Brief tutorial on Scilab Prepared by: Jayendran Venkateswaran, IEOR, IIT Bombay

Scilab is free open source software for numerical computations

This tutorial is designed to introduce you to the use of Scilab for use in numerical computations, and visualization. Scilab features a family of specialized toolboxes that are application specific collections of script files (.sce or .sci files) that extend the SCILAB environment to solve particular classes of problems such as: basic computations, symbolic, curve fitting, financial derivatives, optimization, simulation, fuzzy logic, neural networks, partial differential equations, etc.

This tutorial is by no means complete with regards to all of SCILAB features. It is strongly recommended that you actively use various sources, including the SCILAB Help, to learn to do computations in Scilab.

1. Installing Scilab
Scilab is already available on lab-PCs. You can download binaries for your laptops from http://www.scilab.org/

2. Getting Started
You can start SCILAB from the start menu. You can also open a terminal and just type
`scilab &`
Wait till the command prompt --> appears on the Console Window.

3. Basic Computations
This section describes basic mathmatical operations in SCILAB.

3.1 Calculator: Calculator functions will work in SCILAB, as expected.

+ is addition, - is subtraction, / is division, * is multiplication, ^ is an exponent.

```
--> 3*(1+4)^2
--> sin(0.5)          // sin, cos, etc takes parameters in radians
--> exp(3)            // e3
--> log(3);           // Natural logarithm. If you use ; result will not be displayed. Type ans to view result.
```

3.2 Variables: Just like most other programming languages, you can assign variables from the SCILAB workspace. Everything in SCILAB is a matrix. If it's a scalar, it's actually a 1x1 matrix, and if it is a vector, it is an $n \times 1$ or $1 \times n$ matrix.

```
--> a = 3
--> b = [1 5 7 3 6]        // Creates a row vector
--> c = [1; 5; 1; 0; -2]   // Creates a column vector
--> b(3)                   // Accesses the 3rd element in array or vector.
--> a*b                    // Is result as expected?
--> b*a                    // Is result as expected?
--> b*c                    // Is result as expected?
--> c*b                    // Is result as expected?
--> a/b                    // Is result as expected? This actually returns a vector x such that b*x=a.
--> c./b                   // Is result as expected? This returns element -wise division
```

```
--> y = exp(b)          → eb
--> log(y)              → You should get b vector defined earlier!
--> sum(b)              → sum of all elements of b
--> prod(b)             → product of all elements of b
```

3.3 Vectors: Creating a vector of values

```
--> c = 1:5             → This creates a vector from 1 to 5 in steps of 1
--> d = 1:2:15          → This creates a vector from 1 to 15 in steps of 2
--> d = %pi/100:%pi/100:%pi    → Is resultant d vector as expected?
```

→ '%' symbol is used for predefined mathematical variables

```
--> w=linspace(1,4,9)   → Creates a linearly spaced vector with 9
                          equal spaced values between 1 and 4
```

4. Saving your work

You can save statements/commands in a .sce file. To open/edit a .sce file, you can just type

```
edit filename.sce
```

You will see a new window 'scinotes'. It is just a text editor. Then type all the commands you want to execute and save it. To run these commands, you can click 'Execute'-->'file with echo ...'

Exercise-1

Create a file ex1.sce with the following contents

```
d = %pi/100:%pi/100:%pi
d*d
w=linspace(1,4,9)
```

Save it as ex1.sce. Execute it by clicking on 'Execute-->'... file with echo'. You should see the output in the main Scilab window.

Exercise-2

Create an array of values for the function (sin x)/x, where x takes on 10 equally spaced values between $\pi/2$ and π. Save all your commands in ex2.sce.

