Optimizing Ecosimpro models with Excel-Solver

José Luis Martínez González
Process Engineer; Puertollano Refinery (Spain); Repsol-YPF
jlmartinezgon@repsolypf.com

Abstract

The simulations field is deeply related to the use of optimization strategies to improve the results and achievements of the models. In the first stage, through simulation, a mathematical implementation which emulates a particular physical system is created. The final objective of this implementation is not only a minor knowledge of the physical phenomena, or a qualitative estimation of its response to variations in the initial conditions or physical parameters which defines the problem. The majority of cases a deep and quantitative knowledge of the model is required to face up a systematic study of the model. A second stage, done in the post-modelling step, is to match this model with the real system or, in other cases, with some previous requirements. Optimization strategies let calculate the most suitable values of the parameters which make the model behave as close to a referring situation, which is previously consider as ideal (maximize profits, minimize cost, minimize errors between model and real experiments…). The behaviour of the system and the results of the mathematical model are established by the value of those parameters (inputs) which appear in the physicochemical equations used (mass, heat capacity, transfer coefficients, viscosity, density…).

Keywords: optimization; parameter adjustment; Excel-Solver; discretisation; finite differences; PDE

Introduction

The modelling capabilities of EcosimPro are well established. Its great potential in solving physical systems represented through differential-algebraic equations (DAE) is a fact contrasted with several and challenging simulations solved with this software. The "first stage" is achieved with the functions implemented in the EcosimPro environment (EL language, mathematical wizards, solvers and graphical display of results). However, the further aim of this effort is not creating only a simulation which is correctly implemented in the computer. The final objective is to develop a realistic simulation which exactly represents the real world. Simulations, consequently, must match some requirements, and must be adjusted to make it agree with the real system being modelled. With this premise, after adjusting and tuning the model, any extrapolation of the results will be completely valid.

In this paper a method to optimize models obtained with EcosimPro will be described. It will be done with the embedding capabilities of EcosimPro to run in any ActiveX compatible application. In this case a powerful optimization tool which is integrated in Microsoft Excel is used: SOLVER [(1)].

Solver is an add-in, distributed with Microsoft Excel, which is capable of solving problems of linear programming (LP), nonlinear optimization (NLP), sequential quadratic programming (SQP), generalized reduced gradient (GRG), mixed-integer optimization, etc… with or without restrictions in the value of the variables (bounds on variables), as well as in the value of the equations (general constrains) which define the objective function.

The conjunction of the powerful modelling capabilities of EcosimPro with the robust optimization routines implemented in Excel-Solver enhances greatly the analysis capabilities of simulations in the post-modelling stage.

Configuring the optimization problem

The strategy is based on the interface of communication between EcosimPro and Excel through the dynamic library "Ecoviewer.dll". It lets execute EcosimPro models, change the value of its parameters, reassigning new values, and plot the final results in any program able to compile Visual Basic code.

The Solver optimization philosophy consists in obtaining the optimum value of a cell (target cell) changing the value of a collection of cells (adjustable cells) which define the inputs to the calculation, and depending of their values the value of the target cell is modified.

To assure a correct working of the methodology of model optimization, Solver must be invoked in a target cell where the numeric value of the objective function must be stored. This value must be obtained through a function written in Visual Basic, VBA:

= MyObjectiveFunction(B2:B3)

where as function input (B2:B3) is referenced a range of cells where the parameters of the model are defined. This procedure forces that the value returned by the function will change each time that the value of any of these parameters are changed. All the instructions, to be executed every time that the inputs are altered, are implemented inside the VBA function.

In the case presented, the optimization has been developed to adjust a desortion process simulation to a collection of experimental results obtained in a pilot plant located in the department of Chemical Engineering (University of Valladolid). In order to implement the objective function (minimization of squared error between the simulated data and the experimental results) the Experiment definition inside EcosimPro has been used. The data consist of an array of the amounts desorbed at different times of the desortion process.

