DESARROLLO DE HERRAMIENTAS DE SIMULACIÓN VIA WEB BASADAS EN ECOSIMPRO

Carmen G. Moles, Antonio A. Alonso y Julio R. Banga*

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

Resumen

En este trabajo se presenta una metodología para el desarrollo de módulos de simulación dinámica vía web. En este proceso, una de las etapas críticas engloba la formulación del problema matemático y su implementación numérica. Para ello hemos utilizado EcosimPro aprovechando su capacidad para generar código C++ exportable a otros entornos. Reutilizando este código C++, se genera una librería dinámica encargada de la simulación del modelo y utilizable desde Matlab, que se encarga de la generación de gráficos. Por último, se crea un interface de usuario y se habilita su uso remoto via navegadores estándar.

Palabras Clave: EcosimPro, Simulación dinámica, Interface gráfico, "web-enabling", Matlab.

1 INTRODUCCION

La simulación vía web emplea un esquema clásico cliente-servidor y tiene una serie de ventajas bien conocidas, sobre todo para los usuarios:

- no implica adquirir nuevo hardware (un cliente puede ser un PC sencillo y un simple navegador)
- no implica la compra, instalación y mantenimiento de software de simulación
- disminuye costes de personal informático

Desde el punto de vista del proveedor del servicio basado en simulación, las ventajas son igualmente interesantes:

- existe un control total sobre la distribución y uso del software
- se simplifica el mantenimiento y actualización del software

En nuestro grupo se vienen desarrollando modelos dinámicos de operaciones y plantas del sector biotecnológico y alimentario desde hace una década. En los últimos años, hemos apostado por realizar un "web-enabling" de estas aplicaciones con el fin de poder facilitar su uso por parte de dichos sectores industriales. Para ello, hemos desarrollado un procedimiento que parte de modelos dinámicos creados con EcosimPro y que permite desarrollar, mantener y documentar aplicaciones web de forma eficiente.

2 DESARROLLO DE SIMULADORES VÍA WEB

El proceso de creación de los módulos de simulación vía web que hemos adoptado consta de los siguientes pasos:

- Elaboración y comprobación del modelo matemático del sistema
  - π implementación mediante el lenguaje EL
  - π generación del código C++
  - π generación de la librería dinámica correspondiente
- Creación del interface gráfico de usuario
- Habilitar su uso remoto mediante navegadores estándar ("web-enabling")

En las siguientes secciones se detalla cada uno de estos pasos finalizando con un ejemplo práctico correspondiente a la simulación de un proceso térmico en envases cilíndricos.

3 ELABORACION Y COMPROBACION DEL MODELO MATEMATICO

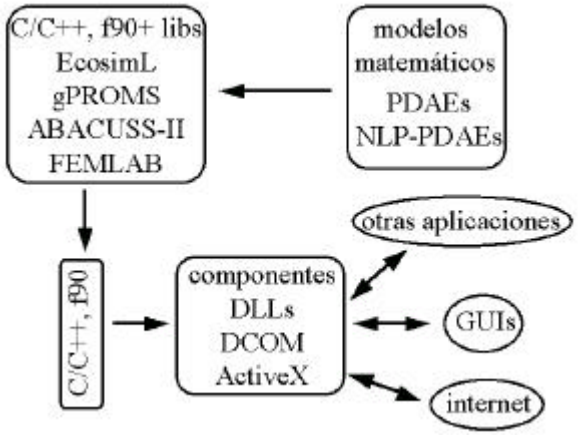
El primer paso en la elaboración del módulo de simulación es el desarrollo del modelo matemático. En esta fase hemos utilizado el entorno de simulación EcosimPro debido principalmente a las facilidades que ofrece en la implementación de sistemas dinámicos, permitiendo incluso el tratamiento de sistemas con eventos discretos.
EcosimPro también permite exportar el código numérico C++ que se crea tras la compilación del componente, pudiendo reutilizarlo en un gran abanico de aplicaciones. De este modo se facilita la creación de interfases gráficas de usuario (GUIs), su integración con otras aplicaciones de cálculo

(optimizadores, controladores, etc.) o incluso de ofimática.