5. Matrices

(i) To enter a matrix in Scilab we simply type the entries (separated by a space or comma) row by row with semicolons separating the rows.

```
--> A=[1 7 3;4 0 6;2,5,-1]
```

or we could simply hit enter after each row is typed

```
--> B=[1 0 -1
 -1 0 1
 0 -1 1]
```

(ii) Play with matrices

```
--> A'                  → Gives transpose of A
--> C=A*B               → Computes matrix product
--> C=A.*B              → Computes dot product i.e. element wise operation
--> D=B^2               → Computes square of matrix
--> E=[B, [2 5 7]']     → creates a combined matrix
--> A(1,2)              → selects the element in 1st row and 2nd column
--> A(:,2)              → selects the 2nd column
--> A(1,:)              → selects the 1st row
```

```
--> A
--> E(2,:)=[]          removes the second row of E
--> E(:,3)=[]          removes the third column of E
--> E
--> size(E)            gives row and columns of E

--> a=3
--> a*A                Is result as expected? Scilab is CaSeSeNsItIvE

--> det(A)             gives determinant of A
--> rank(A)            gives rank of A
--> evals=spec(A)      gives eigenvalues of A
--> [P,LAMBDA] = spec(A)  gives eigenvectors of A, where P is the
                       matrix of eigenvectors and LAMDA is the diagonal
                       matrix of eigenvalues such that A=P(LAMDA)P–1 .
--> inv(A)             gives inverse of A, if it exists
--> A*inv(A)           Should the result be identity matrix?
--> inv(A)*A           Should the result be identity matrix?
--> poly(A,"x")        gives characteristic equation of polynomial of A
--> coeff(P)           gives coefficients of polynomial of A

--> g = [ 1 2; 3 4 ]
--> h = [ 1 2; 0 1 ]
--> g/h                Does it give same result as g*inv(h) ?
--> g\h                Does it give same result as inv(g)*h ?
--> ones(3,4)          To create matrix of 1s of size 3x4
--> zeros(4,3)         To create matrix of 0s of size 4x3
--> rand(5,4)          To create  matrix of random numbers of size 5x4
--> eye(4,4)           To create Identify matrix of size 4x4
--> eye(4,5)           What does it do?

--> d=1:5;
--> main=diag(d)       create matrix with values on the main diagonal
--> super=diag(d,1)    create matrix with values on the super diagonal
--> sub=diag(d,-1)     create matrix with values on the subdiagonal
```

Exercise-3

Let .

Compute the following (if they exist) and report the values in your lab-notebook.

Determinant, Rank, Inverse and Eigenvalues of the product $D=ABC$. Save your work in ex3.sce.

6. Using plot

```
--> x = -%pi:0.25:%pi;
--> y = sin(x);
--> plot(x,y)
```

```
--> plot(x,y),  set(gca(),"grid",[1 1])

--> plot(x,y), title('MY PLOT')
```
 Note: The title() to be enclosed in single quotes.
```
--> plot(x,y,'y-',x,y,'go')
--> help plot
```
 Read the help to understand how to use plot.

6.1 To have different plots in different windows.

```
--> plot(x,sin(x),'b')
--> figure
```
 The figure function creates figure windows
```
--> plot(x,cos(x),'r')
```

6.2 To superimpose multiple plots on same window

```
--> figure(10)
--> plot(x,sin(x),'b',x,cos(x),'r-')
```

Exercise-4

a. Create a plot of function **e–*x*cos(6π*x*);**, where *x* takes on 100 data points over interval (0,1).
b. On the same figure, plot functions **e–*x*** and **–e–*x*.** Give different colors/patterns for each curve.
c. Save this plot as a pdf file. (In the plot window, click on File, then Export To …, then select pdf as the option.
d. Explain in your report why the first curve lies between the other two curves and find out the number of points where any two curves meet.

6.3 Creating graphs of two variables

```
--> [x,y] = meshgrid(-3:.2:3, -3:.2:3);
--> z = (y-0.5).* x .* exp(-x.^2 - y.^2);
--> surf(x,y,z)
```
 generates surface plot
```
--> contour(x(1,:),y(:,1),z,20)
```
 generates 20 contour lines

7. Writing functions in .sci files

Like other programming languages, Scilab also has provisions to write your own functions. Usually, functions are defined in files with an editor (like 'scinotes') and loaded into Scilab using the **exec** function. Function files should have a .sci extension in the filename so as to distinguish them from .sce files.

Example 1: This simple function shows basic parts of a program file.
Any line that begins with // is used for commenting and will not be executed

Enter the below program in the .sci file. Save the file as **fact.sci** *in the default directory.*

```
function [f] = fact(n)                    Function definition line
// Compute a factorial value.             H1 line
// FACT(N) returns the factorial of N.    Help text
// Put simply, FACT(N) is PROD(1:N).      Comment
f = prod(1:n);                            Function body
```

You can execute it from 'scinotes' itself or go back to the Console window of SCILAB and type

```
--> exec('fact.sci')

--> fact(4)          ⇨ result of 4! is to be displayed.
```

Additional Reading: This table briefly describes parts of a program file.

