Introduction to Scilab

Scilab Talk

Department of Chemical Engineering,
IIT Bombay
One Day Teacher Training Programme Sant Gadge Baba Amravati Universtiy Amravati

August 08, 2009

Outline

- Introduction to Scilab
- Basic Features
- Ordinary Differential Equations Integration Examples

Software Issues

- Last 30-40 years: Significant (order of 10^3) advances in hardware technology
- Software: still the same time (improvement of order of $2-3$) to write programs as before
- Low programming productivity
- Scientific Computing?
  - No formal training to scientists and engineers in programming
  - Maybe one or max 2 courses in Fortran or C or C++.
  - Aim: To solve specific engineering problems without investing too much time or knowing "software programming".

Ideas and Implementation

- R&D Stage: Evaluation of ideas
  - Ability to rapidly test different ideas
- When ideas work, then production phase
  - Generalize code for repeated used
  - Use code repeatedly
  - Need efficient code

College Education= Idea Testing

- New things learnt in college
- Extensive idea testing phase
- Need a tool for idea testing
- Performance of code often does not matter
- All of us need to be able to do some basic computations to increase our teaching effectiveness

Scilab fits the bill

Applied Research Institutions

- High productivity platform for idea testing
- Support for “research" topics
- Developing efficient codes for tested ideas

Scilab fits the bill

Scilab

- Scilab 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

Mathematical Functions

- Special Functions
  - Bessel, gamma, error function, elliptic integral
- Polynomial Functions
  - Characteristic polynomial, roots, multiplication, division
- Matrix norms, ranks, condition number
- Equation Solving
  - System of linear and nonlinear equations, differential equations, differential algebraic equations
- Optimization
  - Linear Programming, Quadratic Programming
- Probability and Statistics
  - Distributions, random number generators, random system generators

Mathematical Functions Continued

- Matrix Decomposition and Factorization Tools
- Signal Processing Tools
- Control Related Tools
- Scicos: A block-diagram based environment

Scilab's Language

- C like language
- Control flow
  - if
  - while
  - select
  - break
- Procedures
  - Scripts
  - Functions
- Other Features
  - Diary
  - Interface with C and Fortran routines

Features of Scilab

- Scilab 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 polynomials, rational functions, linear systems, etc.
- Various contributed “toolboxes": eg. support vector machines (SVM).

Where do I get Scilab?

- Can be downloaded from:
  http://www.scilab.org
- Available for various platforms (binaries and source codes):
  Windows (XP, Vista), Linux, Mac

Basic Arithmetic- I

```
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
- diary
  - Stores all commands and resulting outputs

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

```
x  =

       2.6
y2  =

       2.
y3  =

       3.
y4  =

       3.
```

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
-->y(5)

ans   =
    0.9510565
```

Vector Operation - 2

```
-->a = 1:5, b = 1:2:7

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

-->c = [b a]

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

-->d = [b(1:2:4) 1 0 1]

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

Vector Operation - 3

```
-->a, b

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

-->a - 2

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

-->2*a-[b 1]

 ans  =
!   1.     1.     1.     1.     9. !
```

Vector Operation - 4

```
-->a

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

-->a.^2

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

-->a.^a

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

Transpose

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

 c  =
!   1. !
!   2. !
!   3. !
-->a=1:3

 a  =
!   1.     2.     3.    !
-->b = a'

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

Submatrix -1

```
-->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-2

```
-->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-3

```
-->A

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

-->B=A(:,2)

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

Logical Operators

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

Submatrix-4

```
-->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 !
```

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
```

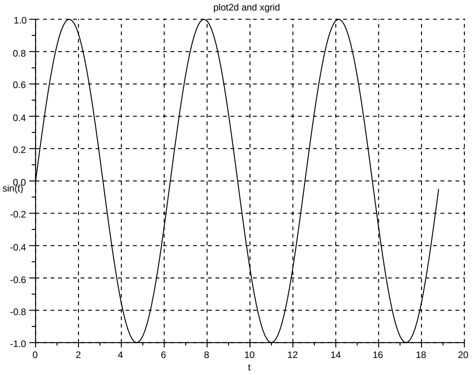
Script File

- Diary to keep record of all executed commands
- Script file: write all working commands
  - When required, simply execute script file
  - No need to work on command prompt

Plots

Go through the Demos!

