DEVELOPMENT VIA THE WEB OF SIMULATION TOOLS BASED ON ECOSIMPRO

Carmen G Moles, Antonio A Alonso and Julio R Banga*

Process Engineering Group, IIM-CSIC
c/ Eduardo Cabello 6, 36208, Vigo, Spain

Abstract

This work presents a methodology for developing dynamic simulation modules via the Web. One of the critical stages in this process includes the formulation of the mathematical problem and its numerical implementation. To this effect we have used EcosimPro, taking advantage of its capability to generate C++ code which can be exported to other environments. Reusing this C++ code, a dynamic library is generated to simulate the model and it can also be used from Matlab to generate graphics. Lastly, a user interface is created and Web-enabled through standard browsers.

Key Words: EcosimPro, Simulation dynamics, Graphics interface, “Web-enabling”, Matlab

1 INTRODUCTION

Simulation via the Web employs a classic client-server schema and has a series of well-known advantages, above all for users:

- It does not involve the acquisition of new hardware (a client could be a single PC and a simple browser)
- It does not involve buying, installing and maintaining simulation software
- It reduces IT personnel costs

The advantages are just as interesting from the point of view of the provider of the simulation-based service:

- There is complete control over the distribution and use of the software
- Software maintenance and updating is simplified

Our team has been developing dynamic models of operations and plants in the biotechnology and food sectors for the past decade. In recent years we have advocated web-enabling these applications in order to facilitate their use by the aforementioned industrial sectors. To achieve this, we have developed a procedure based on dynamic models created with EcosimPro which enables Web applications to be efficiently developed, maintained and documented.

2 DEVELOPMENT OF SIMULATORS VIA THE WEB

The process of creating simulation modules via the Web consists in carrying out the following steps:

- Construction and verification of the mathematical model of the system
  - π Implementation by means of the EL language
  - π Generation of the C++ code
  - π Generation of the corresponding dynamic library
- Creation of the user graphics interface
- Web-enabling through standard browsers

Each of these steps is discussed in detail in the following sections, ending with a practical example of the simulation of thermal processing in cylindrical containers.

3 CONSTRUCTION AND VERIFICATION OF THE MATHEMATICAL MODEL

The first step in the construction of the simulation module is the development of the mathematical model. In this stage we have used the EcosimPro simulation environment mainly because of the facilities it offers in the implementation of dynamic systems, which even make it possible to process systems with discrete events.

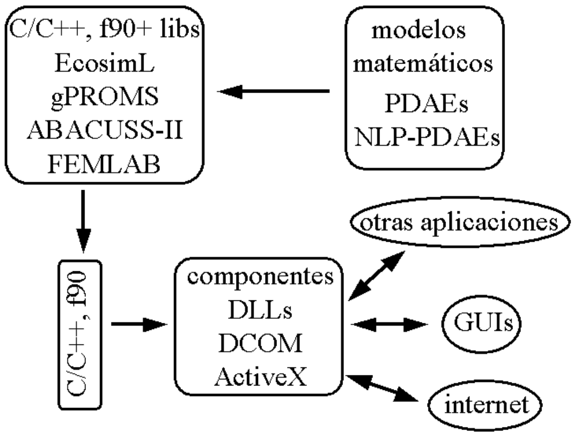
Since EcosimPro also makes it possible to export the C++ numerical code which is created after component compilation, it can be re-used in a wide range of applications. This facilitates the creation of graphical user interfaces (GUIs), its integration with other calculation applications (optimisers, controllers, etc.) or even office automation.

In the second stage, the C++ code generated by EcosimPro is used to create a dynamic library which simulates the system and which can be called from different environments.

This dynamic library was created so that it would be compatible with Matlab. This made it necessary to develop a source code consisting of two parts: a computational routine (C++ exportable from EcosimPro) and a gateway for Matlab (an example is included in the Appendix). The latter facilitates the correct sizing and assignment of the calculation model input and output variables. In other words, its main function is to connect the computational routine generated from the model written in EL with Matlab.

