# SCILAB- An Introduction

**Dr. Balasaheb M. Patre,**
Professor and Head,
Department of Instrumentation Engineering,
SGGS Institute of Engineering and Technology,
Vishnupuri, Nanded-431606.
E-mail: bmpatre@yahoo.com


# What is SCILAB?

(1) SCILAB is a freely distributed and open source scientific software package

(2) A powerful open computing environment for Engineering and Scientific applications

(3) Developed since 1990 by researchers from INRIA (Institut Nationale de Recherche en Informatique et en Automatique) and ENPC (National School of Bridges and Roads).

(4) Now maintained and developed by Scilab consortium since 2003.

(5) Integrated into Digiteo foundation in July 2008

(6) The current version is 5.2.1 (February 2010)

# What is SCILAB? ...contd

7) Since 1994 it is distributed freely along with source code through the Internet. (www.scilab.org)

8) Scilab users can develop their own module so that they can solve their particular problems.

7) The Scilab language allows to dynamically compile and link other languages such as Fortran and C: this way, external libraries can be used as if they were a part of Scilab built-in features.

8) Scilab also interfaces LabVIEW, a platform and development environment for a visual programming language from National Instruments.

# Scilab's Main Features:

1. A high-level programming language
2. Scilab is an interpreted language
3. Integarated object-oriented 2-D and 3-D graphics with animation
4. A dedicated Editor
5. An XML-based help system
6. Interface with symbolic computing packages (Maple and MuPAD 3.0)
7. An interface with Tcl/Tk
8. Scilab works with most Unix systems including GNU/Linux and on Windows (9X/NT/2000/XP/Vista/7), and Mac operating system

# Scilab coded Toolboxes

1. Linear algebra and Sparse matrices
2. Polynomials and Rational functions
3. 2-D and 3-D graphics with animation
4. Interpolation and Approximations
5. Linear, Quadratic and Nonlinear Optimization
6. ODE solver and DAE solver
7. Classical and Robust Control, LMI Optimization
8. Differentiable and Non-differential Optimization
9. Signal Processing
10. Statistic
11. Scicos: A hybrid dynamic system modeler and simulator
12. Parallel Scialab using PVM
13. Metanet: Graphs and Networks


## Typical uses

- Educational Institutes, Research centers and companies
- Math and computation
- Algorithm development
- Modeling, simulation, and visualization
- Scientific and engineering graphics, exported to various formats so that can be included into documents.
- Application development, including GUI building

## Basic data element (Matrix)

Array : not require dimensioning

Allow to solve problem with matrix and vector formulations

**Desktop tool and development environment**

Set of tools and facilities

Graphical UI : Scilab Console, Sciab editor, Scilab help bowser, MATLAB to Scilab Translator

**Mathematics Function Library**

Collection of computational algorithm : sum, sine, matrix functions

**Language**

High-level matrix/array language with flow, functions, structure

**Graphics**

Extensive facilities for displaying vectors and matrices as graphs

High-level functions for 2-D and 3-D data visualization

**External Interfaces**

Allows to write C and Fortran programs that interact with SCILAB

# Getting Started with Scilab

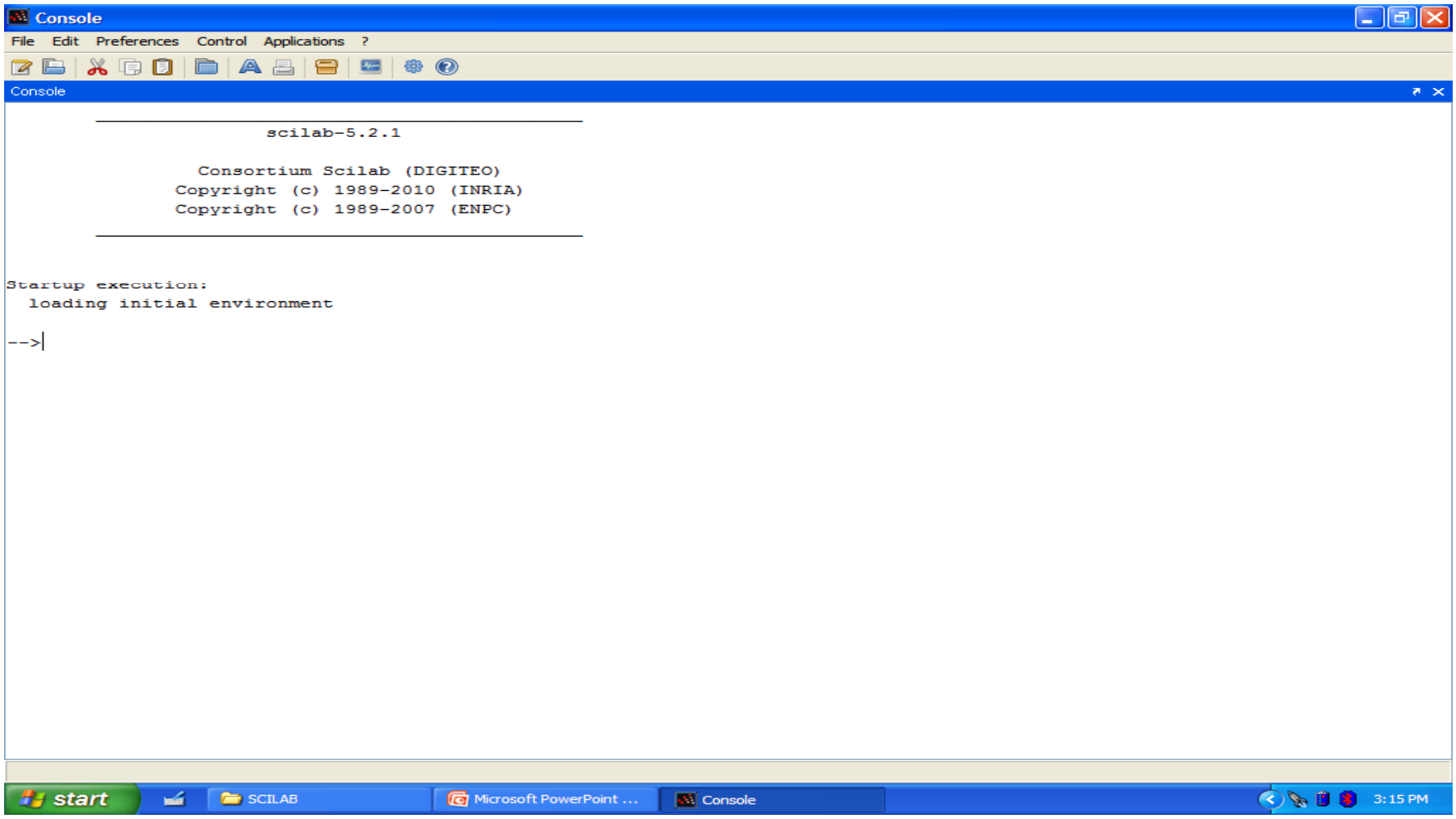
**Stating the Scilab program**

- Start the Scilab program by double-clicking Scilab-5.2.1 icon on the desktop
- Start button on the desktop >Programs>Scilab-5.2.1>Scilab-5.2.1
  Automatically loading
  Tools for managing files, variables and applications

