Introduction to Scilab

Kannan M. Moudgalya
IIT Bombay
www.moudgalya.org
kannan@iitb.ac.in

Scilab Workshop
Bhaskaracharya Pratishtana
4 July 2009

Outline

- Software engineering
  - Implication to scientific computations
  - Scilab as a possible solution
- Scilab - a tutorial introduction
  - Simple arithmetic
  - Matrix operations
  - Vector arithmetic
  - Conditionals
- Plots
- Installation

Software - Bottleneck

- In the past 40 years hardware technology advanced more than 1,000 times
- Software technology about same - people take same time to write programs as before - perhaps twice or thrice faster, with good editors, etc.
- Productivity: 5 lines/man day at IBM with 1 error/mloc (million lines of code)
- The above includes testing, debugging and documentation
- Such a low productivity with sophisticated software tools and that too by experienced programmers
- Others will produce even less

Shortcomings in Science Education

- Scientists and engineers often have no formal course in programming
- Perhaps one course mainly Fortran, perhaps some C
- Objective: to solve engineering problems. Good programming is not a must
- Computer science departments do not spend much time on numerical programming (e.g. does not teach Fortran)
- Shortage of computer scientists
- Scientists and engineers teach programming to their students
- Many of them would have had no formal course in programming

Gap between Computer Scientists and Engineers

- Numerical software is not of interest to computer scientists
- Computers scientists do not carry out numerical simulations
- Their programs work mostly with characters - may be integers
- This, combined with shortcomings in science education, as mentioned in the last slide, makes scientific computations a difficult problem

Idea to Implementation

- R&D Stage: Evaluation of ideas
- Need rapid idea testing workbench
- Come up with idea

  loop

  Test it

  Modify it

  end loop
- When idea works, go to production phase:
  - Convert code for repeated use
  - Use code repeatedly
- Need efficient code

College Education = Idea Testing

- College teaches new things
- Extensive idea testing phase
- Need a tool for idea testing
- Performance of code often does not matter

- Scilab fits the bill

Applied Research Institutions

- High productivity platform for idea testing
- Developing efficient code for tested ideas
- Scilab is one such tool

Scilab

- Environment for numerical computer applications
- Good mathematical library in compiled C code
- Interpreted high level language
- High productivity tool   Scilab:C = C:Assembly
- Can work with Fortran, C: Transition to production phase possible
- Good graphics capability
- Large installed base
- A lot of algorithms implemented in interpreted language as well
- Free
- Check out www.scilab.org or www.scilab.in

History of Scilab

- Idea of Cleve Moler, CS Prof. New Mexico State University
- Was in Linpack, Eispack (robust algorithms) projects
- NSF sponsored project to develop Matlab in Fortran - Free
- Many companies started using this idea
  - Matrix_x
  - CTRL-C
  - Matlab
  - Scilab
- Used extensively for linear algebra, simulation, control system design

Mathematical Library

- Special functions
  - Bessel
  - Gamma
  - Error function
  - Elliptic integral
- Polynomials
  - Characteristic polynomial
  - Roots
  - Multiplication
  - Division
  - Curve fitting
- Matrix condition
  - Condition number
  - 1,2,F and ∞ norms
  - rank

Mathematical Library - Continued

- Matrix functions
  - Exponential
  - Powers
  - Log
  - Square root
- Decomposition & factorisation
  - LU
  - QR
  - SVD
  - Cholesky
  - Schur
  - Inverse
- Signal processing
  - FFT, FFT2, IFFT, IFFT2
  - Convolution
  - Deconvolution
  - Correlation coefficient

Scilab's Language

- C like langugae
- Control flow
  - if
  - while
  - select
  - break
- Procedures
  - Scripts
  - Functions
- Other features
  - Diary
  - Can call C and Fortran programs

Features of Scilab

- Scilab is made up of three distinct parts:
  - An interpreter
  - Libraries of functions (Scilab procedures)
  - Libraries of Fortran and C routines
- It includes hundreds of mathematical functions with the possibility to interactively add programs from various languages (C, Fortran).
- It has sophisticated data structures including lists, polynomials, rational functions, linear systems, etc.

How to Download Scilab?

- Scilab can be downloaded from ftp://ftp.iitb.ac.in/misc_packages/Scilab/
- The website of Scilab is www.scilab.org
- It is distributed in source code format.
- Binaries for Windows95/NT, Unix/Linux/Mac OS/X are also available. All the binary versions include tk/tcl interface.