The dynamic library obtained can be executed from other programs and it can also be used to establish client/server relations between different environments. Figure 1 gives a more intuitive illustration of the complete process followed till now.

Figure 1: Steps followed in the Generation of Simulator Components



The dynamic library was validated by comparing its behaviour when the system was simulated with real experimental data and when it was simulated with data obtained through other simulators.

4 GRAPHIC USER INTERFACE (GUI) AND WEB-ENABLING

The term user interface refers to the path between a program and its users. This path may be graphical, as it is in this particular case. The speed with which a program interacts with the user also constitutes an important part of the interface.

Selection of the correct path guarantees adequate use of the tools as well as success. It manages data input/output in addition to pre/postprocessing tasks.

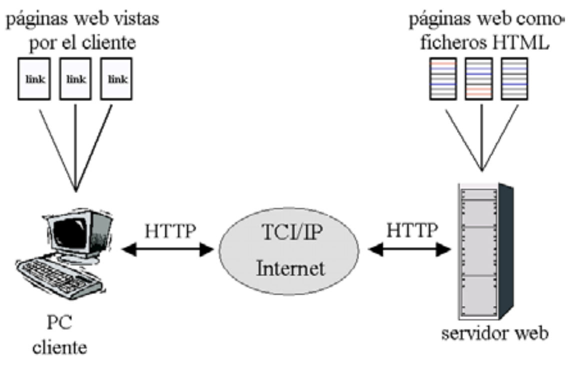
From among the numerous possibilities which exist for developing GUIs that are afterwards accessible via the Web, we have selected a classic client-server schema comprising the following elements:

1. Generation of data input forms via HTML and CGI (Common Gateway Interface)
2. Subsequent to postprocessing, the generation of graphs and tables of results by means of Matlab and dynamic HTML
3. Web-enabling using the Matlab WebServer toolbox and a standard server (eg Apache)

In the final stage (3), the application is web-enabled so that it can be accessed from any computer using standard browsers and without having to install any software in the client machine.

On the other hand, the Matlab WebServer is relatively easy to implement and maintain, and it provides for straightforward access to the great postprocessing capabilities of Matlab.

Figure 2: Client-Server Schema


páginas web vistas por el cliente
link
link
link
páginas web como ficheros HTML
HTTP
TCI/IP
Internet
HTTP
PC cliente
servidor web

5 EXAMPLE: SIMULATION VIA THE WEB OF AN INDUSTRIAL THERMAL STERILISATION PROCESS

The main objective is to provide users from different industries and technological centres with a tool which enables them to evaluate, by means of simulation, the effects on the quality and safety of foods treated in a thermal process.

5.1 MATHEMATICAL MODEL

The mathematical model used is that of Banga and Co [1], where:

Fourier equation and boundary conditions:

$$\frac{dT}{\partial t} = \alpha\left(\frac{\partial^2 T}{\partial r^2} + \frac{1}{r}\frac{\partial T}{\partial r} + \frac{\partial^2 T}{\partial z^2}\right) \qquad (1)$$

$$\frac{\partial T(r,L)}{\partial z} = T_{retort}$$
$$\frac{\partial T(R,z)}{\partial r} = T_{retort} \qquad (2)$$
$$\frac{\partial T(0,z)}{\partial r} = 0$$
$$\frac{\partial T(r,0)}{\partial z} = 0$$

Kinetics of destruction of micro-organisms and nutrients:

$$\frac{dX(r,z)}{dt} = -\frac{2.3025}{D^*_{microorg}} X(r,z) \cdot e^{2.3025(T(r,z)-T^*_{microorg})/Z_{microorg}}$$
$$\frac{dS(r,z)}{dt} = -\frac{2.3025}{D^*_{nutrient}} S(r,z) \cdot e^{2.3025(T(r,z)-T^*_{nutrient})/Z_{nutrient}} \qquad (3)$$

Calculation of final average concentrations of micro-organisms and nutrients:

$$X_{media} = \frac{2}{V_t} \int_0^R \int_0^L rX(r,z) dr \cdot dz \qquad (4)$$
$$S_{media} = \frac{2}{V_t} \int_0^R \int_0^L rS(r,z) dr \cdot dz$$

