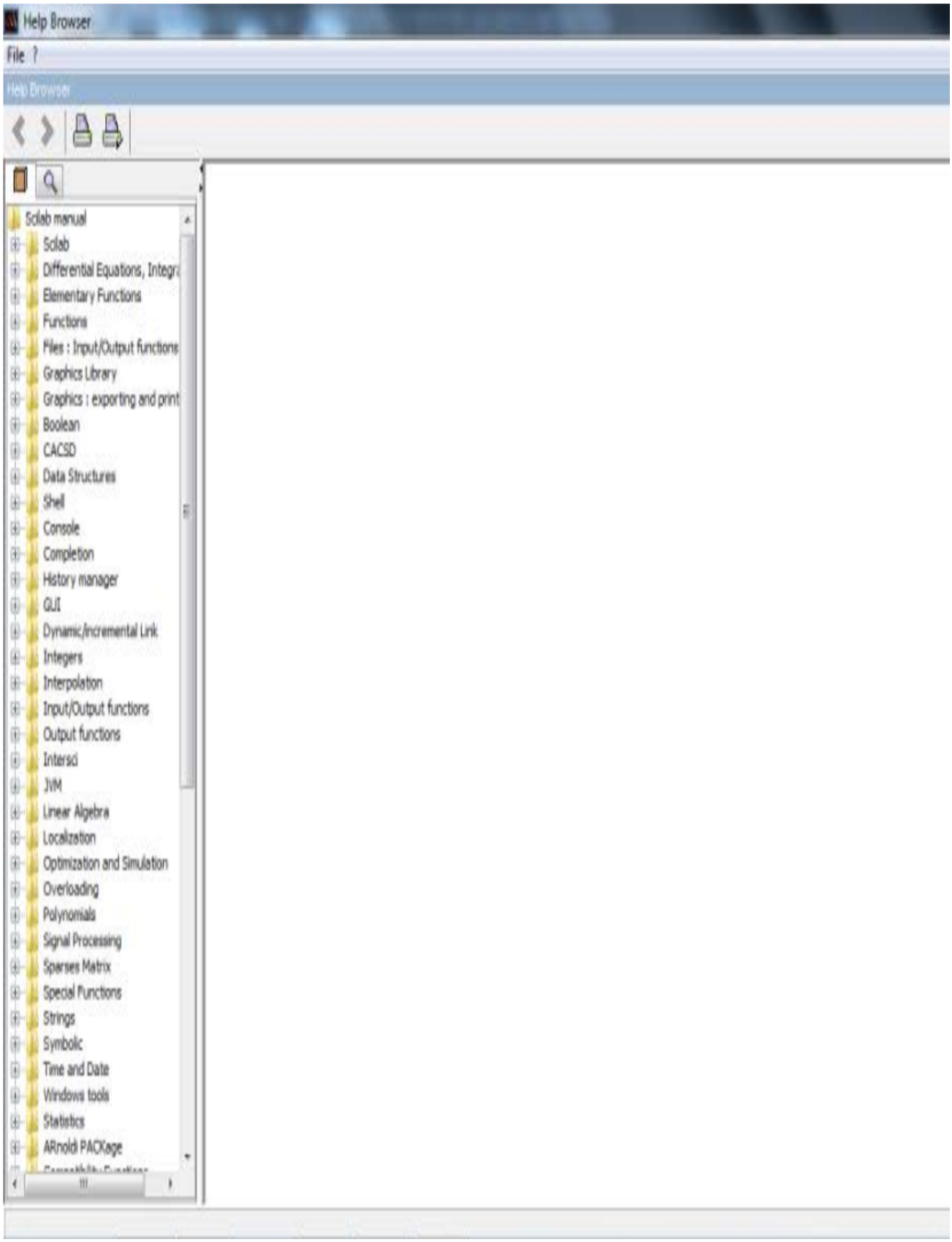

Help Browser
File ?
Help Browser
Scilab manual
Scilab
Differential Equations, Integr
Elementary Functions
Functions
Files : Input/Output functions
Graphics Library
Graphics : exporting and print
Boolean
CACSD
Data Structures
Shell
Console
Completion
History manager
GUI
Dynamic/Incremental Link
Integers
Interpolation
Input/Output functions
Output functions
Intersci
JVM
Linear Algebra
Localization
Optimization and Simulation
Overloading
Polynomials
Signal Processing
Sparses Matrix
Special Functions
Strings
Symbolic
Time and Date
Windows tools
Statistics
ARnoldi PACKage

Fig. 2. Scilab help menu.