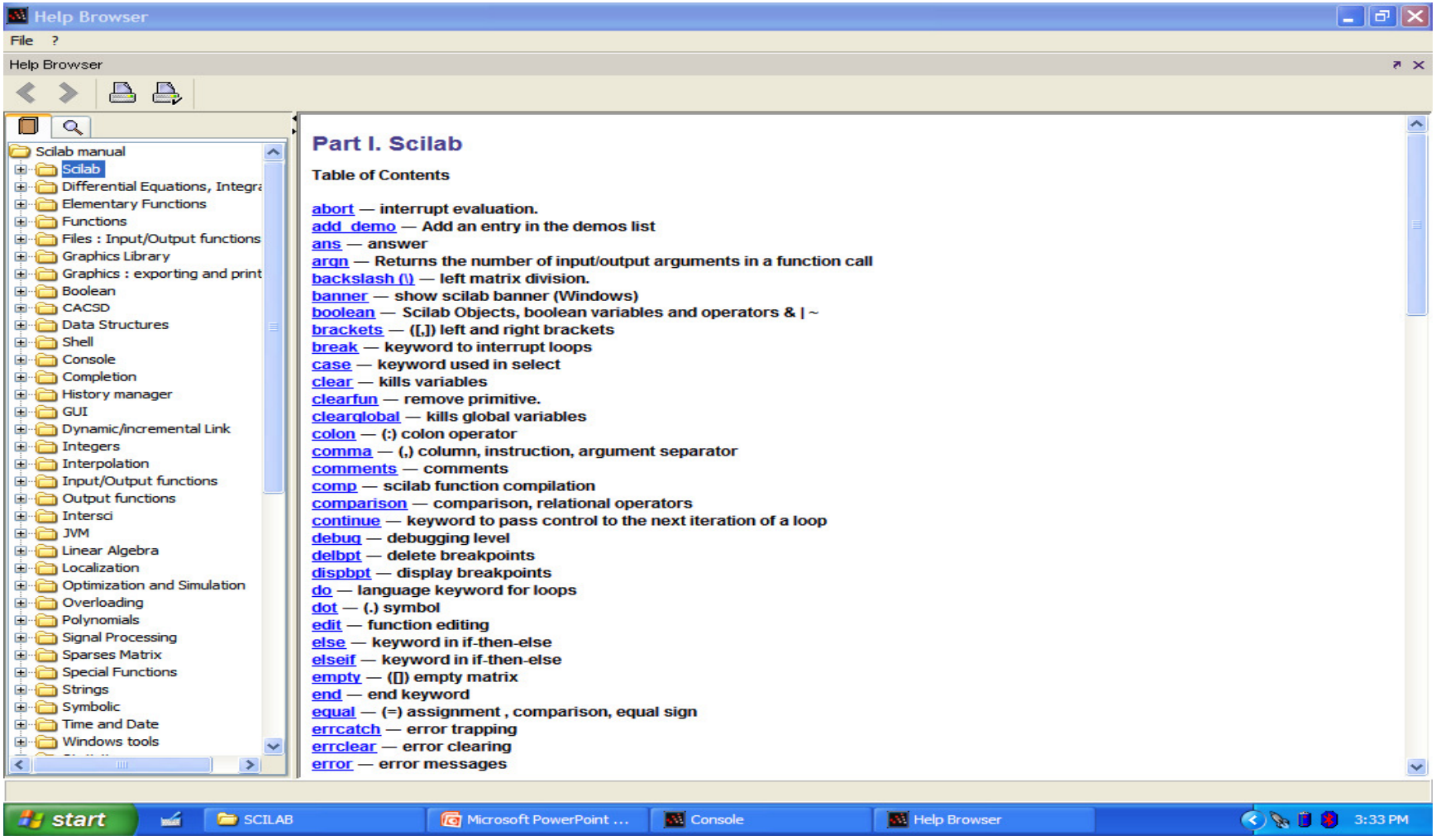
**Quitting the Scilab program**

- To end SCILAB, **File > quit** in the scilab console
- Type '**quit'** in the Scilab Console
- The user enters commands at the prompt ---->

# Scilab Default Console



# Scilab Help Browser



# Help Features

**To open SCILAB help, click help icon (?) in the toolbar or type help at the command prompt ----->**

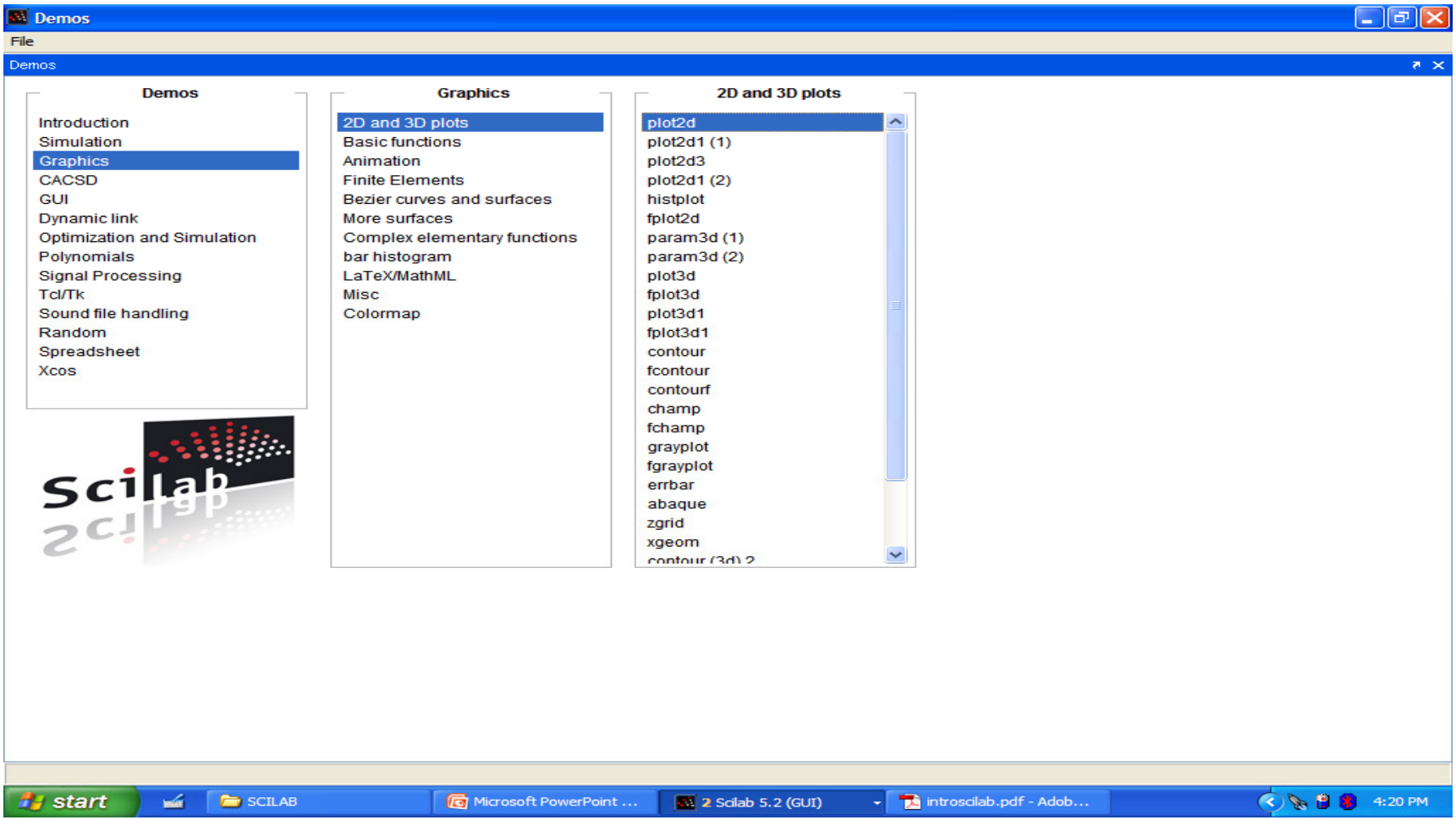
- **Help Browser**
- **help command (help inv, help optim)**