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



Weight Reduction Model

- Weight of person = x kg
- Tries to reduce weight
- Weight loss per month = 10% of weight
- Starting weight = 100 kg

$$\frac{dx}{dt} = -0.1x$$

Initial conditions:

$$x = 100 \quad \text{at t=0}$$

Determine $x(t)$ as a function of t.

Analytical Solution of Simple Model

Recall the model:

$$\frac{dx}{dt} = -0.1x; \; x(t=0) = 100$$

Cross multiplying, $$\frac{dx}{x} = -0.1dt$$

Integrating both sides from 0 to t,

$$\int \frac{dx}{x} = -0.1 \int dt$$

$$C + \ln x(t) = -0.1t$$

Using initial conditions,

$$C = -\ln 100$$

Thus, the final solution is,

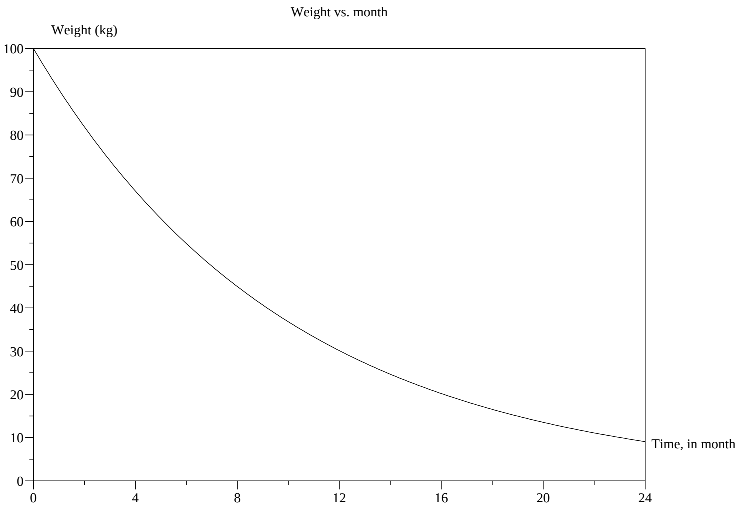
$$\ln \frac{x(t)}{100} = -0.1t \quad \text{or} \quad x(t) = 100e^{-0.1t}$$

- Weight of person = x kg
- Tries to reduce weight
- Weight loss per month = 10% of weight
- Starting weight = 100 kg

$$x(t) = 100e^{-0.1t}$$

Compute and plot for two years, i.e. for 24 months:

```
T=0:0.1:24;
plot2d(T,100*exp(-0.1*T));
xtitle('Weight vs. month','Time in months',...
        'Weight (kg)')
```


Weight vs. month
Weight (kg)
100
90
80
70
60
50
40
30
20
10
0
0
4
8
12
16
20
24
Time, in month

Need for Numerical Solution

- Exact solution is ok for simple models
- What if the model is complicated?
- Consider integrating the more difficult problem:

$$\frac{dx}{dt} = \frac{2 + 18t + 68t^2 + 180t^3 + 250t^4 + 250t^5}{x^2}$$

with initial condition,

$$x(t = 0) = 1$$

- Analytical (i.e. exact) solution difficult to find

Simplest Numerical Solution: Explicit Euler

- Suppose that we want to integrate the following system:

$$\frac{dx}{dt} = g(x, t)$$

with initial condition:

$$x(t = 0) = x_0$$

- Approximate numerical method - divide time into equal intervals: t_0, t_1, t_2, etc.

$$\frac{x_n - x_{n-1}}{\Delta t} = g(x_{n-1}, t_{n-1})$$

- Simplifying,

$$\begin{aligned} x_n - x_{n-1} &= \Delta t \, g(x_{n-1}, t_{n-1}) \\ x_n &= x_{n-1} + \Delta t \, g(x_{n-1}, t_{n-1}) \end{aligned}$$

- Given x_0, march forward and determine x_n for all future n.