Also, we can get the help on any command by writing 'help command' at command line. e.g. if you want to get help for function such as 'abs' then you have to write as following:

```
-->help abs
```

You can do simple calculations in scilab as follows.

## 8.3 Simple Operations using SCILAB

```
-->a=2

-->b=2

-->a + b
```

Then you get the out put as

ans =

4.

Before going into further discussion about numerical calculations with scilab, let us discuss about some very useful commands helpful for using Scilab.

***pwd*** – it will print the current working directory of scilab (when you start scilab, the default directory is C:\Program Files (x86)\scilab-5.2.2

***ls*** - list all the files in the current working directory

***cd or chdir*** – these commands will change the current working directory. e.g. if you want to change current directory to scilab folder (directory) in D drive then you have to write

as ***cd D:\scilab***

***diary*** – Saves the entire content of the scilab work space to one file.

e.g. ***diary ('content.txt')*** will save the entire content of the scilab work space to content.txt. When enter the command ***diary('content.txt')***, you will be prompted as

ans =

1

Then you may type whatever commands you want to execute.

After completing the current work, we need to close the directory by using the command ***diary(0)***. You will now find that the file content.txt is saved in the current working directory. In place of content.txt, you may choose any other file name.

We will discuss the other useful commands in the corresponding sections in this chapter.

Scilab can be used as a scientific calculator. Look at the following information

```
-->a=2 ; b= 3; c=8;

--> a * b *c

-->ans =

48.

-->sum(1:10)

-->ans =

55.
```

```
-->prod(1:10)

-->ans  =

3628800.

-->sqrt(456)

-->ans =

21.354157

-->x = sin(30)

-->x =

- 0.9880316 //(value radians)
```

In scilab, some predefined constants are defined as follows.

***%i*** = imaginary number 'i' = sqrt (-1)

***%e*** = Euler's constant=2.7182818

***%pi*** = π =3.1415927 In Scilab π value is in radians only.

***%t*** and ***%f*** are Boolean constants that are used to indicate true and false.

***%f = ~%t***

***%inf*** is used to indicate an infinite number...

***%nan*** is used to indicate not a number

In scilab, // symbol indicates comment (i.e., whatever is entered after // will not read in Scilab).

## 8.4 Calculations with Matrices

--> ***a = [7 2 4; 1 2 4; 7 8 9]*** // this is a way to define a 3 x 3 matrix in Scilab.

--> ***b = a'//*** is the transpose of the matrix ***a,*** prime is denoted by ***'***

--> ***a * b // matrix*** multiplication

--> ***c= inv(a)*** it will give the inverse of the matrix ***a***

--> ***d = det(a)*** it will give the determinant of matrix ***a***

Diagonalization of matrices is very easy using scilab. Look at the following

-->A = [1 4 3; 0 2 5; 1 3 -4]

The diagonal form of matrix A, $A_D$ is given by $X^{-1}$ A X. Here, X is the matrix formed by collecting all the eigenvectors (in the form of columns) of the matrix into a single matrix.

The Scilab command for getting the eigenvalues and eigenvectors is:

--> ***[Lam, X] = bdiag (A)***//Lam is $A_D$ (the diagonal form of matrix A)

**X = - 1.8071512 1.1659151 - 0.0630461**

**0.4062757 0.7586045 - 0.5671129**

**- 0.1359822 0.3988603 0.9023204**

**Lam = 0.3264781 0. 0.**

**0. 4.628908 0.**

**0. 0. - 5.9553861**

Here, in place of Lam and X, we can use any other variable names. Typing ***bdiag(A)*** only gives Lam (i.e., $A_D$ )

//The eigenvalues can be obtained using the command "***spec***".