  **(This is useful when the name of the function is already Known)**
- **To obtain a list of Scilab functions corresponding to a keyword, the command apropos followed by the keyword should be used.**

-->apropos  eigenvalues <Enter>

- **Help can also be from Scilab demonstrations**
- **This is available from the console, in the menu ? > Scilab Demonstrations.**

# Scilab Demos Window



# Arithmetic Operations:

- + addition
- - subtraction
- * multiplication
- / right division i.e. $X / Y = XY^{-1}$
- \ left division i.e. $X \backslash Y = X^{-1}Y$
- ^ power i.e. $X^{Y}$
- ** power (same as ^)
- ' transpose conjugate

# Scilab as a Calculator

```
-->6+5
 ans  =

    11.

-->6+5;

-->7+8/2
 ans  =

    11.

-->(7+8)/2
 ans  =

    7.5
```

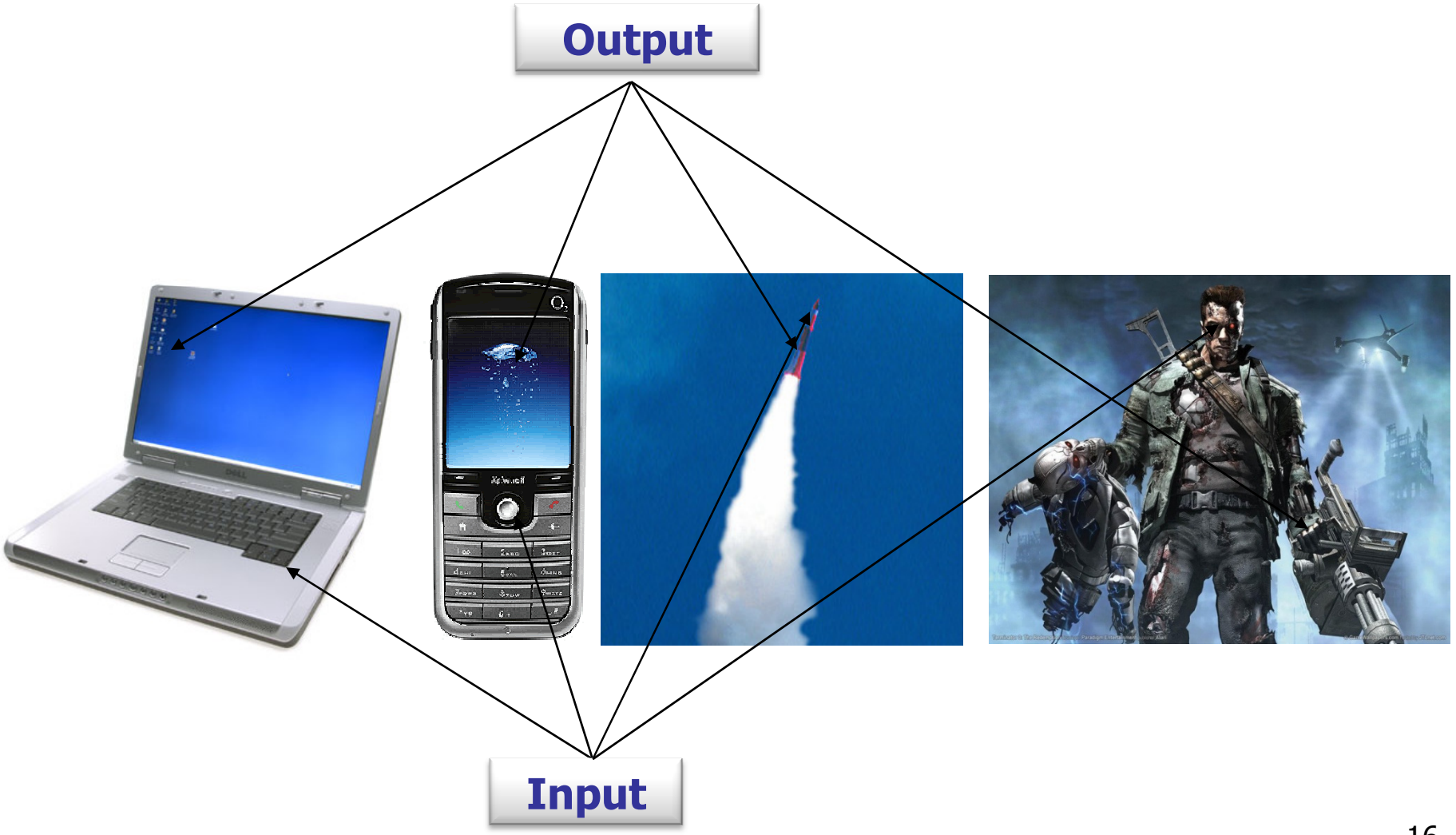
```
-->4+5/3+2
 ans  =

    7.6666667

-->5^3/2
 ans  =

    62.5

-->27^(1/3)+32^0.2
 ans  =

    5.

-->27^1/3+32^0.2
 ans  =

    11.
```