File Element	Description
Function definition line (functions only)	Defines the function name, and the number and order of input and output arguments
H1 line	A one line summary description of the program, displayed when you request help on an entire folder, or when you use lookfor
Help text	A more detailed description of the program, displayed together with the H1 line when you request help on a specific function
Function or script body	Program code that performs the actual computations and assigns values to any output arguments
Comments	Text in the body of the program that explains the internal workings of the program

Function Definition Line

Function Arguments. If the function has multiple output values, enclose the output argument list in square brackets. Input arguments, if present, are enclosed in parentheses following the function name. Use commas to separate multiple input or output arguments.

Here is the declaration for a function named sphere that has three inputs and three outputs:
```
function [x, y, z] = sphere(theta, phi, rho)
```

If there is no output, leave the output blank `function printresults(x)`
or use empty square brackets: `function [] = printresults(x)`

The variables that you pass to the function do not need to have the same name as the variables in the function definition line.

Example 2: Let us solve a second order algebraic equation $ax2 + bx + c = 0$. The solution is given in analytical form as
We want to write an *sci-file* with the name *secroot.sci,* which produces the analytical solution

```
function [r1,r2]=secroot(a,b,c);
// Find Determinant
Det= b^2 - 4 * a * c;
if (Det < 0),
     r1=- (-b + %i* sqrt(-Det))/2/a;
```

```
     r2 = (-b -%i* sqrt(-Det))/2/a;
     disp('The two roots are complex conjugates');
elseif(Det == 0),
     r1 = -b/2/a;
     r2 = -b/2/a;
     disp('There are two repeated roots');
else(Det > 0)
     r1 = (-b + sqrt(Det))/2/a;
     r2 = (-b - sqrt(Det))/2/a;
     disp('The two roots are real');
endfunction
```

Exercise-5

A. Use `secroot` program you have written to find the roots of (i) $x2 - 1 = 0$ (ii) $x2 - 2x = 1$ (iii) $3x2 + 4x + 5 = 0$. Explain the working of your program in your report.

B. Write a function *MatrixSwap* which takes as input a n x n matrix, and two integers a and b (where $1 \leq a,b \leq n$) , then swaps the rows a and b of the matrix and output the modified matrix. Explain the working of your program in your report.

Exercise-6

Create a function *ArrowMatrix* which takes as input a number n and returns an $(n+1)$ x $(n+1)$ matrix whose element (i, i) is i, element(n+1,i) is 2i and element (i,n+1) is 2i for i=1,...,n. Element (n+1,n+1) must be i. All other elements are zero. You should not use any loops or if conditions. Explain the steps in your report.

7. Using Loops

Much like other programming languages, SCILAB also provides 'for' and 'while' loops. Here are are some simple 'for' loops:

To display squares of first 5 natural numbers

```
--> a = zeros(5);
--> for i = 1:5
-->    a(i) = i*i;
--> end
```

To display squares of first 10 even numbers in reverse order

```
--> a = zeros(10);
--> for i = 20:-2:2
-->    i*i
--> end
```

Additional Material: Creating multiple subplots

subplot(m,n,i) breaks the figure window into an m-by-n matrix of small subplots and selects the ithe subplot for the current plot. The plots are numbered along the top row of the figure window, then the second row, and so forth.

```
     --> t = 0:%pi/20:2*%pi;
     --> [x,y] = meshgrid(t);
     --> subplot(2,2,1)
     --> plot(sin(t),cos(t))
```

```
--> a = gca();
--> a.isoview = 'on';
```
 equivalent to MATLAB code axis equal

```
--> subplot(2,2,2)
--> z = sin(x)+cos(y);
--> plot(t,z)
--> isoview(0,2*%pi,-2,2)
--> subplot(2,2,3)
--> z = sin(x).*cos(y);
--> plot(t,z)
--> isoview(0,2*%pi,-1,1)
--> subplot(2,2,4)
--> z = (sin(x).^2)-(cos(y).^2);
--> plot(t,z)
--> isoview(0,2*%pi,-1,1)
```