Usage of Scilab

Simple Arithmetic - 1

```
4+6+12

 ans  =
     22.

a = 4, b = 6; c = 12

 a  =
     4.
 c  =
     12.

a+b+c

 ans  =
    22.
```

Useful Commands

- demos
  - Gives demos on several different things
- apropos
  - Helps locate commands associated with a word
- help
- functional invocation with no arguments
  - Helps draw plots
- diary
  - Stores all commands and resulting outputs

Simple Arithmetic & Display

```
a = 4; b = 6; c = 12;
d = a+b+c
```

```
 d  =
    22.
```

```
d = a+b+c;

d
```

```
 d  =

    22.
```

Simple Arithmetic

```
 format('v',10)
 e = 1/30

e  =
    0.0333333

format('v',20)
 e

e  =
    0.0333333333333333

format('e',20)
 e

e  =
    3.333333333333E-02
```

Simple Arithmetic

```
format('v',10)
x = sqrt(2)/2, y = asin(x)

 x  =

    0.7071068
 y  =

    0.7853982

y_deg = y * 180 /%pi

 y_deg  =

    45.
```

Rounding, Truncation, etc.

```
x = 2.6, y1 = fix(x), y2 = floor(x), y3 = ceil(x), ...
y4 = round(x)
```

```
 x  =

    2.6
 y1  =

    2.
 y2  =

    2.
 y3  =

    3.
 y4  =
```

Different Ways to Specify a List

The following three commands produce identical result:

```
x = [0 .1*%pi .2*%pi .3*%pi .4*%pi .5*%pi .6*%pi ...
.7*%pi .8*%pi .9*%pi %pi];
```

```
x = (0:0.1:1)*%pi;
```

```
x = linspace(0,%pi,11);
```

Vector Operation - 1

```
-->x = (0:0.1:1)*%pi;

-->y = sin(x)

 y  =
         column 1 to 6

! 0. 0.3090170 .5877853  0.8090170  0.9510565  1. !

         column  7 to 11

! 0.9510565  0.8090170  0.5877853  0.3090170  1.225E-16 !
-->y(5)

ans  =
    0.9510565
```

Vector Operation - 2

```
-->a = 1:5, b = 1:2:9

 a  =
 !   1.    2.    3.    4.    5. !
 b  =
 !   1.    3.    5.    7.    9. !

-->c = [b a]

c  =
! 1.    3.    5.    7.    9.    1.    2.    3.    4.    5. !

-->d = [b(1:2:5) 1 0 1]

 d  =
!   1.    5.    9.    1.    0.    1. !
```

Vector Operation - 3

```
-->a, b

 a  =
 !   1.    2.    3.    4.    5. !
 b  =
 !   1.    3.    5.    7.    9. !

-->a - 2

ans  =
 ! - 1.    0.    1.    2.    3. !

-->2*a-b

 ans  =
!   1.    1.    1.    1.    1. !
```

Vector Operation - 5

```
-->a

 a  =
 !   1.     2.     3.     4.     5. !

-->a.^2

ans  =
!   1.     4.     9.     16.     25. !

-->a.^a

ans  =
!   1.     4.     27.     256.     3125. !
```

Vector Operation - 6

```
-->a, b

a  =
 !   1.    2.    3.    4.    5. !
 b  =
 !   1.    3.    5.    7.    9. !

-->a./b

ans  =
!   1.    0.6666667    0.6    0.5714286    0.5555556 !

-->b.\a

ans  =
!   1.    0.6666667    0.6    0.5714286    0.5555556 !
```

Vector Operation - 7

```
-->a, b

a  =
 !   1.     2.     3.     4.     5. !
 b  =
 !   1.     3.     5.     7.     9. !

-->a/b

ans  =
    0.5757576
```

Vector Operation - 8

```
-->a, b

a  =
 !   1.     2.     3.     4.     5. !
 b  =
 !   1.     3.     5.     7.     9. !

-->a\b

ans  =
!   0.      0.      0.     0.      0.  !
!   0.      0.      0.     0.      0.  !
!   0.      0.      0.     0.      0.  !
!   0.      0.      0.     0.      0.  !
!   0.2     0.6     1.     1.4     1.8 !
```