# Scilab as a Calculator

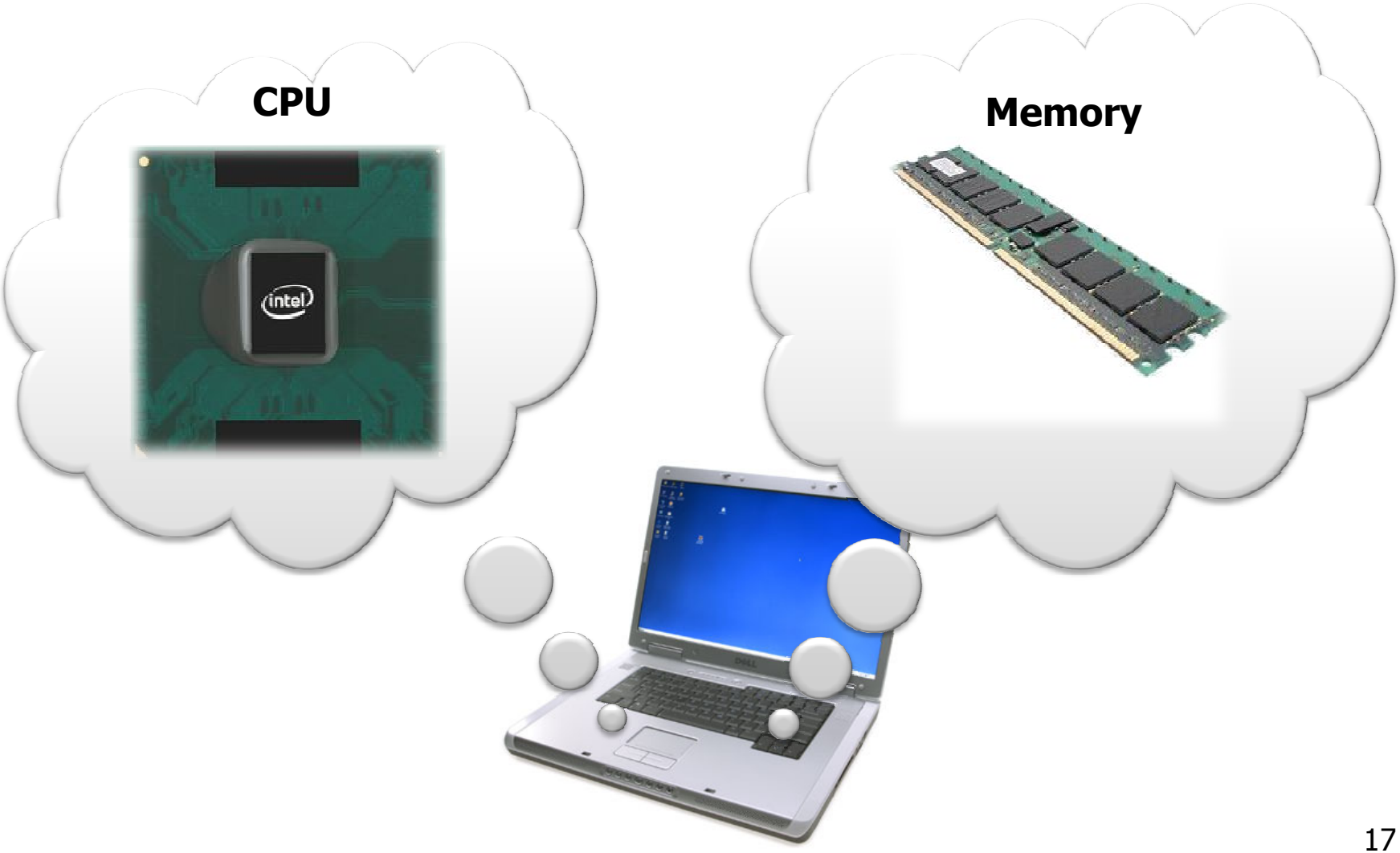
```
-->0.7854-(0.7854)^3/(1*2*3)+0.785^5/(1*2*3*4*5)..
-->-(0.785)^7/(1*2*3*4*5*6*7)
 ans  =
    0.7071016

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

- In Scilab, any line which ends with two dots is considered to be the start of a new continuation line.
- Any line which begins with two slashes "//" is considered by Scilab as a comment and is ignored.
- More than one command can be entered on the same line by separating the commands by semicolon (;) or a comma (,)

# Background - computers



# Background - hardware



# Background - software

**High level languages**

Examples: C, C++, C#, Java, Pascal, Perl, Lisp, ***Scilab***

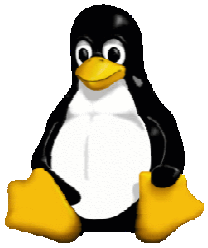
**Low level language**

Example: Assembly

**Machine language**

Example: 011101010111101…

Another important player:
The operating system




# Basic Elements of Scilab:

- In Scilab, everything is a matrix
- All real, complex, Boolean, integer, string, and polynomial variables are matrices.
- Scilab is an interpreted language, which implies that there is no need to declare a variable before using it. Variables are created at the moment where they are first set.
- In Scilab "=" sign is called assignment operator.

```
-->x=10              10 is assigned to variable x
 x  =
    10.
-->x=3*x-12          A new value is assigned to x. The new values is
 x  =                three time of previous value of x minus 12.
    18.
```

- Varaiable names may be as long as user wants but only first 24 characters are taken into account.
- Scilab is case sensitive. A is not equal to a.

# Predefined Variables:

Certain variables are predefined and write-protected

| | | | |
|---|---|---|---|
| %i | $i = \sqrt{-1}$ | | immaginary unit |
| %pi | $\pi = 3.1415927\ldots$ | | pi grek |
| %e | $e = 2.718281\ldots$ | | number of Nepero |
| %eps | $\varepsilon = 2.2 \times 10^{-16}$ | | precision (machine dependent) |
| %inf | | | infinity |
| %nan | | | NotANumber |
| %s | | s | polynomial variable |
| %z | | z | polynomial variable |
| %t | | true | boolean variable |
| %f | | false | boolean variable |

# Some useful Scilab Commands

General commands:

- clock — Provide clock time and date as a vector [year month day hour minute seconds]

```
-->clock
 ans  =
    2010.    4.    20.    23.    38.    59.
```

- date — Current date a string

```
-->date
 ans  =
  20-Apr-2010
```

- ver — Version information for Scilab

```
-->ver
 ans  =
 !Scilab Version:            5.2.0.1266391513    !
```


## Some useful Scilab Commands ……contd.

- Workspace Commands:

| | |
|---|---|
| who | Lists the variables currently in the scilab workspace |
| whos | Same a who but provides more information on size, type |
| whos -type constants | List the variables that can store real or complex constant |
| Whos –name a | List all variables with name starting with the letter 'a' |
| what | Lists the scilab primitives |
| clear | Kills the variables which are not protected. |
| clear xyz | Kills the variables specified in the command |
| clc | Clears screen |
| clf | Clears figure window |
| diary | List of current session commands |

## Some useful Scilab commands ….contd

- Directory commands:

pwd Provides scilab current working directory

```
-->pwd
 ans  =
  C:\Program Files\scilab-5.2.1
```

copyfile Copies a file

mkdir Makes a a new directory/folder in the current directory

- Termination Commands:

quit Quits Scilab

exit Same as quit command

# Creating Arrays (Vectors and Matrices)

```
-->a=[1 2 3 4 5 6 7 8 9 10]                    Create a row vector
 a  =
    1.    2.    3.    4.    5.    6.    7.    8.    9.    10.
 -->a=[1,2,3,4,5,6,7,8,9,10]          Another way of creating a row vector
 a  =
    1.    2.    3.    4.    5.    6.    7.    8.    9.    10.
 -->a=[1;2;3;4;5;6;7;8;9;10]                    Create a column vector
a  =
   1.
   2.
   3.
   4.
   5.
   6.
   7.
   8.
   9.
   10.
```

# Vectors and matrices ……contd.

