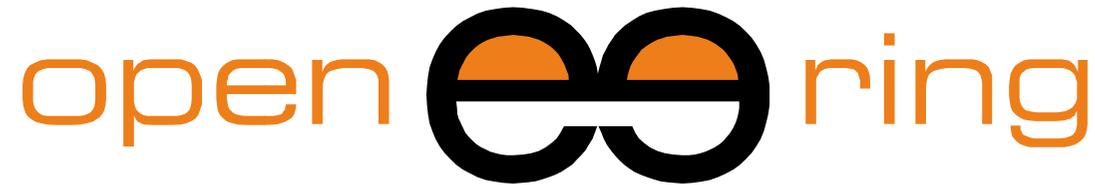


OPEN SOURCE ENGINEERING



A SCILAB PROFESSIONAL PARTNER



MODELING IN SCILAB: PAY ATTENTION TO THE RIGHT APPROACH – PART 1

In this tutorial we show how to model a physical system described by ODE using Scilab standard programming language. The same model solution is also described in Xcos and Xcos + Modelica in two other tutorials.

Level



This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License.



www.openeering.com

Step 1: The purpose of this tutorial

In Scilab there are *three different approaches* (see figure) for modeling a physical system which are described by Ordinary Differential Equations (ODE).

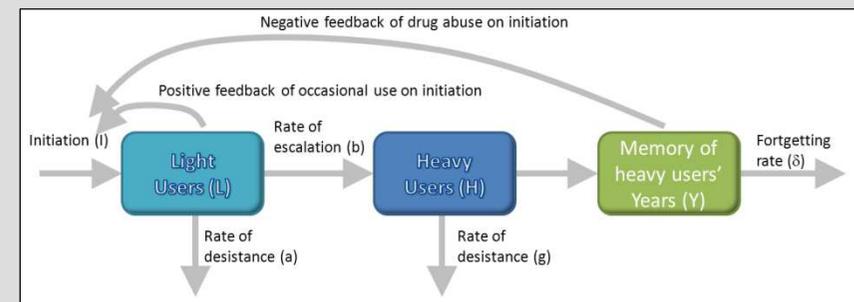
For showing all these capabilities we selected a *common physical system*, the *LHY model* for drug abuse. This model is used in our tutorials as a common problem to show the main features of each strategy. We are going to recurrently refer to this problem to allow the reader to better focus on the Scilab approach rather than on mathematical details.

In this first tutorial we show, step by step, how the LHY model problem can be implemented in Scilab using *standard Scilab programming*. The sample code can be downloaded from the Openeering web site.

1	Standard Scilab Programming
2	Xcos Programming
3	Xcos + Modelica

Step 2: Model description

The considered model is the LHY model used in the study of drug abuse. This model is a continuous-time dynamical system of drug demand for two different classes of users: *light users* (denoted by $L(t)$) and *heavy users* (denoted by $H(t)$) which are functions of time t . There is another state in the model that represents the decaying *memory of heavy users in the years* (denoted by $Y(t)$) that acts as a deterrent for new light users. In other words the increase of the deterrent power of memory of drug abuse reduces the contagious aspect of initiation. This approach presents a positive feedback which corresponds to the fact that light users promote initiation of new users and, moreover, it presents a negative feedback which corresponds to the fact that heavy users have a negative impact on initiation. Light users become heavy users at the *rate of escalation* b and leave this state at the *rate of desistance* a . The heavy users leave this state at the *rate of desistance* g .



Step 3: Mathematical model

The mathematical model is a system of **ODE (Ordinary Differential Equation)** in the **unknowns**:

- $L(t)$, **number of light users**;
- $H(t)$, **number of heavy users**;
- $Y(t)$, **decaying of heavy user years**.

The **initiation function** contains a “spontaneous” initiation τ and a memory effect modeled with a negative exponential as a function of the memory of year of drug abuse relative to the number of current light users.

The problem is completed with the specification of the **initial conditions** at the time t_0 .

