# TEMA IV: ANÁLISIS DE HERRAMIENTAS DE MODELIZACIÓN Y DESARROLLO

## 1 INTRODUCCIÓN.

## 2 HERRAMIENTAS DE MODELIZACIÓN

## 3. HERRAMIENTAS DE DESARROLLO DE SIMULACIÓN: PROGRAMACIÓN GENERAL

## 4. HERRAMIENTAS ESPECÍFICAS

## INTRODUCCIÓN.

Existe hoy en día una amplia gama de sofwares para la modelización y para la realización de simulación digital, destacando tres grupos fundamentales:

- Lenguajes de Programación
- Lenguajes de Simulación
- Paquetes de Simulación

Los Lenguajes de Programación son de propósito generales tales como el FORTRAN, PASCAL, C, etc..., requieren menos tiempo de ejecución que los lenguajes de simulación, y están disponibles en cualquier computador.

Los Lenguajes de Simulación están diseñados específicamente para los fines de simulación y requieren un menor tiempo de programación. Están especialmente creados para resolver sistemas en tiempo continuo, discreto y eventos discretos, pudiendo algunos simular conjuntamente más de uno de estos tipos de sistemas.

Los Paquetes de Simulación son colecciones de rutinas que pueden ser incluidas en otros programas de uso general. Son en realidad herramientas que dotan de un carácter interactivo al proceso de modelado y simulación de sistemas, siendo empleadas en CAD, CAM y CAE, control de sistemas y diseño de sistemas en Ingeniería.

Entre los tres tipos se va a hacer un mayor énfasis en los lenguajes de simulación. Dentro de esta clase, se pueden distinguir una serie de aspectos que diferencian unos lenguajes de otros, a saber:

- Objetivo de lenguaje
- Código de base del lenguaje
- Grado de documentación
- Mecanismo de avance del tiempo
- Algoritmo de generación de números aleatorios
- Inicialización del programa
- Entrada de datos
- Salida de informes
- Métodos para análisis de datos

Entre los lenguajes de simulación que serán utilizados destacan el ACSL, MATLAB-SIMULINK. DYNAMO, SIMAN y SIMCRIPT, alguno de los cuales será objeto de explicación en temas posteriores.

Por último cabe mencionar a los Sistemas Expertos y Redes Neuronales como herramienta de simulación en el caso de sistemas cuyo comportamiento viene expresado por un conjunto de reglas heurísticas y datos de datos de funcionamiento.

## 2 HERRAMIENTAS DE MODELIZACIÓN

En general son aplicaciones informáticas para el análisis estadístico de un gran volumen de datos. Entre los numerosos programas existentes, cabe destacar dos SPSS y otro de uso libre Programa R.

### FUNDAMENTO Y MANEJO DE SPSS

**Statistical Package for the Social Sciences (SPSS)** es un programa estadístico informático muy usado en las ciencias sociales y las empresas de investigación de mercado. En la actualidad, la sigla se usa tanto para designar el programa estadístico como la empresa que lo produce. Originalmente SPSS fue creado como el acrónimo de *Statistical Package for the Social Sciences* ya que se está popularizando la idea de traducir el acrónimo como "Statistical Product and Service Solutions". Sin embargo, aunque realizando búsquedas por internet estas pueden llevar a la página

web de la empresa, dentro de la página misma de la empresa no se encuentra dicha denominación.

Como programa estadístico es muy popular su uso debido a la capacidad de trabajar con bases de datos de gran tamaño. En la versión 12 es de 2 millones de registros y 250.000 variables. Además, de permitir la recodificación de las variables y registros según las necesidades del usuario. El programa consiste en un módulo base y módulos anexos que se han ido actualizando constantemente con nuevos procedimientos estadísticos. Cada uno de estos módulos se compra por separado.

Actualmente, compite no solo con softwares licenciados como lo son SAS, MatLab, Statistica, Stata, sino también con software de código abierto y libre, de los cuales el más destacado es el Lenguaje R.

Fue creado en 1968 por Norman H. Nie, C. Hadlai (Tex) Hull y Dale H. Bent. Entre 1969 y 1975 la Universidad de Chicago por medio de su *National Opinion Research Center* estuvo a cargo del desarrollo, distribución y venta del programa. A partir de 1975 corresponde a SPSS Inc.