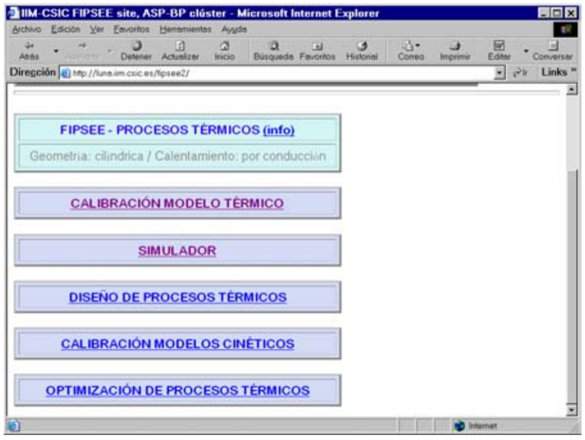
Value of lethality at the critical point F_c and total F_s:

$$\frac{dF_c}{dt} = 10^{(T(0,0)-T^*_{microorg})/Z_{microorg}} \qquad (5)$$
$$F_S = -D^*_{microorg} \log\left(\frac{X_{media}}{X_0}\right)$$

5.2 SIMULATION MODULE

From the first window (see Figure 3) the user can choose among the options to simulate a process, calibrate the model, design processes (temperature/time profile) for certain final sterility conditions, and optimise said process to ensure, for example, maximum final quality respecting the microbiological safety restrictions.

Figure 3: First Window of the Simulation Module


IIM-CSIC FIPSEE site, ASP-BP clúster - Microsoft Internet Explorer

Dirección http://luna.iim.csic.es/fipsee2/

FIPSEE - PROCESOS TÉRMICOS (info)

Geometria: cilindrica / Calentamiento: por conducción

CALIBRACIÓN MODELO TÉRMICO

SIMULADOR

DISEÑO DE PROCESOS TÉRMICOS

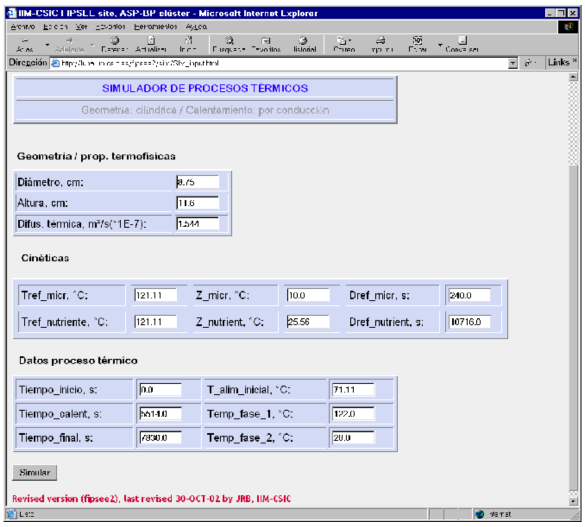
CALIBRACIÓN MODELOS CINÉTICOS

OPTIMIZACIÓN DE PROCESOS TÉRMICOS

Archivo Edición Ver Favoritos Herramientas Ayuda
Atrás Adelante Detener Actualizar Inicio Búsqueda Favoritos Historial Correo Imprimir Editar Conversar
Links
Internet

After selecting in this case the simulation of thermal processing, a second window is displayed (see Figure 4) from which different system parameters can be selected (geometry, thermophysical properties, kinetic parameters, operating profile of the thermal process, etc).