Functions

Online function definition

```
deff('[y]=avgfunc(x1,x2)','y=(x1+x2)/2')
a=avgfunc(3,5)
```

Answer will be 4

Functions in Files

Create a file `myavgfunc.sci` with the following three lines:

```
function y=myavgfunc(x1,x2)
y=(x1+x2)/2;
endfunction
```

usage

```
--> getf('myavgfunc.sci') \\
-->a=myavgfunc(3,5);
```

Answer: a=4

Example revisited

Recall the problem statement for numerical solution:

$$\frac{dx}{dt} = \frac{2 + 18t + 68t^2 + 180t^3 + 250t^4 + 250t^5}{x^2}$$

with initial condition,

$$x(t = 0) = 1$$

Recall the Euler method:

$$\frac{dx}{dt} = g(x, t)$$

Solution for initial condition, $x(t = 0) = x_0$ is,

$$x_n = x_{n-1} + \Delta t \, g(x_{n-1}, t_{n-1})$$

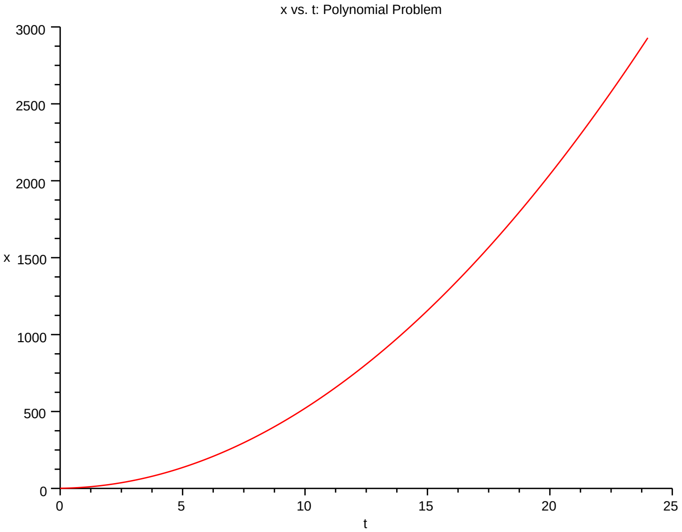
Scilab Code

```
getf("diff1.sci");
getf("Euler.sci");
x0=1; t0=0; T=0:0.1:24;
sol = Euler(x0,t0,T,diff1);
// sol = ode(x0,t0,T,diff1);
plot2d(T,sol)
xtitle('x␣vs.␣t:␣Polynomial␣Problem','t','x')
```

```
function x = Euler(x0,t0,t,g)
n = length(t), x = x0;
for j = 1:n-1
      x0 = x0 + (t(j+1)-t(j))*g(t(j),x0);
      x = [x x0];
end;
```

```
function xdot = diff1(t,x)
xdot = (2+18*t+68*t^2+180*t^3+250*t^4+250*t^5)/x^2;
```

Numerical Solution



Predator-Prey Problem

- Population dynamics of predator-prey
- Prey can find food, but gets killed on meeting predator
- Examples: parasites and certain hosts; wolves and rabbits
- $x_1(t)$ - number of prey; $x_2(t)$ - number of predator at time t
- Prey, if left alone, grows at a rate proportional to x_1
- Predator, on meeting prey, kills it $\Rightarrow$ proportional to $x_1 x_2$

$$\frac{dx_1}{dt} = 0.25x_1 - 0.01x_1x_2$$

- Predator, if left alone, decrease by natural causes
- Predators increase their number on meeting prey

$$\frac{dx_2}{dt} = -x_2 + 0.01x_1x_2$$

- Determine $x_1(t)$, $x_2(t)$ when $x_1(0) = 80$, $x_2(0) = 30$

Explicit Euler for a System of Equations

$$\frac{dx_1}{dt} = g_1(x_1, \ldots, x_n, t)$$

$$\vdots$$

$$\frac{dx_n}{dt} = g_n(x_1, \ldots, x_n, t)$$

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} g_1(x_1, \ldots, x_n, t-1) \\ \vdots \\ g_n(x_1, \ldots, x_n, t-1) \end{bmatrix}$$

$$\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}_t = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}_{t-1} + \Delta t \begin{bmatrix} g_1((x_1, \ldots, x_n)|_{t-1}, t-1) \\ \vdots \\ g_N((x_1, \ldots, x_n)|_{t-1}, t-1) \end{bmatrix}$$

Solution in vector form:

$$\underline{x}_t = \underline{x}_{t-1} + \Delta t \underline{g}(\underline{x}_{t-1})$$