Originalmente el programa fue creado para grandes computadores. En 1970 se publica el primer manual de usuario del SPSS por Nie y Hall. Este manual populariza el programa entre las instituciones de educación superior en EE. UU. En 1984 sale la primera versión para computadores personales.

Desde la versión 14, pero más específicamente desde la versión 15 se ha implantado la posibilidad de hacer uso de las librerías de objetos del SPSS desde diversos lenguajes de programación. Aunque principalmente se ha implementado para Python, también existe la posibilidad de trabajar desde Visual Basic, C++ y otros lenguajes.

El 28 de Junio de 2009 se anuncia que IBM, meses después de ver frustrado su intento de compra de Sun Microsystems, el gigante informático estadounidense anuncia la adquisición de SPSS, por 1.200 millones de dólares.

## Módulos del SPSS

El sistema de módulos de SPSS, como los de otros programas (similar al de algunos lenguajes de programación) provee toda una serie de capacidades adicionales a las existentes en el sistema base. Algunos de los módulos disponibles son:

- **Modelos de Regresión**
- **Modelos Avanzados**
    - **Reducción de datos**: Permite crear variables sintéticas a partir de variables colineales por medio del Análisis Factorial.
    - **Clasificación**: Permite realizar agrupaciones de observaciones o de variables (*cluster analysis*) mediante tres algoritmos distintos.
    - **Pruebas no paramétricas**: Permite realizar distintas pruebas estadísticas especializadas en distribuciones no normales.
- **Tablas**: Permite al usuario dar un formato especial a las salidas de los datos para su uso posterior. Existe una cierta tendencia dentro de los usuarios y de los desarrolladores del software por dejar de lado el sistema original de TABLES para hacer uso más extensivo de las llamadas CUSTOM TABLES.
- **Tendencias**
- **Categorías**: Permite realizar análisis multivariados de variables normalmente categorías. También se pueden usar variables métricas siempre que se realice el proceso de recodificación adecuado de las mismas.
- **Análisis Conjunto**: Permite realizar el análisis de datos recogidos para este tipo especifico de pruebas estadísticas.
- **Mapas**: Permite la representación geográfica de la información contenida en un fichero (descontinuado para SPSS 16).

- **Pruebas Exactas**: permite realizar pruebas estadísticas en muestras pequeñas.
- **Análisis de Valores Perdidos**: Regresión simple basada en imputaciones sobre los valores ausentes.
- **Muestras Complejas**: permite trabajar para la creación de muestras estratificadas, por conglomerados u otros tipos de muestras.
- **SamplePower** (cálculo de tamaños muestrales)
- **Árboles de Clasificación**: Permite formular árboles de clasificación y/o decisión con lo cual se puede identificar la conformación de grupos y predecir la conducta de sus miembros.
- **Validación de Datos**: Permite al usuario realizar revisiones lógicas de la información contenida en un fichero.sav. y obtener reportes de los valores considerados extraños. Es similar al uso de sintaxis o scripts para realizar revisiones de los ficheros. De la misma forma que estos mecanismos es posterior a la digitalización de los datos.
- **SPSS Programmability Extension** (SPSS 14 en adelante). Permite utilizar el lenguaje de programación Python para un mejor control de diversos procesos dentro del programa que hasta ahora eran realizados principalmente mediante scripts (con el lenguaje SAX Basic). Existe también la posibilidad de usar las tecnologías .NET de Microsoft para hacer uso de las librerías del SPSS. Aunque algunos usuarios han cuestionado sobre la necesidad de incluir otros lenguajes, la empresa no tiene esto entre sus objetivos inmediatos.

Desde el SPSS/PC hay una versión adjunta denomina **SPSS Student** que es un programa completo de la versión correspondiente pero limitada en su capacidad en cuanto al número de registros y variables que puede procesar. Esta versión es para fines de enseñanza del manejo del programa.