Eigenvals = ***spec (A)***

```
EigenVals  =

  - 5.9553861

    4.628908

    0.3264781
```

To get an identity matrix, the command is ***eye (5, 5).*** Here, *5 x 5* is dimension of matrix

```
-->eye (5, 5)

ans  =

   1.   0.   0.   0.   0.
   0.   1.   0.   0.   0.
   0.   0.   1.   0.   0.
   0.   0.   0.   1.   0.
   0.   0.   0.   0.   1.
```

The command ***zeros (4, 8)*** gives a null matrix of 4 rows and 8 columns. It should be noted that the size of dimensions is not unlimited and is determined by your computer memory.

## 8.5 Functions with Scilab

To obtain a simple integral over a sine function over the range 0 to $\pi$,

```
-->x0=0;x1=%pi;//range of x is from x=x0 to x= x1

-->X=integrate ('sin(x)','x', x0, x1)//X is the value of the definite integral

(1)
```

Here sin(x) is our function. We can integrate any other function in the limiting range ***x0 to x1.***

You can define a function by using the following commands.

Suppose dy/dt=y^2-y sin(t)+cos(t), y(0)=0 is the differential equation that we need to solve.

```
 //to solve a ode(ordinary differential equation)
```

You have to write function like this

-->***function ydot=f(x,y),ydot=y^2-y*sin(x)+cos(x),endfunction***

-->***y0=0;x0=0;x=0:0.1:%pi;***

Here y0 and x0 are the initial conditions. Now we require the solution of above ode and range of x from 0 to π with the spacing of 0.1.

To get the solution, you have to write a command like this.

-->***y=ode (y0, x0, x, f)***//this gives the answer y in the form of a columns

```
y =

  Column 1 to 12

   0.   0.0998334   0.1986694   0.2955202   0.3894184   0.4794257   0.5646425
0.6442177   0.7173561   0.7833270   0.8414710   0.8912074

      Column 13 to 24

   0.9320391   0.9635582   0.9854498   0.9974951   0.9995737   0.9916649
0.9738477   0.9463002   0.9092975   0.8632095   0.8084966   0.7457055

      Column 25 to 32

   0.6754635   0.5984725   0.5155018   0.4273803   0.3349886   0.2392498
0.1411205   0.0415812
```

You can also ask the answer in a single line

-->***y=ode (y0, x0, x, f)'*** //this gives y in a single column. // ***(3)***

```
y =
   0.
   0.0998334
   0.1986694
   0.2955202
   0.3894184
   0.4794257
   0.5646425
   0.6442177
   0.7173561
   0.7833270
   0.8414710
```

**0.8912074**
**0.9320391**
**0.9635582**
**0.9854498**
**0.9974951**
**0.9995737**
**0.9916649**
**0.9738477**
**0.9463002**
**0.9092975**
**0.8632095**
**0.8084966**
**0.7457055**
**0.6754635**
**0.5984725**
**0.5155018**
**0.4273803**
**0.3349886**
**0.2392498**
**0.1411205**
**0.0415812**

## 8.5 Scilab programming

Scilab allows programming commands similar to other programming languages. There is no compilation as in FORTRAN, but simply execution of the commands in the Scilab environment.

-->***x=1:20;***

-->***for i=1:10,y(i)=x(i)+2;end;***

Here ***for*** is a loop like do loop in FORTRAN.

```
-->y
 y =

    3.
    4.
    5.
    6.
    7.
    8.
    9.
    10.
```

11.
12

***Integration using Trapezoidal rule:***

```
Function approx = trapez (a,b,n,func)

h = (b-a)/n; sum trap = 0; // initialize for trapezoidal rule

for i = 1:n,

sum _trap = sum trap + func(a+(i-1)*h) + func(a+i*h);end;

approx = sum_trap*h/2;

endfunction

function value=func(x), value=sin(x), endfunction     //              (4)
```

```
x= 0.000000
0.500000
1.000000
1.500000
2.000000
2.500000
3.000000
3.500000
4.000000
4.500000
5.000000
5.500000
6.000000
6.500000
7.000000
7.500000
8.000000
8.500000
9.000000
9.500000
10.00000
```

```
func=
0.000000
0.479426
```

```
0.841471
0.997495
0.909297
0.598472
0.141120
-0.350783
-0.756802
-0.977530
-0.958924
-0.705540
-0.279415
0.215120
0.656987
0.938000
0.989358
0.798487
0.412118
-0.075151
-0.544021
a=1;
b=10;
n=20;

approx  =
 2.7629658
```

We will discuss some more programs after learning how to draw graphs using scilab.