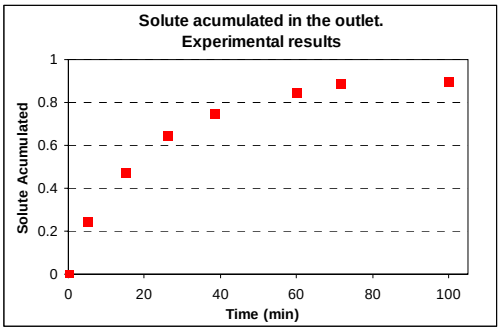


Figure 1. Experimental results obtained in the pilot plant during desortion of polluted soil.

As a part of the EcosimPro experiment a two dimensional array containing these data and time of the sample has been defined (REAL data[10, 2]). To obtain the value of the objective function the next implementation inside EcosimPro has been used (forcing a sequential integration):

```
ndata = 10
p = 1
f_objetivo = 0
WHILE ( p <=ndata )
 INTEG_TO(data[p,1],CINT)
 f_objetivo=f_objetivo+(data[p,2]-y)**2
 p = p + 1
END WHILE
```

A numeric value of the objective function is stored in the variable "f_objetivo" every time this routine is invoked and integrated. This value represents the sum of the squared errors between the experiment and the model.

$$obj = \sum_{i=1}^{N} (y_{\exp} - y_{calc})^2$$

The result will depend on two parameters (mass transfer coefficient and equilibrium solubility constant) as seen next in the physicochemical model which defines the system.

Model of the desortion column

The pilot plant operates in the laboratories of the Chemical Engineering Department of the University of Valladolid. It is designed to desorb the pollutant substances that impregnate soil exposed to hazardous chemical agents (regeneration of soil). To this end, a flow of supercritical CO_2 is introduced, which acts as a solvent of these substances. It desorbs them from the sandy particles and carries them out of the system. The model representing this system is a model in partial differential equations (PDE). For an isothermal system it can be shown that the mass balances [2] throughout the column loaded with the polluted soil are:

Balance of pollutant in the supercritical phase

$$\varepsilon \frac{\partial c}{\partial t} + \frac{u}{L}\frac{\partial c}{\partial z} = -k_g \cdot a \cdot (c - c^*)$$

Initial Condition (t=0) $C = 0$
Boundary Condition at z=0 $C = C_{inlet} = 0$

Balance of pollutant in the solid phase

$$(1-\varepsilon)\frac{\partial c_s}{\partial t} = k_g \cdot a \cdot (c - c^*)$$

Initial Condition (t = 0) $C_s = C_{so}$

Desortion equilibrium at the interphase

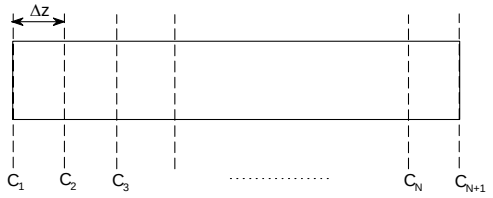
$$c^* = h \cdot c_s$$

The partial differential equation is discretised using the method of finite differences[3] (with nine internal points of discretisation $\Rightarrow$ C1, C2…, C9). This method divides the spatial dimension in several slices in which a differential equation is defined. Therefore the

partial differential equation is transformed into a group of ordinary differential equations and one algebraic equation for each boundary condition. The spatial derivatives are approximated as follows:

$$\frac{\partial c_i}{\partial z} \approx \frac{c_i - c_{i-1}}{\Delta z}$$

$$\frac{\partial^2 c_i}{\partial z^2} \approx \frac{c_{i+1} - 2 \cdot c_i + c_{i-1}}{\Delta z^2}$$



This is a very efficient method which is very easy to understand and implement with EcosimPro:

```
COMPONENT col_desorcion (INTEGER n=9)

DATA
  --Adjust:  h & kg
  REAL tau1=  0.20       -- h
  REAL tau2=  0.015      -- kg
  .....
  ...

DECLS

  REAL cliq[n+1]
  REAL csol[n+1]
  REAL y
  ...

INIT
  FOR (i IN 1,n+1)
    cliq[i]=cliq_ini
    csol[i]=csol_ini
  END FOR
  y=0

CONTINUOUS

  deltaz=1/(n+1)
  cliq[1]=c_entrada --Boundary cond. z=0

--Discretisation by finite differences
--of the mass balances

EXPAND_BLOCK (i IN 2,n+1)

   e*cliq[i]'+u/L*(cliq[i]-cliq[i-1])
   /deltaz= - tau2*av*(cliq[i]-
   tau1*csol[i]) -- Ccon in the liquid

   (1-e)*csol[i]'=tau2*av*(cliq[i]-
   tau1*csol[i]) -- Ccon in the solid

END EXPAND_BLOCK

   (1-e)*csol[1]'=tau2*av*(cliq[1]-
      tau1*csol[1])

-- Accumulation of cliq in the outlet:

  y'=cliq[n+1]

END COMPONENT
```