## LENGUAJE DE PROGRAMACIÓN R

**R** es un lenguaje y entorno de programación para análisis estadístico y gráfico.

Se trata de un proyecto de software libre, resultado de la implementación GNU del premiado lenguaje S. R y S-Plus -versión comercial de S- son, probablemente, los dos lenguajes más utilizados en investigación por la comunidad estadística, siendo además muy populares en el campo de la investigación biomédica, la bioinformática y las matemáticas financieras. A esto contribuye la posibilidad de cargar diferentes librerías o paquetes con finalidades específicas de cálculo o gráfico.

Fue desarrollado inicialmente por Robert Gentleman y Ross Ihaka del Departamento de Estadística de la Universidad de Auckland en 1993.

CARACTERÍSTICAS

R proporciona un amplio abanico de herramientas estadísticas (modelos lineales y no lineales, tests estadísticos, análisis de series temporales, algoritmos de clasificación y agrupamiento, etc.) y gráficas.

Se trata de un lenguaje de programación, lo que permite que los usuarios lo extiendan definiendo sus propias funciones. De hecho, gran parte de las funciones de R están escritas en el mismo R, aunque para algoritmos computacionalmente exigentes es posible desarrollar librerías en C, C++ o Fortran que se cargan dinámicamente. Los usuarios más avanzados pueden también manipular los objetos de R directamente desde código desarrollado en C. R también puede extenderse a través de paquetes desarrollados por su comunidad de usuarios.

R hereda de S su orientación a objetos( como Visual C). Además, R puede integrarse con distintas bases de datos y existen librerías que facilitan su utilización desde lenguajes de programación interpretados como Perl y Python.

Otra de las características de R es su capacidad gráfica, que permite generar gráficos con alta calidad. R posee su propio formato para la documentación basado en LaTeX.

R también puede usarse como herramienta de cálculo numérico, campo en el que puede ser tan eficaz como otras herramientas específicas tales como GNU Octave y su versión comercial, MATLAB.

EXTENSIONES Y PAQUETES

R forma parte de un proyecto colaborativo y abierto. Sus usuarios pueden publicar paquetes que extienden su configuración básica. Existe un repositorio oficial de paquetes cuyo número superó en otoño de 2009 la cifra de los 2000.

Dado el enorme número de nuevos paquetes, éstos se han organizado en vistas (o temas), que permiten agruparlos según su naturaleza y función. Por ejemplo, hay grupos de paquetes relacionados con estadística bayesiana, econometría, series temporales, etc.

Para facilitar el desarrollo de nuevos paquetes, se ha puesto a servicio de la comunidad una forja de desarrollo que facilita las tareas relativas a dicho proceso.

PROYECTOS RELACIONADOS

- Bioconductor, un conjunto de paquetes para el análisis de datos en genómica.
- Rmetrics, orientado al análisis de los mercados financieros y la valoración de instrumentos de inversión.
-

HERRAMIENTAS DE PRODUCTIVIDAD

Existen diversas interfaces que facilitan el trabajo con R.

**INTERFACES GRÁFICAS**

- JGR o *Java GUI for R*, una terminal de R multiplataforma basada en Java

- R Commander (Rcmdr), una interfaz gráfica multiplataforma basada en tcltk
- RExcel, que permite usar R y Rcmdr desde Microsoft Excel
- rggobi, una interfaz a GGobi para visualización
- RKWard, basado en KDE
- Sage
- Statistical Lab
- nexusBPM, una herramienta de automatización
- RWorkBench

**EDITORES E IDES**

De entre los editores de texto e IDEs con soporte para R se cuentan:

Bluefish,[6] Crimson Editor, ConTEXT, Eclipse,[7] Emacs (Emacs Speaks Statistics), Geany, jEdit,[8] Kate,[9] Syn, TextMate, Tinn-R, Vim, gedit, SciTE, WinEdt (R Package RWinEdt) y notepad++.[10]

Sweave es un procesador de documentos que puede ejecutar código de R incrustado en código de LaTeX y para insertar código, resultados y gráficos en el documento escrito en LaTeX. LyX puede usarse para crear y compilar documentos desarrollados en Sweave. El paquete odfWeave es similar, generando documentos en el formato OpenDocument (ODF); extensiones en estado experimental también permiten generar documentos del tipo presentación u hoja de cálculo.