## 8.6 Graphs

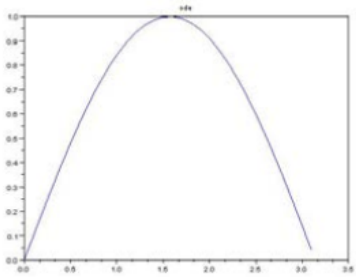
To draw a plot for x vs. y the command is plot(x, y). In Scilab, there are two types of plots. You can invoke them by using following commands.

***plot2d ()***

***plot3d ()***

Now let us plot for function (2) given above.

***Plot(x, y)*** gives a 2d plot. The plot will look like this.



You can give a title to the plot as shown in the above figure by giving a command like,

xtitle ('ode');

You can give the legends to the plot by the command

***legend ('y');***

And you can save the plot in your directory by giving ***xs2jpg (0,'ode')***

## 8.7 Legendre polynomials with Scilab

The recursion formula for Legendre polynomials is

$$P_n(x) = ((2n-1)xP_{n-1}(x) - (n-1)P_{n-2}(x))/n$$

The first few Legendre polynomials are:

| n | $P_n(x)$ |
|---|---|
| 0 | $1$ |
| 1 | $x$ |
| 2 | $\frac{1}{2}(3x^2-1)$ |
| 3 | $\frac{1}{2}(5x^3-3x)$ |
| 4 | $\frac{1}{8}(35x^4-30x^2+3)$ |

5 $\frac{1}{8}(63x^5 - 70x^3 + 15x)$

6 $\frac{1}{16}(231x^6 - 315x^4 + 105x^2 - 5)$

7 $\frac{1}{16}(429x^7 - 693x^5 + 315x^3 - 35x)$

8 $\frac{1}{128}(6435x^8 - 12012x^6 + 6930x^4 - 1260x^2 + 35)$

9 $\frac{1}{128}(12155x^9 - 25740x^7 + 18018x^5 - 4620x^3 + 315x)$

10 $\frac{1}{256}(46189x^{10} - 109395x^8 + 90090x^6 - 30030x^4 + 3465x^2 - 63)$

e.g. Scilab code to draw 3rd polynomials is

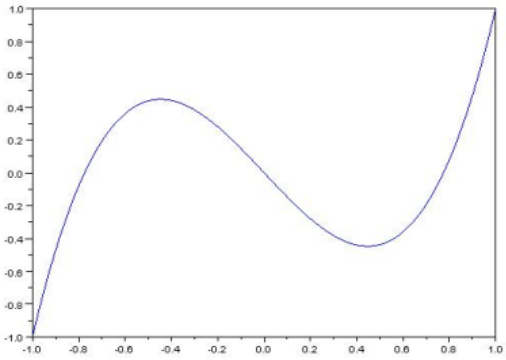
***->x=(-1:0.01:1);***

***-->for i=1:201, y(i)=((5*x(i)^3)-3*x(i))/2;end;***

***--> y=***

Then you can plot using the command

***Plot(x, y)*** will give the following plot




We can write all commands in one page using the command called ***editor*** command. When you type ***editor*** on Scilab screen, a new window will appear where you

can write all the commands and going to execute later. To execute the editor file, you have to write ***exec filename.sce***.

e.g., Using this command, you can plot all the above 10 Legendre polynomials at once. You have to write Legendre polynomials as follows..