En una segunda fase se utiliza el código C++ generado por EcosimPro para crear una librería dinámica que simule el sistema y que pueda ser llamada desde diferentes entornos.
La creación de esta librería dinámica se hizo de forma compatible con Matlab. Para ello es preciso desarrollar un código fuente que consta de dos partes, una rutina computacional (C++ exportable de EcosimPro) y una rutina pasarela para Matlab ("gateway", se incluye un ejemplo en el apéndice). Esta última permite establecer correctamente el dimensionado y asignación de las variables de entrada y salida del módulo de cálculo. En otras palabras, su principal función es la de comunicar la rutina computacional generada a partir del modelo en EL con Matlab.

La librería dinámica obtenida puede ejecutarse desde otros programas al igual que utilizarse para establecer relaciones cliente/servidor entre diferentes entornos.
La figura 1 ilustra de un modo más intuitivo todo el proceso seguido hasta ahora.

Figura 1: Pasos seguidos en la generación de componentes de los simuladores



La validación de la librería dinámica se realizó comparando su comportamiento al simular el sistema con datos experimentales reales y con los obtenidos mediante otros simuladores.

4 INTERFACE GRAFICO DE USUARIO (GUI) y "WEB-ENABLING"

El término interface de usuario se refiere a los caminos de comunicación entre un programa y sus usuarios, pudiendo ser, como en este caso, de índole gráfico. Incluso la velocidad con la que un programa interactua con el usuario constituye una parte importante del interface.

Su correcta selección garantiza el uso adecuado de las herramientas, así como su éxito. Se encarga de gestionar la entrada/salida de datos además de tareas de pre/postproceso.

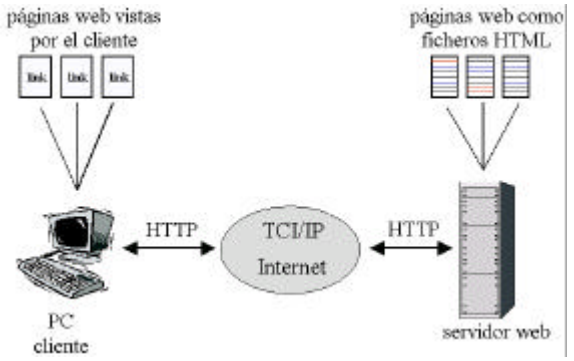
Entre las numerosas posibilidades que existen para desarrollar GUIs que luego sean accesibles vía web, hemos seleccionado un clásico esquema cliente-servidor que consta de los siguientes elementos:

1. Generar formularios de entrada de datos mediante HTML y CGI (Common Gateway Interface)
2. Generar gráficos y tablas de resultados, tras el postproceso, mediante Matlab y HTML dinámico
3. Realizar el "web-enabling" mediante la toolbox Matlab WebServer y un servidor estándar (ej. Apache)

En la etapa final (3) de "web-enabling" se habilita la aplicación para su uso remoto desde cualquier ordenador mediante navegadores estándar y sin necesidad de instalar ningún software en la máquina cliente.

Por otro lado, el Matlab WebServer es relativamente fácil de implementar y mantener y posibilita un acceso sencillo a las enormes capacidades gráficas de postprocesado que tiene Matlab.

Figura 2: Esquema cliente-servidor



5 EJEMPLO: SIMULADOR VÍA WEB DEL PROCESO DE ESTERILIZACIÓN TÉRMICA INDUSTRIAL

El principal propósito es proporcionar a distintos usuarios industriales y de centros tecnológicos una herramienta que permita evaluar, mediante simulación, los efectos en la calidad y la seguridad de los alimentos tratados en un proceso térmico.

5.1 MODELO MATEMATICO

El modelo matemático se ha tomado de Banga y co. [1], siendo:

Ecuación de Fourier y condiciones frontera:

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

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

Cinéticas de destrucción de microorganismos y nutrientes:

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

Cálculo de concentraciones medias finales de microorganismo y nutriente:

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

Valor de la letalidad en el punto crítico F_c y total F_s:

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

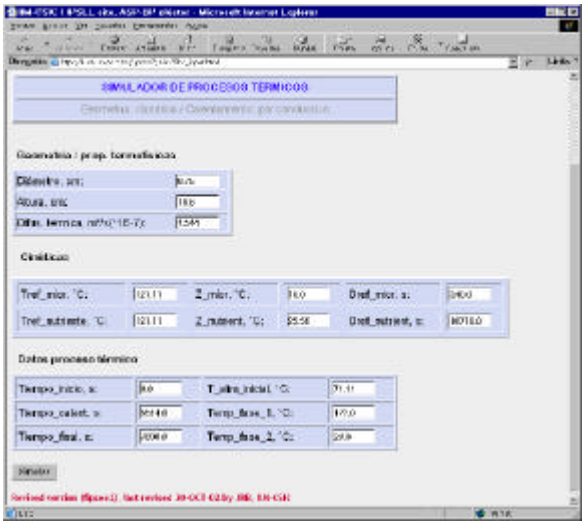
5.2 MODULO DE SIMULACION

El módulo permite, en una primera ventana (ver figura 3) la selección entre simular un proceso, calibrar el modelo, diseñar procesos (perfil temperatura/tiempo) para unas condiciones finales de esterilidad determinadas, y optimizar dicho proceso para asegurar p.ej. una calidad final máxima respetando las restricciones de seguridad microbiológica.