## 3 HERRAMIENTAS DE DESARROLLO DE SIMULACIÓN

PRÁCTICA CON UN SISTEMA CONTINUO

Trataremos de exponer como se realizaría la simulación de un sistema dinámico mediante diversas herramientas, comenzando en C, un

lenguaje de propósito general, hasta finalizar con ACSL o SimuLink, entornos específicos de simulación.

Dado el propósito didáctico, el sistema que se va a simular no será muy complicado. De este modo nos prodremos fijar más en la metodología y olvidar un tanto el complejo desarrollo matemático de este tipo de sistemas. En este caso se ha escogido un sistema de *amortiguación de un coche*. Una vez conocido el sistema, el siguiente paso consiste en modelar este sistema mediante elementos que nos sean conocidos y que seamos capaces de describir matemáticamente.

En esta práctica, el modelo de una amortiguación de un coche se puede expresar como una masa unida a un muelle con un amortiguador en paralelo (ver figura1).

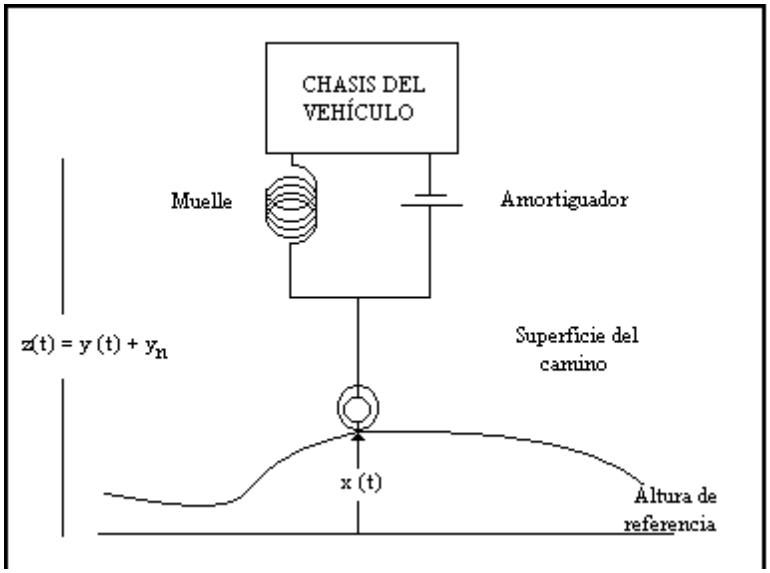



Figura 1:

El objetivo de la simulación será determinar los valores óptimos de las constantes que definen tanto al muelle como al amortiguador, para que la comodidad dentro del coche sea máxima dentro de sus límites de

seguridad necesarios. De esta manera, realizaremos la simulación para un rango de valores de 'K' y 'V', de modo que, observando los resultados obtenidos seamos capaces de diseñar la amortiguación del coche.

Únicamente con nuestra computadora, conseguiremos diseñar un sistema que de un modo tradicional hubiera necesitado complicados y caros mecánismos físicos.

El modelo matemático de este sistema sería el siguiente:

$$\binom{d'.y(t)}{dt'} + \binom{v}{m}_{d(t)}^{dy(t)} + \binom{k}{m}.y.t = \binom{v}{m}_{d(t)}^{dx(t)} + \binom{k}{m}.y.(t)$$

Donde 'm' representa el valor de la masa, 'k', la constante de amortiguador. La función 'x(t)' representa la forma que tendrá la carretera sobre la que avanza el coche. En general será una combinación de una función de baja frecuencia, que será la forma de la carretera, si está en cuesta, etc..., y una función de alta frecuencia, que serán los baches que haya en el camino. En un caso real, se deberían conocer estos datos, normalmente con una función interpolada o mediante valores numéricos. En este artículo, para facilitar el cálculo utilizaremos una rampa, de modo que "x(t) = t". Como condiciones de contorno supondremos que el amortiguador está parado en el instante inicial de tiempos:

$$\frac{dy(0)}{dt} = 0$$
$$y(0) = 0$$

Para realizar nuestro ejemplo, daremos unos valores a las constantes un tanto arbitrarios para facilitar los cálculos. En el caso de una simulación real, deberíamos tomar los valores reales de 'm' 'k' y 'v'. Los valores usados serán:

$m = 0.5$
$k = 1$
$v = 1.5$

luego la ecuación a resolver será:

$$\begin{pmatrix} d'.y(t) \\ dt' \end{pmatrix} + 3\frac{dy(t)}{dt} + 2.y(t) = 3^{dx(t)} + 2y(t)$$

La solución general de esta ecuación con las condiciones de valor inicial indicadas anteriormente será de la forma:

$$y(t) = c_1.e + c_2.e^t + a + b.t$$

Esta información nos será útil para comprobar los resultados de la simulación con el valor real y ver cuan acertado es nuestro método. La forma gráfica de esta solución para un valor de 't' grande será aproximadamente la de una rampa, puesto que el sistema trata de seguir la entrada, que en este caso es 'x(t)'. Si llegamos a variar los valores de las constantes obtendríamos otras soluciones que, en general, seguirán a la rampa pero provocarán oscilaciones que en nuestro sistema darán lugar a pequeños desplazamientos verticales del coche que disminuyen el conforrt de los ocupantes.

En los siguientes puntos se realizará la simulación mediante varias técnicas diferentes que nos permitirán ver los diferentes pros y contras de cada metodología. Para ello realizaremos la simulación en *C*, un lenguaje de propósito general, muy adecuado para ello, en *Matlab*, una herramienta matemática, en *ACSL*, un lenguaje específico de simulación de sistemas y mediante *SimuLink*, una Toolbox de Matlab, que permite simular de una forma casi visual y muy sencilla.

## HERRAMIENTAS DE PROGRAMACIÓN GENERAL: SIMULACIÓN EN C.

Se ha escogido C porque es uno de los lenguajes más extendidos por su enorme versatilidad, que permite al programador todo tipo de programas. Esto a su vez es una desventaja para nuestro caso, que es un problema muy específico. Al no disponer de una herramienta especializada, deberemos resolver la ecuación diferencial numéricamente por nuestra cuenta. Esto es, deberemos implementar nosotros mismos un algoritmo de resolución de ecuaciones diferenciales.

Para ello, se ha utilizado el algoritmo de Eüler, tras reducir la ecuación a un sistema de ecuaciones de primer orden, que es el método más sencillo. Para obtener un resultado mejor podríamos haber utilizado un algoritmo de Runge-Kutta de orden superior.

Por otra parte, de cara a realizar simulaciones más serias utilizando este lenguaje, muy popular en entornos Unix, sería adecuado realizar, o bien obtener, alguna librería de cálculo numérico. De este modo, evitaríamos tener que realizar los algoritmos numéricos de resolución de las ecuaciones diferenciales. Tan sólo tendríamos que utilizar las funciones propias de las librerías y luego enlazar nuestro programa con ellas.

En Internet se pueden encontrar algunas librerías de este tipo distribuidas libremente, aunque las que más abundan son las de Fortran, un lenguaje muy popular en entornos matemáticos y preferidos por algunos profesionales hasta hace muy poco tiempo.

El código de la simulación será el siguiente:

```
#include <stlib.h>
#include <stdio.h>

void main()
{
float t[200],y[200],x[200];
float k,h,c,m;
/* El incremento del tiempo será de 0.5 segundos */
float it=0.05;
int n;
FILE *fichero;

/* Abriremos el fichero de salida*/
fichero=fopen("salida.dat", "w");

/* Introducimos los valores iniciales*/
y[0]=0;
x[0]=0;
t[0]=0;
fprint(fichero,"%f%f\n",t[0],y[0]);

/* Fijamos un valor de 0.5 para h que nos asegura la estabilidad*/
h=0.5;

/*  Bucle de resolución de la ecuación diferencial*/
for (n=0;n<200;n++)
{
      t[n+1]=(n+1)*it;
      /* Resolvemos la ecuación diferencial mediante el método de Eüler*/
      x[n+1]=x[n]+h*(3+2*t[n]-3*x[n]-2*y[n]);
```

```
        y[n+1]=y[n]+h*x[n];
        /* Guardamos los datos en el fichero*/
        fprint(fichero, "%f%f\n", t[]n+1),y[n+1];
    }
    /*Cerramos el fichero de salida*/
    fclose(fichero);
    }
```

Los resultados de salida, se obtienen en un fichero llamado "salida.dat", que generará el programa durante su ejecución. Está formado por dos columnas. La primera contiene los valores del tiempo en segundos y la segunda el valor de 'y(t)', donde esta función representa la distancia desde la carrocería hasta la base de cotas. Además, se podría representar gráficamente, mediante un programa exterior, por ejemplo Excel, la salida obtenida, que tendrá una forma muy similar a la obtenida de una forma teórica.