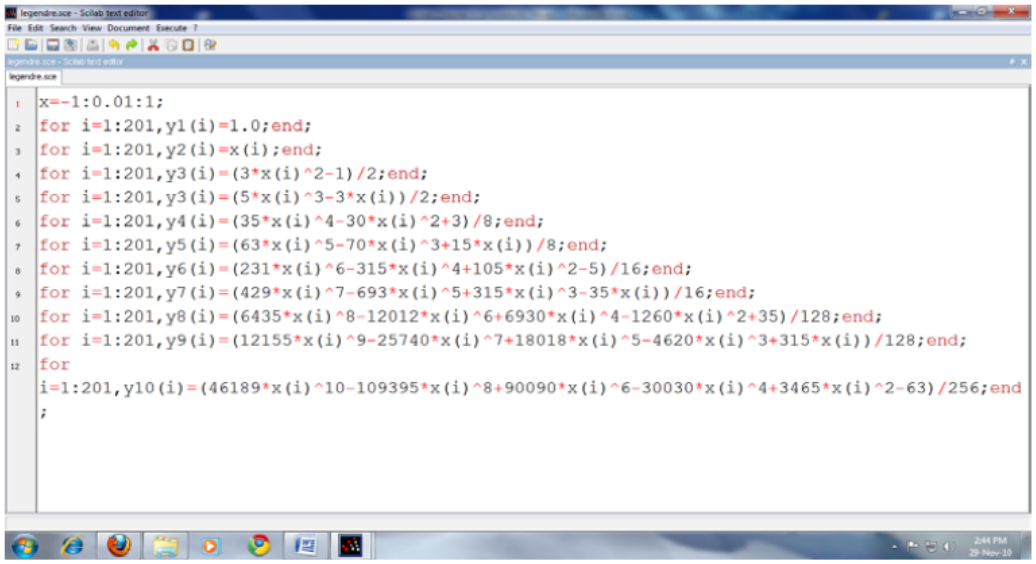



Fig 7: The window of the files which calculates the 10 Legendre polynomials.

You have to save the file with legendre***.sce*** file extension. And call back by the command ***exec legendre.sce*** and then you can plot using the command,

***plot2d(x, [y1 y2 y3 y4 y5 y6 y7 y8 y9 y10]);***

## 8.8 Curve fitting using Scilab

Fitting functions to a given data set.

X, Y data is given. You need to write a Scilab program as follows.

The program given below is for an exponential fit

```
X= (0.1:0.2:2.1);
Y=[0.9,0.75,0.6,0.5,0.4,0.33,0.27,0.20,0.18,0.15,0.13];
function y=FF(x,p), y=exp(-p*x), endfunction
```

```
Z=[Y;X];
//The criterion function
function e=G(p,z),
  y=z(1),x=z(2);
  e=y-FF(x,p),
endfunction
//Solve the problem
p0=0.8
[p, err]= datafit(G,Z,p0);
scf(0);clf()
plot2d(X, Y,-1) // the plot with data given by us.
plot2d(X,FF(X,p),12) //the plot with fitting function.
```

In the above program FF is the fitting function, here it is exponential function. Here Z is 2 x n matrix (here n is number of data points given) and two rows corresponds to Y and X data. Therefore you need to give the data of X and Y, in a single row (i.e. X and Y should be 1 x n matrices). In the above program the Z is looks like this

```
Z =
   0.9   0.75   0.6   0.5   0.4   0.33   0.27   0.2   0.18   0.15   0.13
   0.1   0.3    0.5   0.7   0.9   1.1     1.3    1.5   1.7     1.9    2.1
```

Then you need to write criterion function as shown in the program to define error bars. But we can't see any variables of this function on the screen. Now you have to give the initial p value p0. Then it starts iteration using the command [p, err] = datafit (G, Z, p0); Here we can see the final p and err values on the screen. Then you have to plot according to commands given in the above program.

Example 2: Program for linear least squares fitting

```
X= (0.1:0.2:2.1);
Y= [0.098,0.31,0.48,0.71,0.91,1.08,1.31,1.50,1.72,1.88,2.15];
function y=FF(x,p), y=p(1)*x+p(2),endfunction
Z=[Y;X];
//The criterion function
function e=G(p,z),
  y=z(1),x=z(2);
  e=y-FF(x,p),
endfunction
//Solve the problem
p0=[50;-100]
[p,err]=datafit(G,Z,p0);
scf(0);clf()
plot2d(X,Y,-1)  // the plot with data given by us.
plot2d(X,FF(X,p),12) // the plot with fitting function.
```

To save a particular variable you have to use the command ***fprintfMat***

e.g. ***fprintfMat('filename.txt', variable)*** Here, .txt is the type of the file to be written to disc., it can be .doc, .docx also. Here, variable is the variable which is on the screen and what you want to save. You can call back the same file by using the command ***fscanfMat***

e. g. ***variable = fscanfMat('filename.txt')*** Here variable is anything which you desire. Most important thing is the format should be the same which you have saved earlier.