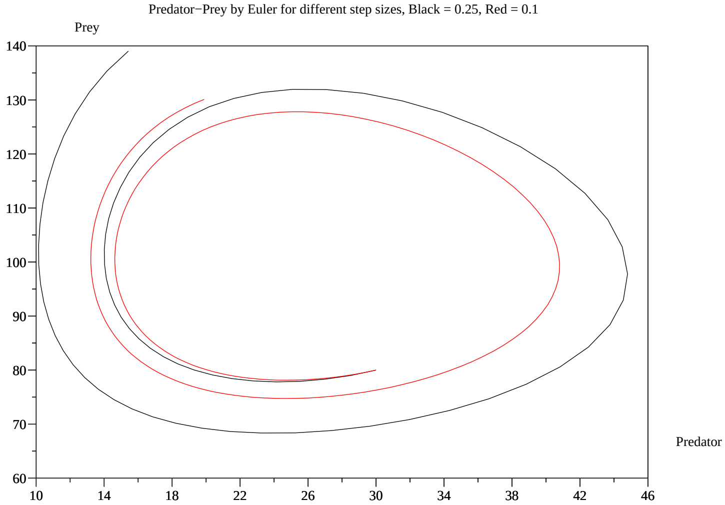
Scilab Code for Predator-Prey Problem

```
getf("pred.sci");
getf("Euler.sci");
x0=[80,30]'; t0=0; T=0:0.1:20; T=T';
sol = Euler(x0,t0,T,pred);
// sol = ode(x0,t0,T,pred);
plot2d(T,sol')
xset('window',1)
plot2d(sol(2,:),sol(1,:))
```

```
function x = Euler(x0,t0,t,g)
n = length(t), x = x0;
for j = 1:n-1
    x0 = x0 + (t(j+1)-t(j))*g(t(j),x0);
    x = [x x0];
end;
```

```
function xdot = pred(t,x)
xdot(1) = 0.25*x(1)-0.01*x(1)*x(2);
xdot(2) = -x(2)+0.01*x(1)*x(2);
```

Predator-Prey Problem: Solution by Euler



As step size increases, solution diverges more from actual.

Better Integration Techniques

- Use other integration techniques (Runge-Kutta 4).
- Use Scilab's ode integrator: "ode" (has various integration routines)
  - Its derived from ODEPACK which has been widely used for last several years

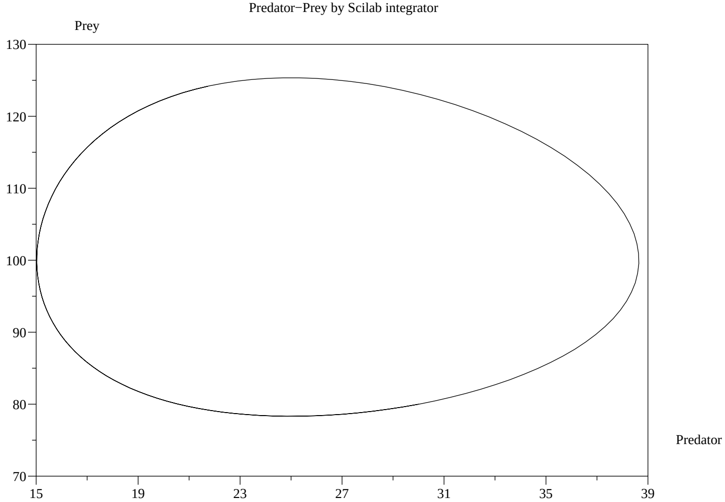
Predator-Prey Problem by Scilab Integrator

Execute the following code, after commenting out `Euler` and uncommenting `ode`:

```
getf("pred.sci");
getf("Euler.sci");
x0=[80,30]'; t0=0; T=0:0.1:20; T=T';
// sol = Euler(x0,t0,T,pred);
sol = ode(x0,t0,T,pred);
plot2d(T,sol')
xset('window',1)
plot2d(sol(2,:),sol(1,:))
```

```
function xdot = pred(t,x)
xdot(1) = 0.25*x(1)-0.01*x(1)*x(2);
xdot(2) = -x(2)+0.01*x(1)*x(2);
```

Predator-Prey Problem: Solution by Scilab Integrator



Use the Scilab built-in integrator to get the correct solution

Conclusions

- Scilab is ideal for educational institutions, including schools
- Built on a sound numerical platform
- It has especially good integrators for differential equations
- It is especially good for polynomial matrix computations
- It is free
- Also suitable for industrial applications
- Scilab:Matlab= Linux:Windows

Thank you