The LHY equations system (omitting time variable t for sake of simplicity) is

$$\begin{cases} \dot{L} &= I(L, Y) - (a + b)L \\ \dot{H} &= bL - gH \\ \dot{Y} &= H - \delta Y \end{cases}$$

where the initiation function is

$$I(L, Y) = \tau + L \cdot \max \left\{ s_{\max}, s \cdot e^{-q\frac{Y}{L}} \right\}$$

The LHY initial conditions are

$$\begin{cases} L(t_0) &= L_0 \\ H(t_0) &= H_0 \\ Y(t_0) &= Y_0 \end{cases}$$

Step 4: Problem data

(Model data)

a : the annual rate at which light users quit

b : the annual rate at which light users escalate to heavy use

g : the annual rate at which heavy users quit

δ : the forgetting rate

(Initiation function)

τ : the number of innovators per year

s : the annual rate at which light users attract non-users

q : the constant which measures the deterrent effect of heavy use

s_{\max} : the maximum rate of generation for initiation

(Initial conditions)

t_0 : the initial simulation time;

L_0 : Light users at the initial time;

H_0 : Heavy users at the initial time;

Y_0 : Decaying heavy users at the initial time.

Model data

$$a = 0.163$$

$$b = 0.024$$

$$g = 0.062$$

$$\delta = 0.291$$

Initiation function

$$\tau = 50000$$

$$s = 0.610$$

$$q = 3.443$$

$$s_{\max} = 0.1$$

Initial conditions

$$t_0 = 1970$$

$$L_0 = 1.4 \times 10^6$$

$$H_0 = 0.13 \times 10^6$$

$$Y_0 = 0.11 \times 10^6$$

Step 5: Scilab programming

With the standard approach, we solve the problem using the **ODE** function available in Scilab. This command solves ordinary differential equations with the syntax described in the right-hand column.

In our problem we have:

- Y_0 equal to $[L_0, H_0, Y_0]$ since the unknowns are $L(t)$, $H(t)$ and $Y(t)$;
- t_0 equal to 1970;
- t generated using the command

```
t = Tbegin:Tstep:(Tend+100*%eps);
```

which generates a regularly spaced vector starting from T_{begin} and ending at T_{end} with a time-step T_{step} . A tolerance is added to ensure that the value T_{end} is reached. See also `linspace` command for an alternative solution.
- f a function of three equations in three unknowns which represents the system to be solved.

The syntax of the ode function is

$$y = \text{ode}(y_0, t_0, t, f)$$

where at least the following four arguments are required:

- Y_0 : the **initial conditions** [vector] (the vector length should be equal to the problem dimension and its values must be in the same order of the equations).
- t_0 : the **initial time** [scalar] (it is the value where the initial conditions are evaluated);
- t : the **times steps** at which the **solution** is computed [vector] (Typically it is obtained using the function **Linspace** or using the colon `:` operator);
- f : the **right-hand side of system** to be solved [function, external, string or list].

Step 6: Roadmap

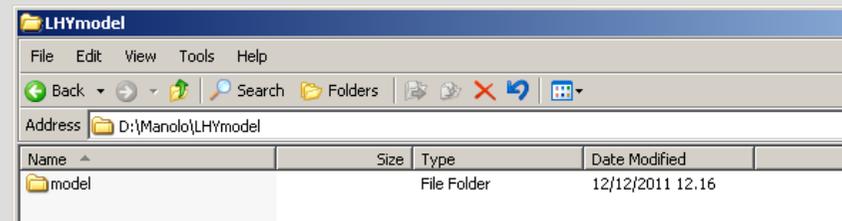
We implement the system in the following way:

- Create a directory where we save the model and its supporting functions. The directory contains the following files:
 - A function that implements the initiation function;
 - A function that implements the system to be solved;
 - A function that plots the results.
- Create a main program that solves the problem;
- Test the program and visualize the results.

- `LHY_Initiation.sci` : The initiation function
- `LHY_System.sci` : The system function
- `LHY_Plot.sci` : The plotting function
- `LHY_MainConsole.sce` : The main function.

Step 7: Create a working directory

We create a directory, name it "model" in the current working folder (`LHYmodel`) as shown in figure.



Step 8: Create the initiation function

Now, we implement the following formula

$$I(L, Y) = \tau + L \cdot \max \left\{ s_{\max}, s \cdot e^{-q \frac{Y}{L}} \right\}$$

as a Scilab function.

The function returns the number of initiated individuals I per years using the current value of Y , L and its parameters (τ, s, q, s_{\max}) . We decide to implement the function parameters as a Scilab `mlist` data structure.