Otros lenguajes pueden ser, Visual C, Visual Basic o los nuevos lenguajes de Programación en red: JAVA, Visual.NET, Flash etc.

## SIMULACIÓN MEDIANTE MATLAB

El entorno Matlab trae herramientas que nos facilitarán mucho el trabajo frente a la programación tradicional. Matlab, para los que no la conozcan, es un entorno orientado a cálculos habituales en el álgebra matricial, que viene complementado con múltiples *toolboxs*, que son librerías que contienen funciones adicionales que permitirán resolver problemas más específicos, como puede ser el control de sistemas, tratamiento de imágenes o lógica difusa (*fuzzy logic*).

Entre otras ventajas de este entorno, nos encontraremos que contiene funciones que resuelven directamente sistemas de ecuaciones diferenciales. De este modo, podemos obviar el algoritmo de resolución de la ecuación y centrarnos en nuestra simulación. Además, posee una

estructura de programación bastante similar al C, lo que nos permitirá dominarlo con mayor facilidad.

En primer lugar, debemos crear un fichero que describa nuestra ecuación diferencial. Se llamará "*edo.m*" y será de la forma:

```
function xdot=edo(t,x)
xdot(1)=x(2);
xdos(2)=3+2*t-3*x(2)-2*x(1);
```

Seguidamente crearemos otro fichero llamado "*simul.m*" que contendrá el proceso de simulación completo:

```
t0=0;%tiempo inicial de la simulación
tf-10;%tiempo final de la simulación
x0=[0,0];%valores iniciales de la ecuación diferencial
[t,y]=ode45('edo',t0,tf,x());%solución numérica de la ecuación
plot(t,y)
```

La función "*ode45"* devolverá un vector 't' con el tiempo y una matriz 'y' con dos columnas, una en la que aparece el valor de 'y' junto con el valor 'de' la derivada de 'y'.

Otra ventaja que se puede observar en este entorno, es que posee una potente función interna de representación gráfica para los resultados, evitando de este modo la necesidad de utilizar un programa externo como ocurrió con la simulación en C.

## SIMULACIÓN EN ACSL

ACSL, es un metalenguaje creado y diseñado específicamente para la simulación de sistemas continuos. Sus siglas provienen de *Advanced*

*Continuous Simulation Languaje*. Con él podremos simular de una forma sencilla los más completos sistemas físicos. Además, como se apoya en el lenguaje Fortran, podremos crear nuevas funciones y librerías en este lenguaje, en el caso en que necesitemos alguna función que el código no traiga implementada.

En la simulación mediante ACSL tendremos que dividir nuestro trabajo en al menos dos programas. En uno, introduciremos el modelo de nuestro sistema, con sus ecuaciones y en otro, los comandos que se ejecutarán en la simulación. En el caso de utilizar rutinas en Fortran o librerías, deberíamos también incluir estos programas. El interprete de ACSL se creará un programa en Fortran a partir de nuestros programas y lo compilará generando un ejecutable. Nuestro modelo (simul.csl) se describirá como sigue:

```
PROGRAM simul
      DECLARATIVE
      CINTERVAL CINT=0.02
CONSTANT XIC=0.0, XDIC=0.0, TSTP=100.0
TIME INTEG(1.0,0.0)
XDD=3+2*RAMP(0)-3*XD-2*X
XD=INTEG(XDD,XDIC)
TERM(T.GE.TSTP)
END
END
```

Aquí la definición es bastante diferente a la presentada en los métodos anteriores. Este lenguaje posee una potente función llamada *INTEG* que será capaz de integrar la ecuación diferencial dándole un valor inicial. De este modo 'XDD' representa a la derivada segunda y 'XD' la derivada primera.