The adjustment of parameters in this case was reduced to only two: the equilibrium of solubility (h = tau1), and the mass transfer coefficient (k_g = tau2).

Integration in Excel and contents of the objective function

The name and initial values of the parameters to be adjusted has been included in two columns of the Excel spreadsheet. In other cell (target cell), the value of the objective function (variable "f_objetivo") must be transferred every time that those parameters are changed. To achieve this requirement a VBA function is called from this cell.

Basically, this function (its inputs is an Excel Range with the values of tau1 & tau2) is formed by the next instructions:

1- Initialization of the executable file which contains the simulation with the EcoViewer routine *LauchExperimentFile "C:\desorcion.exe"* , which is included in the macro Init()

2- Pause in order to take effect the initialization command. This is achieved with the next code:

```
TimeInit = Timer
Do While Timer – TimeInit < PauseTime
      DoEvents
Loop
```

3- Assign the values of parameters (tau1 and tau2) which are passed as inputs of this VBA function to the simulation file with the EcoViewer function *AssignValues()*

4- Execution of the simulations with the macro Running() ⇒ *Ecoviewer.EcosimViewer.Run*

5- New pause to let the CPU finish the execution. The 'PauseTime' must be adjusted for each particular case to let a correct execution of the code. Its value is related with the CPU speed and the complexity of the simulation. In this

case a value of a 0.1 s. is enough to assure a correct calculation.

6- Reading the variable which stores the value of the objective function inside EcosimPro ("f_objetivo") with the property:
Ecoviewer.ecmExpVars("f_objetivo").Value

7- Output this value as the value of the VBA function to be written in the Excel spreadsheet.

After the step of initialization and execution of the simulations (steps 1 & 4) it has been noted that a short pause in the code is needed to prevent from generating errors in the code. The elapsed time is not the same in all CPUs and surely also depends on the simulation size. Therefore, it should be adjusted in each particular case.

In the appendix the full code of this function is attached to help in future implementations. It includes the basic commands that conforms the skeleton to carry out any optimization strategy via Excel-Solver.

Optimization results

As starting point of the calculation a guess values of the mass transfer coefficient and solubility equilibrium were taken as follows: h = 0.20; kg = 0.015 min^{-1}. The sum of squared error between the simulated results and the experimental data (objective function) was 0.1768763. In figure 2 it is shown the initial state of the simulation and the final situation after attempting the optimization of both parameters. The aim was to minimize the objective function in order to get a set of good values of these parameters which adjust the model to the behaviour founded in the experimental tests.

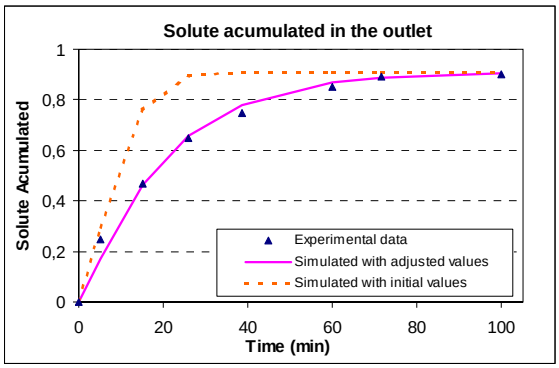


Figure 2. Situation before and after the adjustment of the model parameters. Experimental data are plotted to compare the improvement achieved.