```
Variable_name=m:q:n   (m=first term, q=spacing, n=last term)
-->a=1:10             Creating a row vector with colon (:) operator
 a  =                 Default incerment is one
    1.   2.   3.   4.   5.   6.   7.   8.   9.   10.
-->a=1:1:10           Specified increment is one
 a  =
    1.   2.   3.   4.   5.   6.   7.   8.   9.   10.
-->a=1:2:11           Specified increment is two
 a  =
    1.   3.   5.   7.   9.   11.
-->a=100:-10:0        Specified increment is -10.
 a  =
    100.   90.   80.   70.   60.   50.   40.   30.   20.   10.   0.
```

# Vectors and matrices ……contd.

```
-->a=[2+3*%i, 4+1*%i, 3, 5, 6]    Vector with complex numbers
 a  =
    2. + 3.i    4. + i      3.    5.    6.
 -->b=[1+6*%i, 4+6*%i 3, 4, 6]
 b  =
    1. + 6.i    4. + 6.i    3.    4.    6.
 -->c=a+b                                  Vector addition
 c  =
    3. + 9.i    8. + 7.i    6.    9.    12.
-->a-b                                     Vector subtraction
 ans  =
    1. - 3.i  - 5.i    0    1.    0
 -->a*b
   !--error 10
Inconsistent multiplication.
```

# Vectors and matrices ……contd.

```
-->a=linspace(0,10,5)
```
Generates a vector of 5 elements, 0 is the first element and 10 is the last element

```
a  =
   0.    2.5    5.    7.5    10.
```

```
-->a=logspace(0,4,3)
```
Generates a logarithmically spaced vector of length 3 between $10^0$ to $10^4$

```
a  =
   1.    100.    10000.

-->a=[1 10 25 50 15]
a  =
   1.    10.    25.    50.    15.
-->a(3)
```
Addressing a vector element

```
ans  =
   25.
```

# Vectors and matrices ……contd.

```
-->a=[1 10 25 50 15]
 a  =
    1.    10.    25.    50.    15.
-->b=sum(a)                          Sum of all elements
 b  =
   101.
 -->c=mean(a)                        Average of the elements
 c  =
    20.2
 -->d=length(a)                      Number of elements in the vector
 d  =
    5.
 -->e=max(a)                         Maximum value in the vector
 e  =
    50.
```

# Vectors and matrices ……contd.

```
-->f=min(a)                          Minimum value in the vector
 f  =
    1.
-->g=prod(a)                         Product of elements in the vector
 g  =
    187500.
-->h=sign(a)                 Returns 1 if the sign of an element is the
                        vector is +ve, 0 if element is 0, -1 if the element is –ve.
 h  =
    1.    1.    1.    1.    1.
-->i=find(a)                 Returns the indices corresponding to the non-zero
                             entry of the array a
 i  =
    1.    2.    3.    4.    5.
```

# Vectors and matrices ……contd.

```
-->p=[1.4 10.7 -1.1 20.9]
 p  =
    1.4    10.7  - 1.1    20.9
 -->a=fix(p)
 a  =
    1.    10.  - 1.    20.
-->b=floor(p)
 b  =
    1.    10.  - 2.    20.
 -->c=ceil(p)
 c  =
    2.    11.  - 1.    21.
 -->d=round(p)
 d  =
    1.    11.  - 1.    21.
-->e=gsort(p)
 e  =
    20.9    10.7    1.4  - 1.1
```

Rounds the elements of the vector p to the nearest integer towards zero

Rounds the elements of the vector p to the nearest integer towards $-\infty$

Rounds the elements of the vector p to the nearest integer towards $+\infty$

Rounds the elements of the vector p to the nearest integer

Sorts the eleemnts of p in descending order

# Vectors and matrices ……contd.

```
-->A=[16 3 2 13;5 10 11 8;9 6 7 12;4 15 14 1]
 A  =
    16.    3.     2.     13.
    5.     10.    11.    8.
    9.     6.     7.     12.
    4.     15.    14.    1.
-->B=sum(A)
 B  =
    136.
-->C=sum(A,'c')
 C  =
   34.
   34.
   34.
   34.
-->D=sum(A,'r')
 D  =
    34.    34.    34.    34.
```

Entering a matrix
use space or , for row elements
use ; to terminate a row

Gives the sum of all the elements

Sum of the elements of column

Sum of the row elements


# Matrix Addressing:

```
-->A=[3 11 6 5;4 7 10 2;13 9 0 8]
 A  =
    3.    11.   6.    5.
    4.    7.    10.   2.
    13.   9.    0.    8.
 -->A(2,3)
 ans  =
     10.
 -->A(:,2)
 ans  =
     11.
     7.
     9.
```

# Matrix Addressing

```
-->A(2,:)
 ans  =

    4.    7.    10.    2.

-->A(9)
 ans  =

    0.

-->A(1:2,1:2)
 ans  =
     3.    11.
     4.    7.
```

# Vectors and matrices ……contd.

```
-->B=A(3:-1:1,1:4)
 B  =
    13.    9.     0.     8.
    4.     7.     10.    2.
    3.     11.    6.     5.
-->B=A(3:-1:1,1:4)
 B  =
    13.    9.     0.     8.
    4.     7.     10.    2.
    3.     11.    6.     5.
 -->A(1:3,4)=[]
 A  =
     3.     11.    6.
     4.     7.     10.
    13.     9.     0.
```


# Vectors and matrices ……contd.

```
-->eye(2,2)
 ans  =
    1.    0.
    0.    1.
 -->ones(2,3)
 ans  =
    1.    1.    1.
    1.    1.    1.
 -->zeros(3,3)
 ans  =
    0.    0.    0.
    0.    0.    0.
    0.    0.    0.
 -->A=[1 2;3 4]; B=[2 3; 5 6];
 -->C=[A,B]
 C  =
    1.    2.    2.    3.
    3.    4.    5.    6.
```

# Vectors and matrices ……contd.

```
-->A=rand(2,3)
 A  =
    0.8497452    0.8782165    0.5608486
    0.6857310    0.0683740    0.6623569
 -->A=[1 2 3; 4 5 6;7 8 9];
 -->B=diag(A)
 B  =
   1.
   5.
   9.
 -->C=diag(A,1)
 C  =
    2.
    6.
 -->D=diag(A,-1)
 D  =
    4.
    8.
```

# Vectors and matrices ……contd.

