

## **Chapter 8: Useful Softwares**

### 8.1 Introduction

In the present chapter, you will be introduced to some very useful public domain softwares which help 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 Plot4.0 in Chapter 4 wherein we studies interpolation.

### 8.2 SCILAB Introduction

**Scilab is free downloadable from the link <http://www.scilab.org/> Copy the file to your computer and install it.**

Scilab regular commands:

Chdir or cd -to change directory to the local directory wherein you are storing your files.

E.g. my directory is in 'D' drive as Scilab, then the command is chdir('D:\Scilab')

ls – List all the files in the corresponding directory.

**Scilab constants:**

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

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

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

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

**%f** =  $\sim$ **%t**

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

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

Scilab cannot give the answers for 1/0 and 0/0 directly;

But by writing `ieee(1)` command, it will give 1/0 as inf and 0/0 as nan

## 8.3 Scilab Commands

**Note: All outputs are given in purple color**

// is comment in Scilab

//add n numbers

`sum (1:20)`

**ans= 210**

//caluculations with matrices

//Addition, Subtraction and Scalar Multiplication

### 8.3(a) Matrix Operations

`p = [4 5 7; -2 0.5 1]; // 2*3 matrix`

`q = 2; // Scalar`

```
r = [2 1 5; 2 -3 7]; // 2*3 matrix
```

```
s = r' //transpose
```

```
s =
```

```
    2.    2.
```

```
    1.   -3.
```

```
    5.    7.
```

```
p + q
```

```
ans =
```

```
    6.    7.    9.
```

```
    0.   2.5    3.
```

```
p-r
```

```
ans =
```

```
    2.    3.    5.
```

```
   -4.  -1.5   -1.
```

```
p*q
```

```
ans =
```

```
    8.   10.   14.
```

```
   -4.    1.    2.
```

```
//In order to multiply two matrices, they need to have matching rows
```

```
//and columns
```

```
//p*r it is wrong
```

```
p*s
```

```
ans =
```

```
   48.   42.
```

1.5 1.5

s\*p

ans =

4. 11. 16.

10. 3.5 4.

6. 28.5 42.

//Scilab provides element-by-element multiplication or

//division of two matrices using the operators “.\*” and “./”

p.\*r//is correct//multiplied element by element

//p^2 (it is wrong)

p.^2 //is correct//each element is squared.

//Eigenvalues, Eigenvectors and Diagonalization

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

//The eigenvalues can be obtained using the command “spec”.

EigenVals = spec (A)

**EigenVals =**

**- 5.9553861**

**0.3264781**

**4.628908**

//Diagonalization of matrix A in the form lambda =

//A (diag) = X\*\*(-1) A X is obtained as follows

[Lam, X] = bdiag (A)//Lam is the set of Eigen values

**X =**

```
- 1.8071512  1.1659151 - 0.0630461i  
 0.4062757  0.7586045 - 0.5671129i  
- 0.1359822  0.3988603  0.9023204i
```

**Lam =**

```
0.3264781  0.          0.  
0.          4.628908  0.  
0.          0.        - 5.9553861i
```

//X contains the matrix of Eigen vectors which diagonalizes A

X \* Lam \* inv(X)

**ans =**

```
1.          4.  3.  
- 4.441D-16  2.  5.  
1.          3. - 4.
```

## 8.3(b) Numerical Integrations

//integration of an expression by quadrature

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

**X =**

```
2.
```

// to solve a definite integral

Function  $y=f(x)$ ,  $y=(x*\sin(x))$ , endfunction //this defines the function y

$I=\text{intg}(0,2*\%pi,f)$  //it integrates the function from 0 to 2 pi

**I =**

**- 6.2831853**

### 8.3(c) Differential Equations

//to solve a ode(ordinary differential equation)

//  $dy/dt=y^2-y\sin(t)+\cos(t)$ ,  $y(0)=0$  is the differential eq.

//with boundary condition

function ydot=f(t,y), ydot= $y^2-y*\sin(t)+\cos(t)$ , endfunction

$y_0=0$ ;  $t_0=0$ ;  $t=0:0.1:\%pi$ ;

//initial condition is  $y_0=0$ , i.e.,  $y(0)=0$

//the range of t is from 0 to pi with interval of 0.1

$y=\text{ode}(y_0,t_0,t,f)$  //this gives the ans 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

y=ode(y0,t0,t,f)//this gives y in a single line .