The ***fscanfMat*** command is very useful because you can call your own data which are stored in one file (txt or doc etc.).

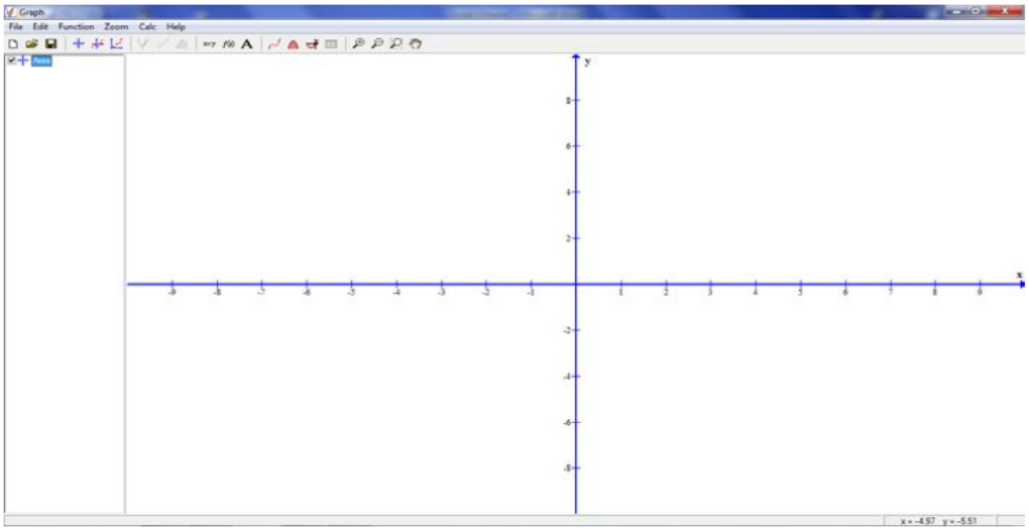
E.g. In the above two programs, the data are given as X and Y. You can also call from your own data files X.txt and Y.txt using the ***fscanfMat*** command. Sometimes, there are problems with these fprintfmat statements. In that case, go to the help on these commands. The files are saved in temporary directories and with some effort, you will be able to access these directories. Some patience and considerable practice will help and the benefits are enormous and all this is free!

**Another useful software for drawing graphs and fitting functions is GRAPH 4.3.** It is a freeware program for drawing graphs in a coordinate system.

You can download the software from the link

http://www.padowan.dk/graph/

Install the graph 4.3 software on your computer, when you click on the graph 4.3 icon the following window will appears.



For entering data you need to select "insert a point series", it is the sixth icon (from the left) in the above window. Then you enter your data and then click ok to get the plot. After that you can fit your data using "insert a curve for the best fit point series". This is the eighth icon from the left in the above window. You can get the online support and tutorial for this by clicking help option in the window.

## Summary

In the present chapter, we have outlined the use of Scilab. For numerical methods, you will find it very useful. For those who are not keen on learning a computer language and start using a computer to solve numerical problems, Scilab is quite handy, although for a person who wants to pursue computational or theoretical chemistry seriously; fair exposure to a programming language is almost a must. We have taken some elementary applications using Scilab such as simple arithmetic operations, matrix operations, differential equations, plotting, writing functions and curve fitting. You may experiment with the large number of other functions which hare in-built in Scilab. You have also been introduced to the plotting software, Graph 4.3. As an additional practice, you may install Xmgrace and Avogadro software in your computer and see how to use them.

## Problems

1. Solve the following equations (Cramer's method)

   I. $-2x + 3y = 8$

   $3x - y = -5$

II. $2x - y + 3z = -3$

$-x - y + 3z = -6$

$x - 2y - z = -2$

2. Find the roots of the following equations

I. $x^6 - 5x^5 + 4x^4 - 3x^3 + 2x^2 - x + 10$

II. $x^8 - 4x^6 + 3x^2 + x - 15$

3. Plot at least ten Hermite polynomials using Scilab.