```
-->A=[1 2;0 4];

-->rank(A)
 ans  =
    2.

-->norm(A)
 ans  =
    4.495358
-->C=A'
C  =
    1.    0.
    2.    4.
```

```
-->det(A)
            ans  =
                 4.

-->trace(A)
            ans  =
                 5.
 -->B=inv(A)
            B  =
                 1.  - 0.5
                 0.    0.25
```

# Vectors and matrices ……contd.

```
-->p=poly(A,'x')
 p  =
                2
   4 - 5x + x
 -->q=spec(A)
 q  =
    1.
    4.
ans  =
    0.    0.    0.
    0.    0.    0.
    0.    0.    0.
 -->A=[1 2;3 4]; B=[2 3; 5 6];
 -->C=[A,B]
 C  =
    1.    2.    2.    3.
    3.    4.    5.    6.
```


# Vectors and matrices ……contd.

## Matrix operators and elementwise operators

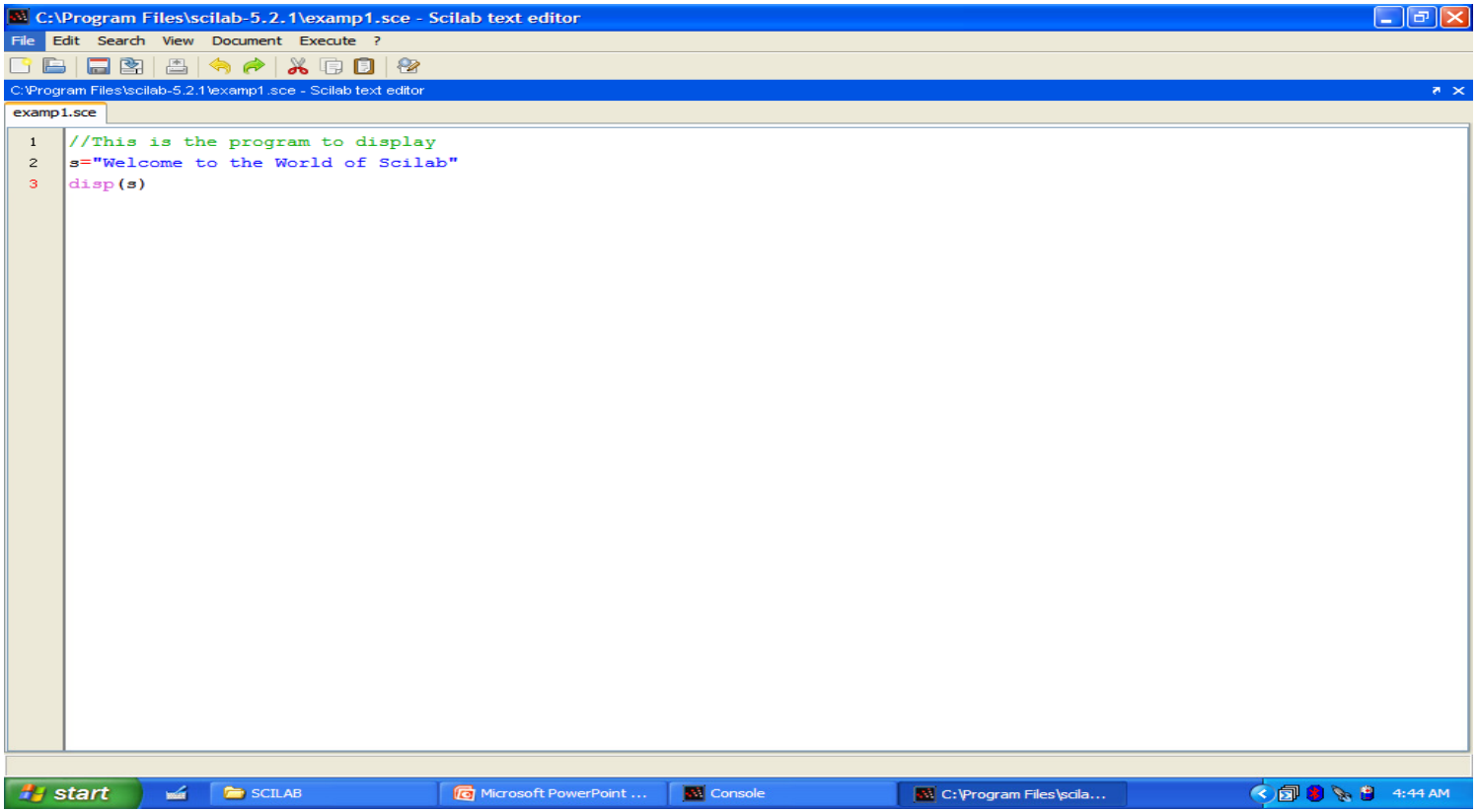
| | |
|---|---|
| + addition | .+ elementwise addition |
| - substraction | .- elementwise substraction |
| * multiplication | .* elementwise multiplication |
| / right division | ./ elementwise right division |
| \ left division | .\ elementwise left division |
| ^ or ** power | .^ elementwise power |
| ' transpose and conjugate | .' transpose (but not conjugate) |

# Scilab Editor

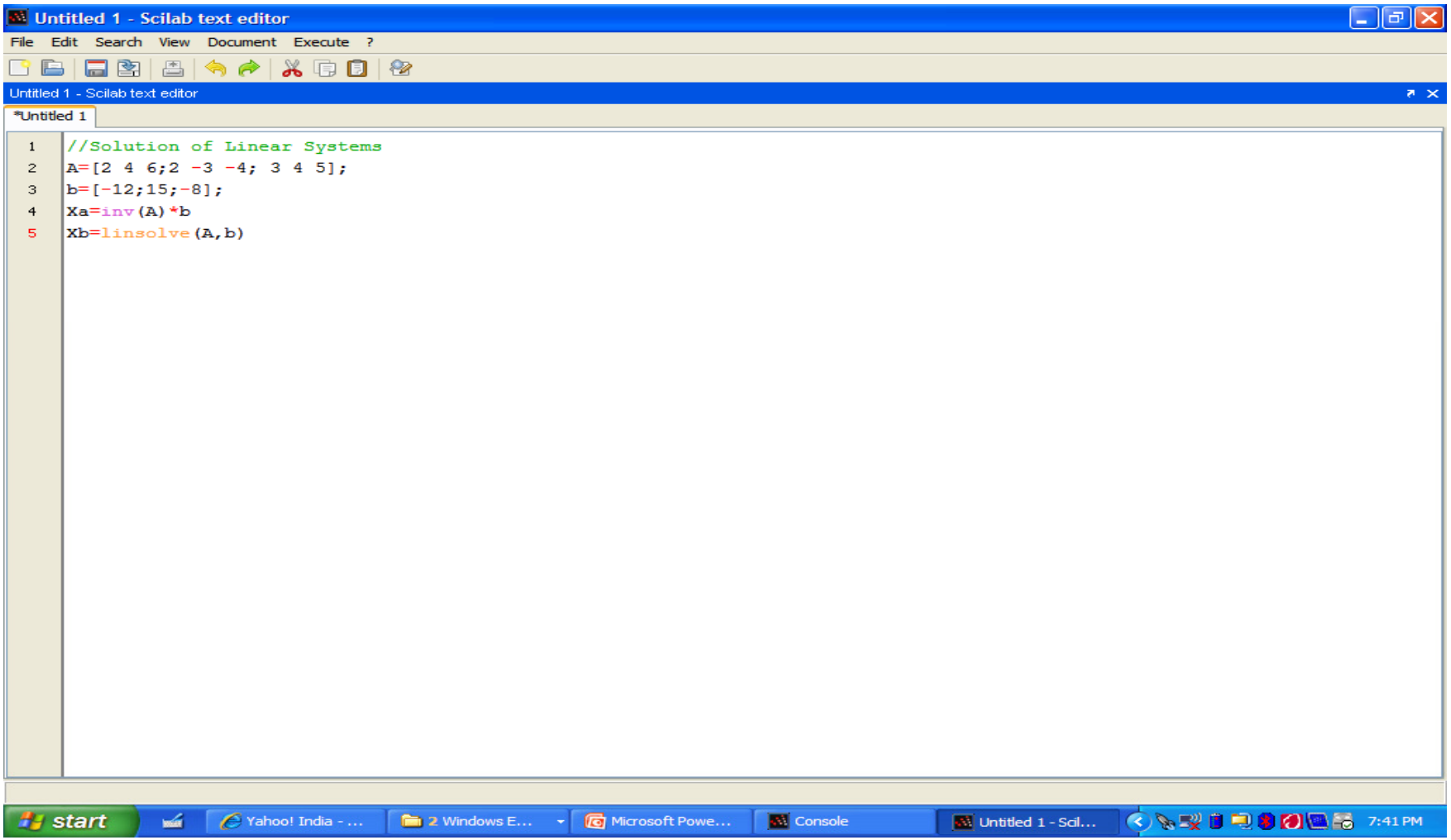
- When several commands are to be executed, it may be more convenient to write these statements into a file with Scilab editor. To execute the commands located in such a file, the exec function can be used, followed by the name of the script. This file generally has the extension .sce or .sci, depending on its content:
- Files having the .sci extension are containing Scilab functions and executing them loads the functions into Scilab environment (but does not execute them),
- Files having the .sce extension are containing both Scilab functions and executable statements.
- Executing a .sce file has generally an effect such as computing several variables and displaying the results in the console, creating 2D plots, reading or writing into a file, etc...

# Our first script (Sce-file)

The editor can be accessed from the menu of the console, under the Applications > Editor menu, or from the console as:
--> editor ()



# Another Script File



# Scilab Functions

- It is possible to define new functions in the scilab.
- To dene a new function, we use the function and endfunction Scilab keywords.