Figura 3: Ventana inicial del módulo de simulación

Tras seleccionar en este caso la simulación del procesamiento térmico, aparece una segunda ventana (ver figura 4) en la cual pueden escogerse diferentes parámetros del sistema (geometría, propiedades termofísicas, parámetros cinéticos, perfil de operación del tratamiento térmico, etc.).

Figura 4: Ventana 2 del módulo de simulación. Introducción de los parámetros del modelo



Finalmente, se obtienen los resultados de la simulación, en forma de tablas (HTML y Excel) y figuras que facilitan su análisis (ver figuras 5 y 6 como ejemplos).

Figura 5: Presentación de resultados

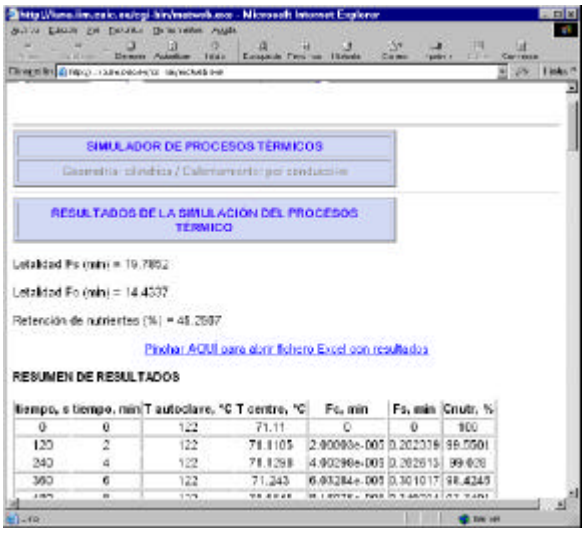
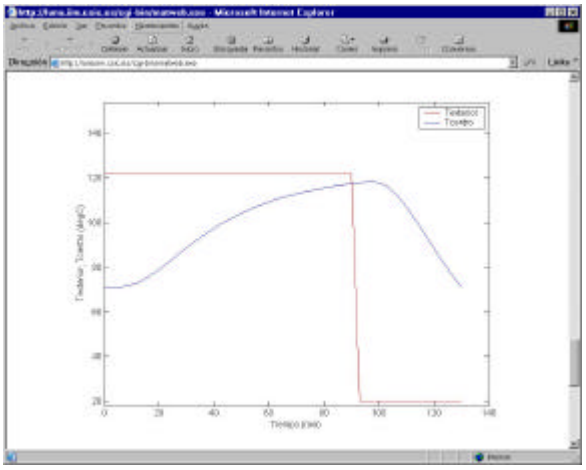


Figura 6: Presentación de resultados (II)



6 CONCLUSIONES

En este trabajo hemos presentado el uso de EcosimPro como herramienta básica para el desarrollo de simuladores dinámicos que puedan ser utilizables de forma remota vía Internet. La combinación del código C++ creado a partir de EcosimPro con Matlab y un servidor web estándar permite desarrollar y mantener de forma sencilla este tipo de aplicaciones.

Referencias

[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), pp. 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.

Apéndice

Este apéndice incluye un ejemplo de la rutina gateway y la definición del experimento con los que se crea la librería dinámica que simula el modelo (además del código C++ exportado desde EcosimPro).

Las variables llamadas xinfo, vx, nvar son variables de entrada a la librería dinámica y cuyo valor se especifica desde Matlab. Estas variables declaran el estado inicial de las variables dinámicas, parámetros de integración, etc., es decir, definen el experimento del modelo.

En cambio las variables fj, yt[1,100], xinfopath[1,100] y xinfomatrix[200,10] son variables de salida de la simulación dinámica que almacenan el resultado de la misma.

· *Definición del experimento:*

```
#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(...) {
            }
}
```

· *Definición de la rutina 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);
```