Machine Epsilon

```
-->num=0; EPS=1;

-->while (1+EPS)>1
-->  EPS = EPS/2;
-->  num = num+1;
-->end

-->num

num  =
    53.

-->EPS=2*EPS

EPS  =
    2.220E-16
```

Logical Operators

$==$	equal to
$<$	less than
$>$	greater than
$<=$	less than or equal to
$>=$	greater than or equal to
$<>$ or $\sim=$	not equal to

Use of Machine Epsilon

```
-->x = (-2:2)/3

x  =
! - 0.6666667  - 0.3333333   0.   0.3333333  0.6666667 !

-->sin(x)./x

  !--error    27
division by zero...
```

Use of Machine Epsilon

```
-->x = x+(x==0)*%eps

x  =
! -0.6666667  -0.3333333  2.22E-16  0.3333333  0.6666667 !

-->sin(x)./x

ans  =
! 0.9275547    0.9815841    1.    0.9815841    0.9275547  !
```

Vector Operations Using Logical Operators

```
-->A = 1:9, B = 9-A

A   =
!   1.     2.     3.     4.     5.     6.     7.     8.     9. !
 B   =
!   8.     7.     6.     5.     4.     3.     2.     1.     0. !

-->tf = A==B

tf   =
! F F F F F F F F F !

-->tf = A>B

tf   =
! F F F F T T T T T !
```

Transpose

```
-->c = [1;2;3]

 c  =
!   1. !
!   2. !
!   3. !

-->a=1:3

a  =
!   1.     2.     3.    !

-->b = a'

b  =
!   1. !
!   2. !
!   3. !
```

Submatrix

```
-->A=[1 2 3;4 5 6;7 8 9]

 A  =

!   1.    2.    3. !
!   4.    5.    6. !
!   7.    8.    9. !

-->A(3,3)=0

A  =

!   1.    2.    3. !
!   4.    5.    6. !
!   7.    8.    0. !
```

Submatrix

```
A

 A  =

!   1.    2.    3. !
!   4.    5.    6. !
!   7.    8.    0. !

-->B=A(3:-1:1,1:3)

B  =
!   7.    8.    0. !
!   4.    5.    6. !
!   1.    2.    3. !
```

Submatrix

```
-->A

A  =
!   1.    2.    3. !
!   1.    4.    7. !
!   7.    8.    0. !

-->B=A(:,2)

B  =
!   2. !
!   4. !
!   8. !
```

Submatrix

```
-->b=[5 -3;2 -4]

 b  =
!   5.   - 3. !
!   2.   - 4. !

-->x=abs(b)>2

x  =
! T T !
! F T !

-->y=b(abs(b)>2)

 y  =
!   5. !
! - 3. !
! - 4. !
```

Special Matrices

```
-->zeros(3,3)

 ans  =
!   0.     0.     0.  !
!   0.     0.     0.  !
!   0.     0.     0.  !

-->ones(2,4)

ans  =
!   1.     1.     1.     1.  !
!   1.     1.     1.     1.  !

-->rand(2,1)

ans  =
!   0.2113249  !
!   0.7560439  !
```

Go for Vector Computation

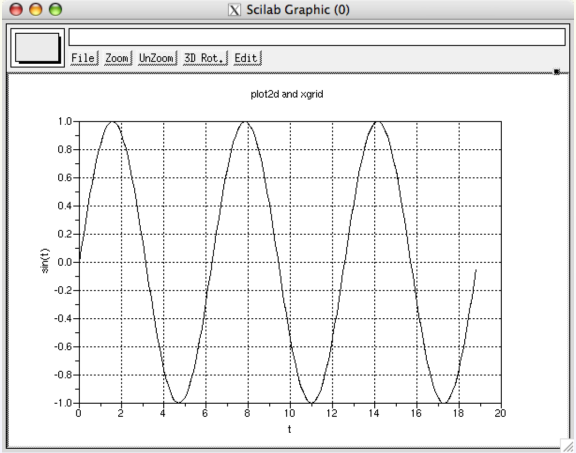
Go for Vector Computation

```
-->a = ones(10000,1);
-->timer()

ans  =
    0.02

-->for i = 1:10000, b(i)=a(i)+a(i); end
-->timer()

ans  =
    0.31

-->c = a+a;
-->timer()

ans  =
    0.03
```