En segundo lugar definiremos el fichero de comandos (simul.cmd):

```
SET TITTLE=”Simulación de amortiguación de un vehículo”
SPARE
SET TCWPRN=72
OUTPUT TIME,XDD,XD,X,”NCIOUT”=20
PREPAR XDD,XD,TIME,X
START
RANGE “ALL”
PLOT “XAXIS”=TIME
PLOT X,XD
PLOT X,”HI”=1.0,”LO”=0.0,XD
PLOT “XAXIS”=XD,X
APARE
STOP
```

Mediante estos comandos hemos configurado la simulación. Antes de ‘START’ hemos definido el formato que tendrá el fichero de salida (simul.out), donde encontraremos los valores numéricos de la simulación. Tras ‘START’, se le ha indicado al metalenguaje ACSL los gráficos que queremos que realice. En función de la versión de ACSL que utilicemos y si tenemos un posprocesador, estos gráficos serán en modo texto o bien en modo gráfico.

## SIMULACIÓN MEDIANTE SIMULINK

Simulink es una *toolbox* de Matlab, por lo que requiere que esté instalado este programa para funcionar. En ella utilizaremos distintos bloques de entrada/salida para definir nuestro sistema. Dispondremos de distintos generadores de señal, varios tipos de gráficos de salida y bloques que describirán las partes de nuestro sistema.

La aproximación al problema aquí es un tanto diferente. Simulink es una herramiente utilizada habitualmente en el mundo de la Ingeniería de

Control. Por ello, los sistemas se suelen describir de dos formas: o bien mediante ecuaciones de estado, o con funciones de transferencia. Para obtener la función de transferencia de un sistema deberemos tomar transformadas de Laplace en la ecuación diferencial que define el sistema.

Al estar definido nuestro sistema por una única ecuación diferencial, será más fácil tomar transformadas y calcular la función de transferencia. En el caso de un sistema de ecuaciones diferenciales sería preferible, por comodidad, trabajar en variable de estado. De cualquier forma, existen bloques para definir nuestro sistema en cualquiera de los dos modelos.

Transformando nuestra ecuación obtendríamos la siguiente función de transferencia:

$$G(s) = \frac{Y(s)}{X(s)} = \frac{3s+2}{s^2+3.s+2}$$

Donde Y(s) representa la señal de salida y X(s) la señal de entrada. En nuestro caso X(s) será una rampa. En el caso de querer utilizar una señal de entrada más complicada deberemos importarla desde un fichero. La posibilidad de importar ficheros como entrada, nos permitirá utilizar modelos de terrenos mucho más complicados que en el caso de tener que describirlos analíticamente. En nuestro caso, la forma del terreno se encuentra en el fichero "simul.mat", por si se desea utilizar el fichero en lugar de la rampa para realizar el ejercicio.

Una vez que tengamos configurada la entrada de nuestro sistema, utilizaremos un bloque de función de transferencia dentro del menú de sistemas lineales. Por último colocaremos un gráfico autoescalable como salida.

Ya queda solamente comenzar la simulación. En el menú de simulación encontraremos la opción "Start". Dentro de este mismo menú encontraremos la opción "Parameters", desde la cual podremos configurar los parámetros de nuestra simulación. Podremos escoger el periodo temporal de simulación, la precisión y el algoritmo utilizado para resolver las ecuaciones diferenciales.

## CONCLUSIONES

Como hemos podido observar, el proceso de simulación no es muy diferente de un entorno a otro. La gran diferencia aparece en forma de facilidad de resolución de nuestro modelo. Los entornos Matlab y ACSL, contienen sus propias rutinas que facilitan enormemente el trabajo, además de incluir posprocesadores gráficos que permiten representar de una forma fácil y cómoda los resultados numéricos que hayamos obtenido tras la simulación. Sin embargo, la simulación no acaba aquí.

Posteriormente, debemos estudiar estos resultados y alterar las variables del sistema hasta que obtengamos el resultado deseado. En nuestro caso vemos que la salida sigue casi perfecta a la entrada. Sería conveniente, que se cambiaran los valores de las variables 'k' y 'v', para observar como varía la respuesta del sistema con valores diferentes, o cambiar la señal x(t) y ver como respondería nuestra amortiguación a terrenos diferentes, como podría ser un terreno de forma senoidal, mucho más exigente para la amortiguación.

