Differential equations using Scilab

Kannan M. Moudgalya
IIT Bombay
kannan@iitb.ac.in

Scilab Workshop
13 September 2010
Outline

- Weight reduction ODE model - analytical solution
- Numerical integration
  - Functions in Scilab
  - Euler’s method
- Predator-prey system
  - Modelling
  - Euler method - user created integrator
  - Backward difference method - built-in function
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:

\( x = 100 \) at \( t=0 \)

Determine \( x(t) \) as a function of \( t \).
Recall the model:

\[
\frac{dx}{dt} = -0.1x \\
x(t = 0) = 100
\]

Cross multiplying,

\[
\frac{dx}{x} = -0.1 dt
\]

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,

\[ \ln \frac{x(t)}{100} = -0.1t \quad \text{or} \quad x(t) = 100e^{-0.1t} \]
Solution, Continued

- 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, plot for 2 years, i.e. for 24 months:

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

Time, in months

Weight (kg)
Need for Numerical Solution

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

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

with initial condition, \( x(t = 0) = 1 \)
- Analytical (i.e. exact) solution difficult to find
- Let us look for a numerical solution
Simple numerical solution: Explicit Euler

- Suppose that we want to integrate
  \[ \frac{dx}{dt} = g(x, t) \]
  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,
  \[ x_n - x_{n-1} = \Delta t \cdot g(x_{n-1}, t_{n-1}) \]
  \[ x_n = x_{n-1} + \Delta t \cdot g(x_{n-1}, t_{n-1}) \]
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, \(x(t = 0) = x_0\)
exec("diff1.scii");
exec("Euler.scii");
x0=1; t0=0; T=0:1:24;
sol = Euler(x0,t0,T,diff1);
// sol = ode(x0,t0,T,diff1);
plot2d(T,sol), pause
plot2d(T,1+2*T+5*T^2,5)
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;
endfunction

function xdot = diff1(t,x)
    xdot = (2+18*t+68*t^2+180*t^3+250*t^4+250*t^5)/x^2;
endfunction
Comparison of numerical, exact solution

x vs. t: Polynomial Problem
Predator-Prey Problem I

- 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-Prey Problem II

- Predator, on meeting prey, kills it \( \Rightarrow \) proportional to \( x_1 x_2 \)

\[
\frac{dx_1}{dt} = 0.25x_1 - 0.01x_1 x_2
\]

- Predator, if left alone, decrease by natural causes

- Predators increase their number on meeting prey

\[
\frac{dx_2}{dt} = -x_2 + 0.01x_1 x_2
\]

- Determine \( x_1(t), x_2(t) \) when \( x_1(0) = 80, x_2(0) = 30 \)
Explicit Euler for a System of Equations

\[
\begin{align*}
\frac{dx_1}{dt} &= g_1(x_1, \ldots, x_n, t) \\
\vdots \\
\frac{dx_N}{dt} &= g_n(x_1, \ldots, x_n, t)
\end{align*}
\]
Explicit Euler for a System of Equations II

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

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

Solution in vector form:

\[
x_t = x_{t-1} + \Delta t g(x_{t-1})
\]
```scilab
exec("pred.sci");
exec("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);
clf();
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;
endfunction

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);
endfunction
As step size increases, the solution diverges.
General method to handle stiff systems

- The predator-prey problem is an example of a **stiff** system
- Results because of sudden changes in the derivative
- Approximation of using previous time values does not work
- General approach to solve this problem:
  \[ x_n = x_{n-1} + \Delta t \, g(x_n, t_n) \]
- Requires solution by trial and error, as \( g(x_n, t_n) \) is unknown
- Scilab has state of the art methods (ode) to solve such systems
General method to handle stiff systems

- Derived from ODEPACK
  - FOSS
  - In use for thirty years
  - Bugs have been removed by millions of users
Execute the following code, after commenting out `Euler` and uncommenting `ode`:

```plaintext
exec("pred.sci");
exec("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);
clf();
plot2d(T,sol')
xset('window',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);
endfunction

plot2d(sol(2,:), sol(1,:))
Use the Scilab built-in integrator to get the correct solution
Parabolic Differential Equations

- Heat conduction equation
- Diffusion equation

\[
\frac{\partial u(t, x)}{\partial t} = c \frac{\partial^2 u(t, x)}{\partial x^2}
\]

- Initial condition:

\[u(0, x) = g(x), \quad 0 \leq x \leq 1\]

- Boundary conditions:

\[u(t, 0) = \alpha, \quad u(t, 1) = \beta, \quad t \geq 0\]

- Let \(u_j^m\) be approximate solution at \(x_j = j\Delta x\), \(t_m = m\Delta t\)

\[
\frac{u_j^{m+1} - u_j^m}{\Delta t} = \frac{c}{(\Delta x)^2} (u_{j-1}^m - 2u_j^m + u_{j+1}^m)
\]
Finite Difference Approach

\[
\frac{u_{j}^{m+1} - u_{j}^{m}}{\Delta t} = \frac{c}{(\Delta x)^2} (u_{j-1}^{m} - 2u_{j}^{m} + u_{j+1}^{m}), \quad \mu = \frac{c\Delta t}{(\Delta x)^2}
\]

\[
u_{j}^{m+1} = \nu_{j}^{m} + \mu (u_{j-1}^{m} - 2u_{j}^{m} + u_{j+1}^{m})
\]

\[
u_{j}^{m+1} = \mu u_{j-1}^{m} + (1 - 2\mu)u_{j}^{m} + \mu u_{j+1}^{m}
\]

Write this equation at every spatial grid:

\[
u_{1}^{m+1} = \mu u_{0}^{m} + (1 - 2\mu)u_{1}^{m} + \mu u_{2}^{m}
\]

\[
u_{2}^{m+1} = \mu u_{1}^{m} + (1 - 2\mu)u_{2}^{m} + \mu u_{3}^{m}
\]

\[\vdots\]

\[
u_{N}^{m+1} = \mu u_{N-1}^{m} + (1 - 2\mu)u_{N}^{m} + \mu u_{N+1}^{m}
\]
\[ u_{1}^{m+1} = \mu u_{0}^{m} + (1 - 2\mu)u_{1}^{m} + \mu u_{2}^{m} \]
\[ u_{2}^{m+1} = \mu u_{1}^{m} + (1 - 2\mu)u_{2}^{m} + \mu u_{3}^{m} \]
\[ \vdots \]
\[ u_{N}^{m+1} = \mu u_{N-1}^{m} + (1 - 2\mu)u_{N}^{m} + \mu u_{N+1}^{m} \]

In matrix form,

\[
\begin{bmatrix}
    u_{1} \\
    u_{2} \\
    \vdots \\
    u_{N}
\end{bmatrix}^{m+1} =
\begin{bmatrix}
    1 - 2\mu & \mu \\
    \mu & 1 - 2\mu & \mu \\
    \vdots & & \ddots & \ddots & \ddots \\
    \mu & 1 - 2\mu & \mu
\end{bmatrix}
\begin{bmatrix}
    u_{1}^{m} \\
    u_{2}^{m} \\
    \vdots \\
    u_{N}^{m}
\end{bmatrix} +
\begin{bmatrix}
    \mu \\
    \mu \\
    \vdots \\
    \mu
\end{bmatrix}
\]
Conclusions

- Scilab is ideal for educational institutions, including schools
- Built on a sound numerical platform
- It has good integrators for differential equations
- It is free
- Also suitable for industrial applications
Thank you