The following code is saved in the file "LHY_Initiation.sci" in the model directory.

```
function I=LHY_Initiation(L, H, Y, param)
// It returns the initiation for the LHY model.
//
// Input:
// L = number of light users,
// H = number of heavy users,
// Y = decaying heavy user years,
// param is a structure with the model parameters:
// param.tau = number of innovators per year,
// param.s   = annual rate at which light users
//            attract non-users,
// param.q   = deterrent effect of heavy users constant,
// param.smax = maximum feedback rate.
//
// Output:
// I = initiation.
//
// Description:
// The initiation function.

// Fetching
tau = param.tau;
s   = param.s;
q   = param.q;
smax = param.smax;

// Compute s effective
seff = s*exp(-q*Y./L);
seff = max(smax,seff);

// Compute initiation (vectorized formula)
I = tau + seff.*L;

Endfunction
```

Step 9: Create the system function

Now, we implement the right hand side of the ODE problem as requested by the Scilab function ode:

$$\begin{cases} \dot{L} &= I(L, Y) - (a + b)L \\ \dot{H} &= bL - gH \\ \dot{Y} &= H - \delta Y \end{cases}$$

All the model parameters are stored in the "param" variables in the proper order.

The following code is saved in the file "LHY_System.sci" in the model directory.

```
function LHYdot=LHY_System(t, LHY, param)
// The LHY system

// Fetching LHY system parameters
a = param.a;
b = param.b;
g = param.g;
delta = param.delta;

// Fetching solution
L = LHY(1,:);
H = LHY(2,:);
Y = LHY(3,:);

// Evaluation of initiation
I = LHY_Initiation(L, H, Y, param);

// Compute Ldot
Ldot = I - (a+b)*L;
Hdot = b*L - g*H;
Ydot = H - delta*Y;

LHYdot = [Ldot; Hdot; Ydot];

Endfunction
```

Step 10: A data plot function

The following function creates a data plot, highlighting the maximum value of each variables. The name of the function is "LHY_Plot.sci" and shall be saved in the model directory.

This function draws the values of L, H, Y and I and also computes their maximum values.

```
function LHY_Plot(t, LHY)
// Nice plot data for the LHY model

// Fetching solution
L = LHY(1,:);
H = LHY(2,:);
Y = LHY(3,:);

// Evaluate initiation
I = LHY_Initiation(L,H,Y, param);

// maximum values for nice plot
[Lmax, Lindmax] = max(L); tL = t(Lindmax);
[Hmax, Hindmax] = max(H); tH = t(Hindmax);
[Ymax, Yindmax] = max(Y); tY = t(Yindmax);
[Imax, Iindmax] = max(I); tI = t(Iindmax);

// Text of the maximum point
Ltext = sprintf(' ( %4.1f , %7.0f)',tL,Lmax);
Htext = sprintf(' ( %4.1f , %7.0f)',tH,Hmax);
Ytext = sprintf(' ( %4.1f , %7.0f)',tY,Ymax);
Itext = sprintf(' ( %4.1f , %7.0f)',tI,Imax);

// Plotting of model data
plot(t,[LHY;I]);
legend(['Light Users';'Heavy users';'Memory';'Initiation']);

// Vertical line
set(gca(),"auto_clear","off");
xpolys([tL,tH,tY,tI;tL,tH,tY,tI],[0,0,0,0;Lmax,Hmax,Ymax,Imax]);

// Text of maximum point
xstring(tL,Lmax,Ltext);
xstring(tH,Hmax,Htext);
xstring(tY,Ymax,Ytext);
xstring(tI,Imax,Itext);
xlabel('Year');

set(gca(),"auto_clear","on");
endfunction
```

Step 11: The main program

The function implements the main program to solve the system.

The code is saved in the file "LHY_MainConsole.sce" in the working directory.

In the main program we use the function **GETD** to import all the developed functions contained in our model directory.

```
// Testing the model using US drug data
clc

// Import LHY functions
getd('model');

// Setting LHY model parameter
param = [];
param.tau = 5e4; // Number of innovators per year (initiation)
param.s = 0.61; // Annual rate at which light users attract
                // non-users (initiation)
param.q = 3.443; // Constant which measures the deterrent
                // effect of heavy users (initiation)
param.smax = 0.1; // Upper bound for s effective (initiation)
param.a = 0.163; // Annual rate at which light users quit
param.b = 0.024; // Annual rate at which light users escalate
                // to heavy use
param.g = 0.062; // Annual rate at which heavy users quit
param.delta = 0.291; // Forgetting rate

// Setting initial conditions
Tbegin = 1970; // Initial time
Tend = 2020; // Final time
Tstep = 1/12 // Time step (one month)
L0 = 1.4e6; // Light users at the initial time
H0 = 0.13e6; // Heavy users at the initial time
Y0 = 0.11e6; // Decaying heavy user at the initial time

// Assigning ODE solver data
y0 = [L0;H0;Y0];
t0 = Tbegin;
t = Tbegin:Tstep:(Tend+100*%eps);
f = LHY_System;

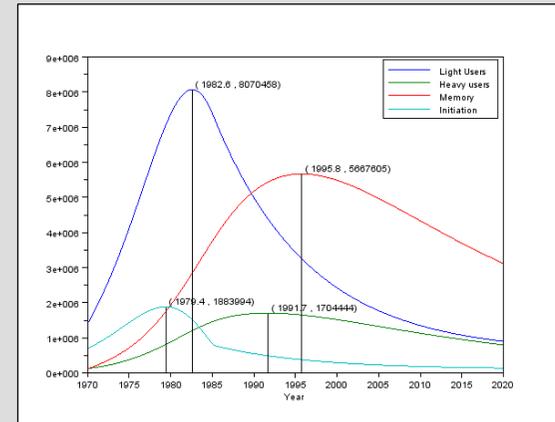
// Solving the system
LHY = ode(y0, t0, t, f);

// Plotting of model data
LHY_Plot(t, LHY);
```