The final objective function obtained with Solver was 0.0079269, which is a value 22 times smaller than the one obtained with the initial parameters. The values of the parameters which represent more accurately the real behaviour are:

h = 0.19381
kg = 0.00045 min^{-1}

Conclusions

It has been shown the necessity of developing models that really represent the real world, or have been optimized to obtain the best performance. In order to ensure this task the simulations have to be complemented with a optimization strategy in the post-modelling stage. Both fields, simulation of the system, and optimization of the model, must be always considered at the same time. The existence of one of them without the other is not a complete view for a deep analysis of the physical system.

The capability of EcosimPro to run its models inside Microsoft Excel let execute Excel Solver as a robust and adequate optimization package. In this article it has been shown that it just needed to write a Visual Basic function to do this task. The typical implementation of this function is included in the appendix. The connection between EcosimPro models and Excel-Solver may be done with not much effort, providing a quick way to optimize any EcosimPro simulation file.

References

(1)- Official site of Frontline Systems Inc. , developers of Excel-Solver software:
http://www.solver.com
(previously http://www.frontsys.com)

(2)- Himmenblau, D.M. & Bischoff, K.B. *Análisis y Simulación de Procesos*. Ed. Reverté

(3)- Finlayson, B.A. *Nonlinear Analysis in Chemical Engineering*. Ed. McGraw-Hill

(4)- Excel files distributed with EcosimPro 3.1 where is shown the VBA code to create an interface between EcosimPro and Excel. ("Libro1.xls")

(5)- Martinez Gonzalez, J.L. *Parameters Optimization and Adjustment by the Simplex Method (Nelder-Mead).* 1st Meeting of EcosimPro Users, UNED, Madrid, 3-4 May 2001. Available at: http://www.ecosimpro.com

[6]- Cristea S. & Prada C. *Real-time Optimization Environment with EcosimPro*. 1st Meeting of EcosimPro Users, UNED, Madrid, 3-4 May 2001. Available at: http://www.ecosimpro.com

Appendix

Code of the VBA function to be written in order to optimize any EcosimPro simulation (with a predefined objective function) using Excel-Solver. It is recommended to take a look previously to the examples provided with Ecosimpro 3.1 ("Libro1.xls") which show the interface with Excel and provide a general understanding of the subject.

```
Function objfun(ByVal Changing As Excel.Range) As Variant

   Dim ObjectiveFunction As Variant, HoraAct As Variant, LastRange As Range

'1- Initialization: (previously to use this function the simulation has to be loaded → Call
InitOpen:
   ' Set xMonitor = Nothing / Set xECM = New clsECM / Set xMonitor = xECM.mMonitor)
   Call Hoja1.Init ' EcoViewer.EcoSimViewer.LaunchSimulationFile("C:\desorcion.exe")

'2- Wait a short time; if not, there is problem in assigning new values:

   HoraAct = Timer
   Do While Timer - HoraAct < 0.10
      DoEvents
   Loop

'3- Assign new values to the parameters, read the variable name and new value from Excel
spreadsheet:

   Dim Item As Excel.Range , NewValValue as Variant, mVarName As String, AssignValue as
Variant

   For Each Item In Changing
      Item.Activate
      Set LastRange = Item
      ' temp = AssignValue() 'AssignValue() is a suplementary function to assign new values
      Dim w As ecmExpVar
      Set w = Nothing
      NewValValue = LastRange.Value
      AssignValue = CDbl(NewValValue)
      mVarName = Cells(CurrentCell.Row, CurrentCell.Column - 1).Value
      Set w = xMonitor.ecmExpVars(mVarName)
      w.Value = AssignValue
   Next Item

'4- Execute the simulation:

   Call Hoja1.Running ' EcoViewer.EcoSimViewer.Run

'5- Wait a short time; if not, there is problem at the end of the execution:

   HoraAct = Timer
   Do While Timer - HoraAct < 0.08
      DoEvents
   Loop

'6- Read the value of the objective function from Ecosimpro when the integration has
concluded:

   Dim v As ecmExpVar
   Set v = Nothing
   ' "f_objetivo": name of object.function inside EcosimPro
   Set v = xMonitor.ecmExpVars("f_objetivo")
   ObjectiveFunction = v.Value

'7- Output the value of the objective function as the value of the VBA function:

   objfun = ObjectiveFunction

End Function
```