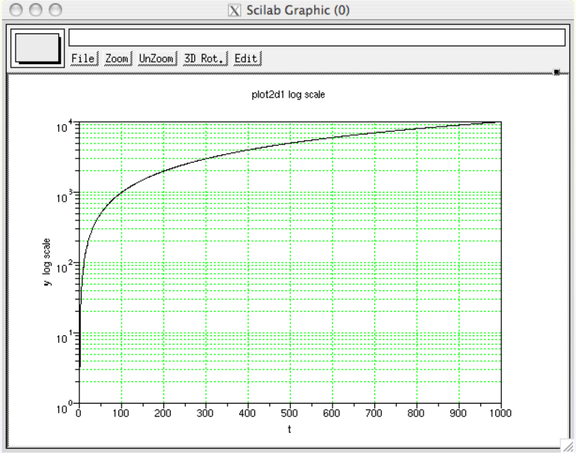
Plots

Go through the Demos!

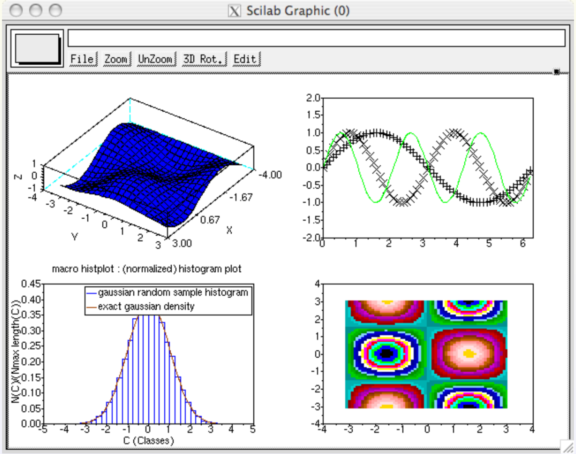
```
t=(0:0.1:6*%pi);
plot2d(t',sin(t)');
xtitle('plot2d and xgrid ','t','sin(t)');
xgrid();
```



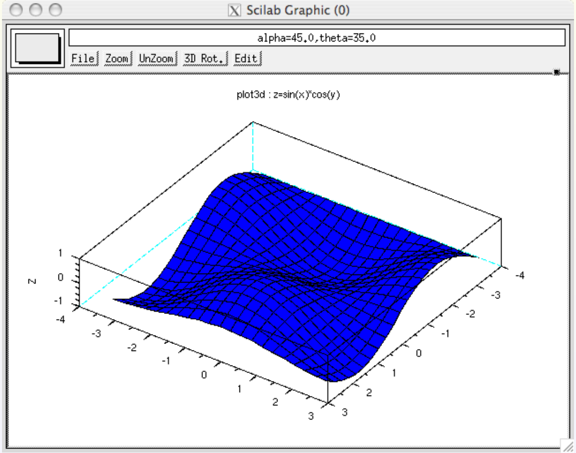
```
plot2d1('enl',1,(1:10:10000)');
xtitle('plot2d1 log scale','t','y  log scale');
xgrid(3);
```



```
subplot(2,2,1); plot3d();
subplot(2,2,2); plot2d();
subplot(2,2,3); histplot();
subplot(2,2,4); grayplot();
```



```
plot3d();
Title=['plot3d : z=sin(x)*cos(y)'];
xtitle(Title,' ',' ');
```



System requirements

- Source version
  - Scilab requires approximately 130 MB of disk storage to unpack and install (all sources included).
  - Also, X Window (X11R4, X11R5 or X11R6), C and Fortran compilers are needed.
- Binary version
  - The minimum requirement for running Scilab (without sources) is about 40 MB when decompressed.
  - Being partially and statically linked, these versions do not require a Fortran compiler.

How to install Scilab?

- Windows
  - Download scilab-4.1.exe
  - Click this file and follow the instructions
  - Launch from its icon on the Desktop.
- Linux
  - Download scilab-4.1.bin.linux-i686.tar.gz
  - Issue the following commands
    - tar zxvf scilab-4.1.bin.linux-i686.tar.gz
    - cd scilab-4.1
    - make
  - The binary is at bin/scilab

Conclusions

- Scilab is ideal for educational institutions, including schools
- Built on a sound numerical platform
- It is free
- Also suitable for industrial applications
- Standard tradeoff between free and commercial applications

Thank you