```
function y = myfunction ( x )
    y = 2 * x
endfunction
```

```
-->y=myfunction(3)
 y  =
    6.
 -->y=myfunction(8)
 y  =
    16.
```

# Scilab Functions ...contd

- Functions can have an arbitrary number of input and output arguments so that the complete syntax for a function which has a fixed number of arguments is the following:

  [o1 , ... , on] = myfunction ( i1 , ... , in )

- The input and output arguments are separated by commas ",". Notice that the input arguments are surrounded by opening and closing braces, while the output arguments are surrounded by opening and closing square braces .

# Computer precision limitations

- How much is:

```
-->0.42 + 0.08 - 0.5
 ans  =

    0.
```

- 

```
-->0.42 - 0.5 + 0.08
 ans  =

  - 1.388D-17
```

# Polynomials

- A polynomial can be created in two ways. One way is to define the polynomial in terms of its roots and the other way is to define it in terms of its coefficients.

```
-->p1 = poly([-1 -2], 'x')
 p1  =
                2
  2 + 3x + x
-->p1 = poly([-1 -2], 'x', 'r')
 p1  =
                 2
   2 + 3x + x
-->p2 = poly([2 3 1], 'x', 'c')
 p2  =
                2
   2 + 3x + x
```

# Polynomials ...contd.

```
-->roots(p1)
 ans  =
  - 1.
 - 2.
-->p3=p1+p2
p3  =
            2
   4 + 6x + 2x
-->p4=p1*p2
p4  =
                    2    3   4
   4 + 12x + 13x + 6x + x
-->p1==p2
ans  =
          T
```

# Polynomials ....contd

```
-->coeff(p1)
 ans  =
    2.    3.    1.
 -->derivat(p1)
 ans  =
    3 + 2x
 -->c=companion(p1)
 c  =
- 3.  - 2.
   1.    0.
 -->spec(c)
 ans  =
            - 2.
            - 1.
```

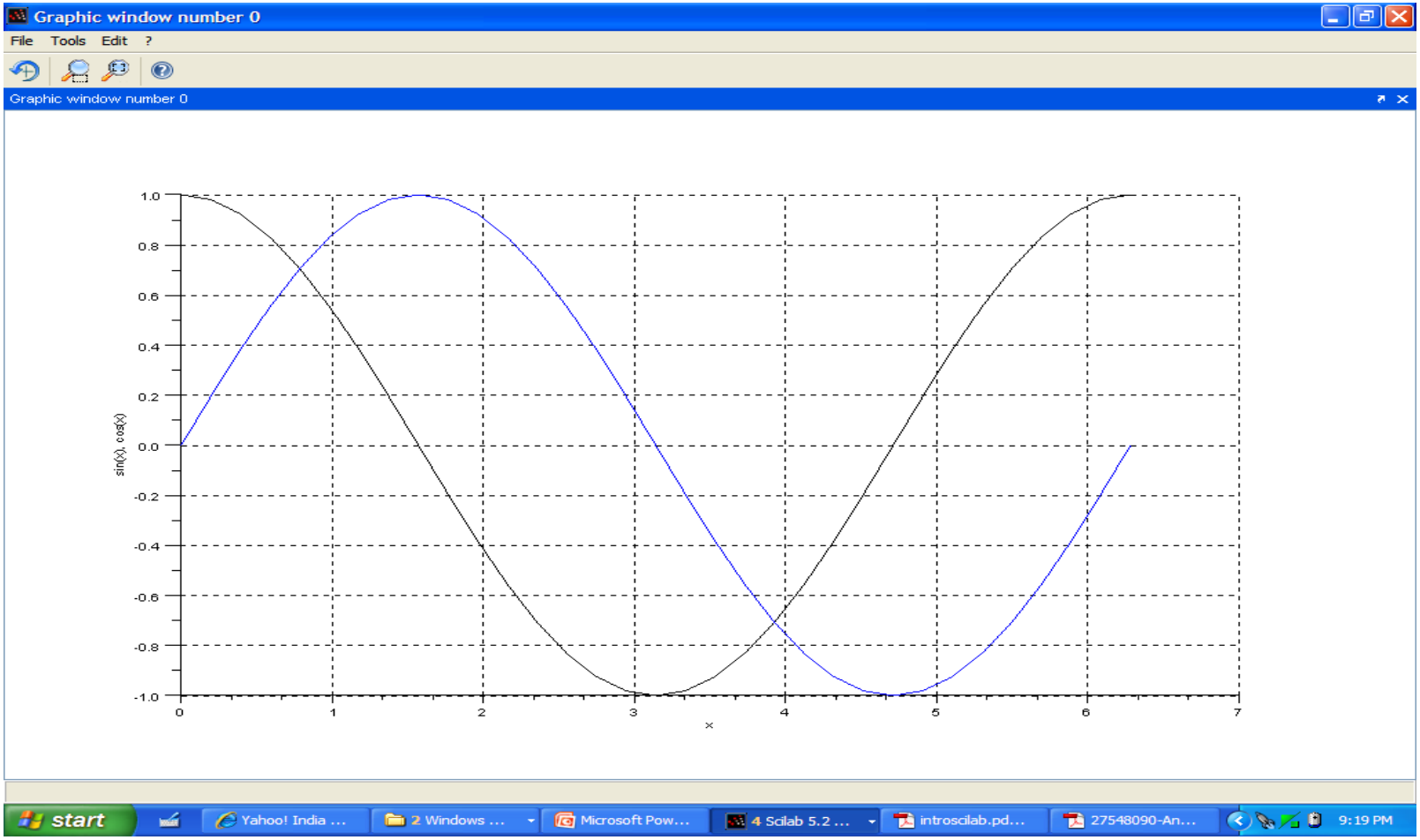
# Polynomials …contd.