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

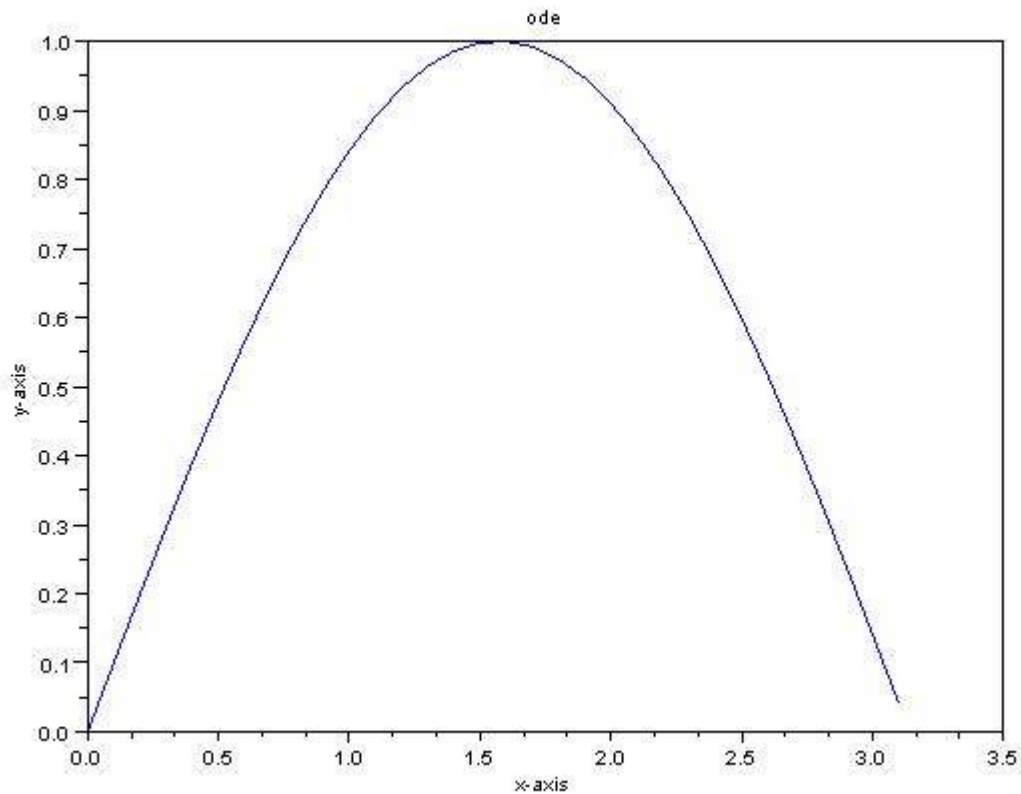
0.0415812

//the following plot function plots values of y vs t in a separate window.

### 8.3(d) Graphs

```
plot(t,y);xtitle('ode','x-axis','y-axis')
```





To save the plot to file:

**xs2jpg(0,'ode')**

0 is window number and ode is file name.

Jpg is the format of the picture. We can save the graph as required format i.e. pdf(xs2pdf).

//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

**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**

**And**

**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

Other Example for integration:

function approx = trapez(n,func)

sum\_trap=0;

for i = 1:n, sum\_trap = sum\_trap + func(i);end;

approx = sum\_trap

endfunction

```
x=
```

```
0.1
```

```
0.2
```

```
0.3
```

```
0.4
```

```
0.5
```

```
0.6
```

```
0.7
```

```
0.8
```

```
0.9
```

```
1.0
```

```
1.1
```

```
func=
```

```
2.3
```

```
2.7
```

```
3.4
```

```
4.0
```

```
5.0
```

```
6.5
```

```
8.0
```

```
10.0
```

```
13.0
```

```
18.0
```

```
25.0
```

```
n=11;
```

```
approx = trapez(n,func)
```

```
approx =
```

```
97.9
```

### 8.3(e) Reading from disk and Writing files to disk

**To save a data to file:**

**fprintfMat('x.txt', x)**

**That means x data saved as a x.txt**

**We can recall the x data as following**

**Y=fscanfMat('x.txt')**

**That means x.txt called as Y data.**