Figure 4: Second Window of the Simulation Module. Introduction of the Parameters into the Model


IIM-CSIC FIPSEE site, ASP-BP clúster - Microsoft Internet Explorer

SIMULADOR DE PROCESOS TÉRMICOS

Geometria: cilindrica / Calentamiento: por conducción

Geometria / prop. termofisicas

Diámetro, cm:	8.75
Altura, cm:	11.6
Difus. térmica, m²/s(*1E-7):	1.544

Cinéticas

Tref_micr, °C:	121.11	Z_micr, °C:	10.0	Dref_micr, s:	240.0
Tref_nutriente, °C:	121.11	Z_nutrient, °C:	25.56	Dref_nutrient, s:	10716.0

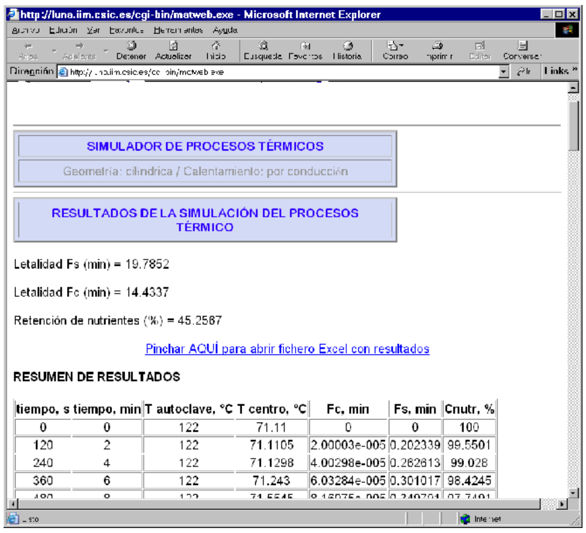
Datos proceso térmico

Tiempo_inicio, s:	0.0	T_alim_inicial, °C:	71.11
Tiempo_calent, s:	5514.0	Temp_fase_1, °C:	122.0
Tiempo_final, s:	7830.0	Temp_fase_2, °C:	20.0

Revised version (fipsee2), last revised 30-OCT-02 by JRB, IIM-CSIC

Finally, we obtain the tabulated results of the simulation (HTML and Excel) and figures which facilitate their analysis (examples are given in Figures 5 and 6).

Figure 5: Presentation of Results


http://luna.iim.csic.es/cgi-bin/matweb.exe - Microsoft Internet Explorer
Archivo Edición Ver Favoritos Herramientas Ayuda
Detener Actualizar Inicio Búsqueda Favoritos Historial Correo Imprimir Editar Conversar
Dirección
Links

SIMULADOR DE PROCESOS TÉRMICOS

Geometría: cilíndrica / Calentamiento: por conducción

RESULTADOS DE LA SIMULACIÓN DEL PROCESOS TÉRMICO

Letalidad Fs (min) = 19.7852

Letalidad Fc (min) = 14.4337

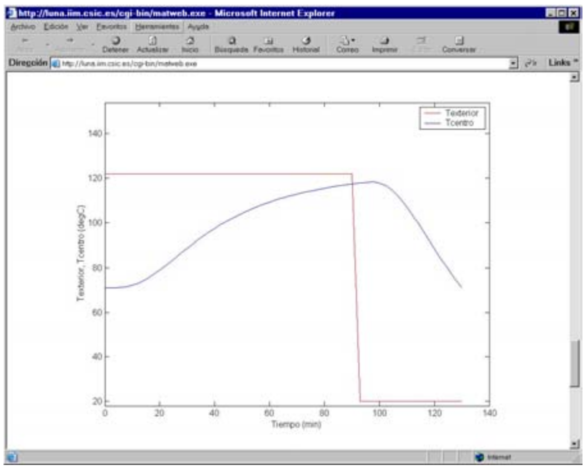
Retención de nutrientes (%) = 45.2567

Pinchar AQUÍ para abrir fichero Excel con resultados

RESUMEN DE RESULTADOS

tiempo, s	tiempo, min	T autoclave, °C	T centro, °C	Fc, min	Fs, min	Cnutr, %
0	0	122	71.11	0	0	100
120	2	122	71.1105	2.00003e-005	0.202339	99.5501
240	4	122	71.1298	4.00298e-005	0.262613	99.028
360	6	122	71.243	6.03284e-005	0.301017	98.4245

Figure 6: Presentation of Results (II)



6 CONCLUSIONS

In this work we have presented the use of EcosimPro as a basic tool for developing dynamic simulators which can be used on the Internet. Combing the C++ code created on the basis of EcosimPro with Matlab and a standard Web server enables these types of applications to be easily developed and maintained.