```
->p6=poly(c,'x')
 p6  =
                2
   2 + 3x + x
-->p=(1+2*x+3*x^2)/(4+5*x+6*x^2)
 p  =
                 2
   1 + 2x + 3x
   ---------- ------
                 2
   4 + 5x + 6x
-->numer(p)
 ans  =
                2
   1 + 2x + 3x
```


# Plotting Graphs (1)

```
-->x=[0:%pi/16:2*%pi]';

-->y=[cos(x) sin(x)];

-->plot2d(x,y)

-->xgrid

-->xlabel('x')

-->ylabel('sin(x), cos(x)')
```

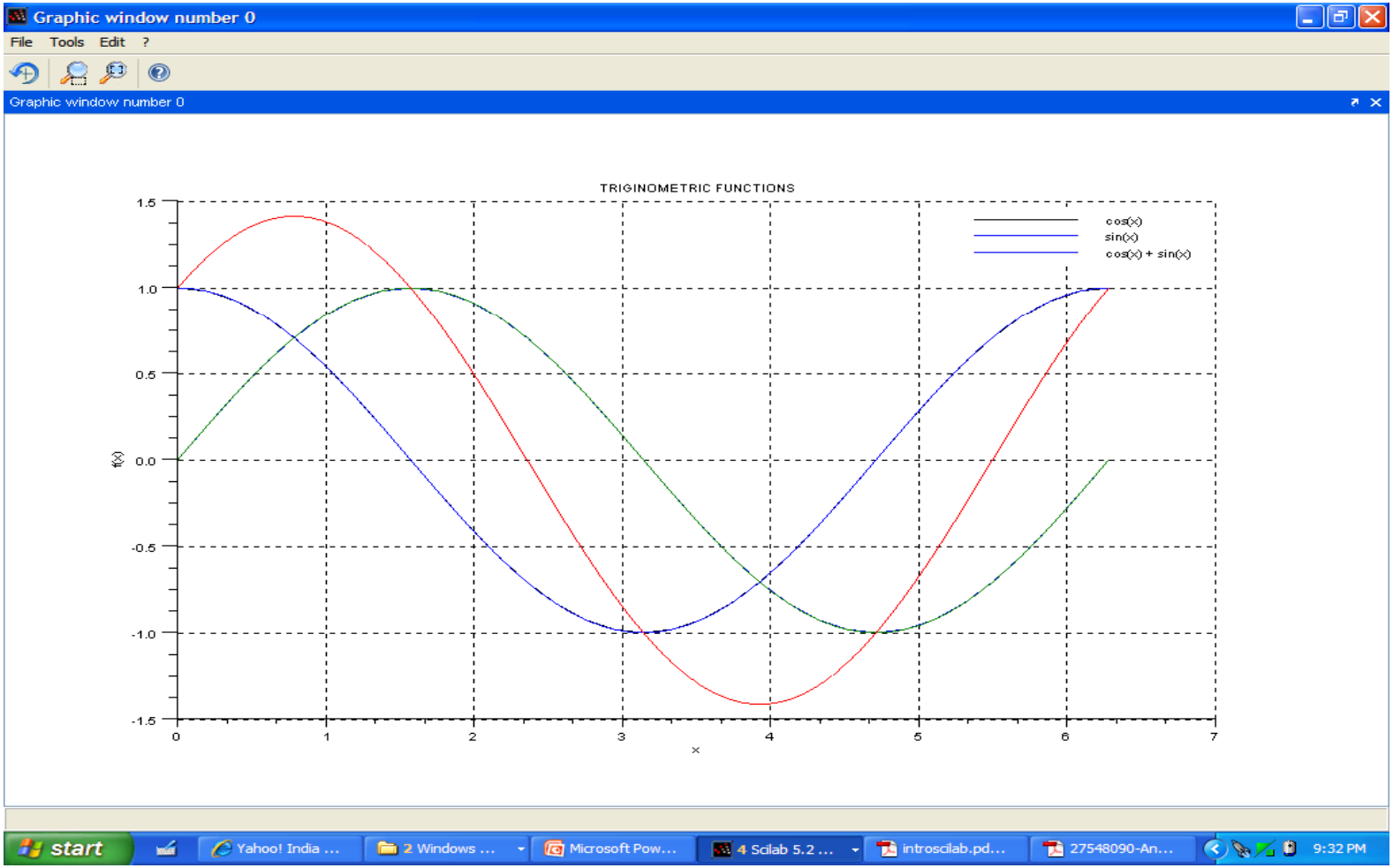
# Plotting Graphs (2)




# Plotting Graphs (3)

```
-->x=[0:%pi/32:2*%pi]';

-->y=[cos(x) sin(x) cos(x)+sin(x)];

-->plot(x, y); xgrid(1);

-->xtitle('TRIGINOMETRIC FUNCTIONS', 'x', 'f(x)');

-->legend('cos(x)', 'sin(x)', 'cos(x) + sin(x)', 1, %F);
```

# Plotting Graphs (4)




# Conclusions

- Scilab is a non-commercial open source platform for Engineering and Scientific computations.
- Scilab is ideal for educational institutes, schools and industries.
- Scilab/Scicos is a better alternative for Matlab/Simulink.
- Students can perform mathematical computations, algorithm development, simulation, prototyping, and data analysis using scilab.
- A valuable tool for researchers at no cost.


# THANK YOU