Step 12: Running and testing

Change the Scilab working directory to the current project working directory (e.g. `cd 'D:\Manolo\LHYmodel'`) and run the command `exec('LHY_MainConsole.sce',-1)`.

This command runs the main program and plots a chart similar to the one shown on the right.

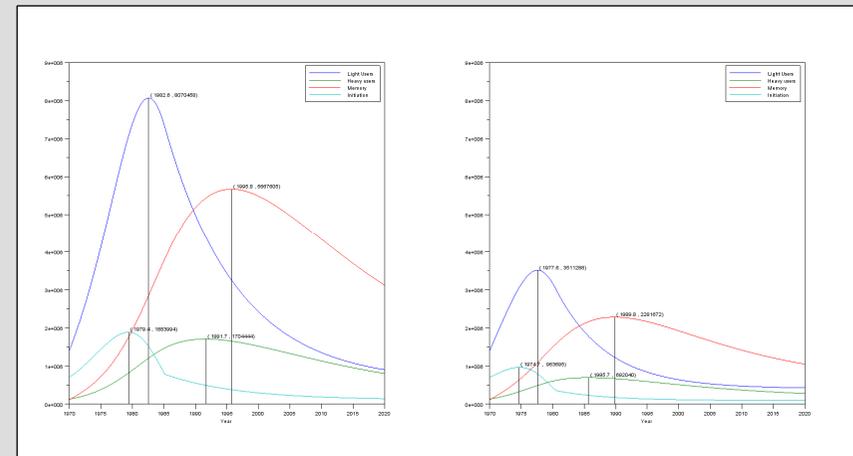


Step 13: Exercise #1

Modify the main program file such that it is possible to compare the “original model” with the same model where the parameter $a = 0.2$. Try to plot the original result and the new simulation results in the same figure for a better comparison.

Hints: Some useful functions

`scf(1)` Set the current graphic figure
`subplot(121)` Divide a graphics window into a matrix of sub-windows (1 row, 2 columns and plot data on first sub-window)
`subplot(122)` Divide a graphics window into a matrix of sub-windows (1 row, 2 columns and plot data on second sub-window)
`a=gca()` Return handle to the current axes
`a.data_bounds=[1970,0;2020,9e+6];` Set bound of axes data



Step 14: Concluding remarks and References

In this tutorial we have shown how the LHY model can be implemented in Scilab with a standard programming approach.

On the right-hand column you may find a list of references for further studies.

1. Scilab Web Page: Available: www.scilab.org.
2. Scilab Ode Online Documentation:
http://help.scilab.org/docs/5.3.3/en_US/ode.html.
3. Openeering: www.openeering.com.
4. D. Winkler, J. P. Caulkins, D. A. Behrens and G. Tragler, "Estimating the relative efficiency of various forms of prevention at different stages of a drug epidemic," Heinz Research, 2002.
<http://repository.cmu.edu/heinzworks/32>.

Step 15: Software content

To report a bug or suggest some improvement please contact Openeering team at the web site www.openeering.com.

Thank you for your attention,

Manolo Venturin

```
-----  
LHY MODEL IN SCILAB  
-----  
  
-----  
Directory: model  
-----  
LHY_Initiation.sci      : Initiation function  
LHY_Plot.sci           : Nice plot of the LHY system  
LHY_System.sci         : LHY system  
  
-----  
Main directory  
-----  
ex1.sce                 : Solution of the exercise  
LHY_MainConsole.sce    : Main console program  
license.txt            : The license file
```