References

[1] Banga, J.R., Martín, R.P., Gallardo, J.M. and Casares, J.J., (1991) "Optimization of Thermal Processing of Conduction-heated Canned Foods: Study of Several Objective Functions", J. Food Engineering, 14(1), p. 25-51

[2] Schiesser, W.E., (1991) The Numerical Method of Lines, Academic Press, New York

[3] Schiesser, W.E., (1994) Computational Mathematics in Engineering and Applied Science: ODEs, DAEs and PDEs, CRC Press, Inc

[4] Silebi, C. and Schiesser, W. (1992) Dynamic Modeling of Transport Process Systems, Academic Press, London

Appendix

Included in this appendix is an example of the gateway and the definition of the experiment which are used to create the dynamic library which simulates the model (in addition to the C++ code exported from EcosimPro).

The variables called xinfo, vx, and nvar are input variables and their values are assigned from Matlab. These variables declare the initial state of the dynamic variables, integration parameters, etc; that is, they define the experiment of the model.

On the other hand, variables fj, yt[1,100], xinfopath[1,100] and xinfomatrix[200,10] are the output variables of the dynamic simulation that store the simulation results.

· *Definition of the experiment:*

```
#include <math.h>
#include <stdio.h>
#include "EstCylind.examp1.h"
#include "mex.h"

double main(double *xinfo, double *vx, double nvar,
double *fj,double *yt, double *xinfopath)
{
int flag1=0;

        static EstCylind _examp1 *exp = NULL;
        if (exp== NULL){
                exp = new EstCylind _examp1;
                initEcosim(exp);
        }
try{

g_logProgramme = false;
g_warnProgramme= false;
g_traceProgramme= false;
g_pprProgramme  = false;


// VARIABLES DINAMICAS INICIALES:
  exp->setValueReal("T_initial",vx[1]);
// INTEGRACION:
  exp->ABS_ERROR = vx[2];
  exp->REL_ERROR = vx[3];
  exp->REL_ERROR_NL = vx[4];

exp->IMETHOD = DASS_SPARSE;

exp->TIME  = vx[5];
exp->CINT  = vx[6];
exp->TSTOP = vx[7];

while (exp->INTEG_CINT() != INTEG_END)
          {
xinfopath[flag1]= exp->getValueReal("TIME");

xinfomatrix[flag1+800]=
exp->getValueReal ("T[1,1]");

xinfomatrix[flag1+1200]=
exp->getValueReal("T[11,1]");
flag1 += 1;
          }

// SALIDA:
yt[1]  = exp->getValueReal("Tfood[1,1]");
yt[2]  = exp->getValueReal("Tfood[11,1]");
}

catch(...) {
          }
}
```

· *Definition of the gateway:*

```
void mexFunction(int nlhs, mxArray *plhs[ ],int nrhs,
const mxArray *prhs[ ])
{
double *xinfo;
double *vx;
double nvar;
double *fj;
double *yt;
double *xinfopath;
double *xinfomatrix;

/* Crea una matriz para los argumentos de retorno */

plhs[0]=mxCreateDoubleMatrix(1,1,mxREAL);
plhs[1]=mxCreateDoubleMatrix(1,100,mxREAL);
plhs[2]=mxCreateDoubleMatrix(1,100,mxREAL);
plhs[3]=mxCreateDoubleMatrix(200,10,mxREAL);

/* Asigna un puntero a cada entrada y salida  */

xinfo= mxGetPr(prhs[0]);
vx    = mxGetPr(prhs[1]);
nvar = mxGetScalar(prhs[2]);
fj      = mxGetPr(plhs[0]);
yt     = mxGetPr(plhs[1]);
xinfopath = mxGetPr(plhs[2]);
xinfomatrix = mxGetPr(plhs[3]);

/* Llamada a la rutina computacional main */
```

```
main(xinfo, vx, nvar, fj, yt, xinfopath,xinfomatrix);
```