Es fundamental estudiar los resultados tras realizar la simulación numérica.

Resumiendo, podemos observar como el lenguaje C, pese a ser un lenguaje de gran versatilidad, es una solución complicada que obliga al usuario a realizar todas las tareas, algo que el resto de entornos facilita ofreciendo herramientas especializadas.

Sin embargo, Matlab no es el más adecuado para grandes simulaciones donde utilicemos sistemas muy complejos. En ese caso es donde emergen con fuerza las herramientas específicas como ACSL o Dymola, que es un entorno cuasi gráfico que genera una salida en ACSL. Por último, reseñar la gran facilidad de manejo que ofrecen las herramientas gráficas tipo Simulink.

Muchas de las herramientas comentadas hasta ahora son de carácter comercial. Sin embargo, para alguna de ellas podemos encontrar opciones freeware o de libre distribución de igual calidad. Este es el caso de Matlab. Existe un programa clónico distribuido bajo licencia de GNU llamado Octave, que existe en versiones para Linux, FreeBSD y Windows'95.

Este paquete incluye una librería llamada Scicos que es muy similar a Simulink, permitiendo la simulación de sistemas, continuos e incluso híbridos, de un modo gráfico. En el caso de ACSL se pueden encontrar versiones para estudiantes a un precio más económico que el paquete profesional, algo que también con Matlab 5.0 y Simulink 2.0.

## 4. HERRAMIENTAS ESPECÍFICAS EN EDUCAION

El creciente desarrollo de herramientas de bajo costo orientadas a la simulación de procesos y al estudio de los distintos operadores técnicos de control ha hecho posible el que podamos plantear la posibilidad de considerar una serie de programas y herramientas aplicables en el aprendizaje de temas relacionados con la Instrumentación, Adquisición de Datos, Automática, Sistemas de regulación y Control, Autómatas Programables, Sensorica, etc..., en los niveles de EE.MM. y estudios Universitarios.

ACERCA DEL DISEÑO Y SIMULACIÓN.

La simulación en el campo de la automática y la electrónica está resuelta a nivel profesional con una serie de herramientas del tipo de MATLAB, LabVIEW, HP VEE, MATRIXx, etc... pero estas herramientas en algunos casos son costosas y requieren de sólidos conocimientos de matemáticas así como de módulos opcionales de estos grandes programas que también encarecen su implantación en el aula. En las tablas 1 y 2 se pueden ver una selección de herramientas clasificadas en dos grandes niveles. La tabla 1 nos representa una serie de herramientas de fácil manejo y bastante asequibles, ya que si bien existen versiones profesionales de ellas también se ofrecen versiones educativas o de estudiante a bajo precio, del mismo modo que de algunas existen "generosas demos" que bien vale la pena evaluar. En la tabla 2 se muestran las más conocidas y a mi juicio poderosas herramientas orientadas al diseño, simulación y elaboración de prototipos de laboratorio que además son de uso muy extendido en la industria.

| HERRAMIENTAS DE COSTO MEDIO BAJO ORIENTADO A LA ENSEÑANZA | | | |
|---|---|---|---|
| Nombre | Fabricante-distribuidor | Versión/Plataforma | Costo Vers.Educ. |
| WinLabPro | © Graf Electronik Systeme GmbH | V:2.98 Win3.1 | -------- |
| Visual Designer | © Inetelligent Instrumentation | V:3.0 Win3.1/Win95 | -------- |
| DasyLab32 | © Dasytec Daten Sysetem Technik GmbH | V:4.0 Win95/NT | 600 US$ |
| WorkBench | © Strawberry Tree, Inc. Y Dasytec GmbH | V:2 Win3.1 | 800 US$ |
| WinFAC 96 | © Ingenieurbüro Dr. J. Kahlert. | V:96/FT Win3.1/Win32 | 2250 DM |

| | | | |
|---|---|---|---|
| DIAdem | © GFS mbH | V:3.0 Win3.1/Win95 | 2000 DM |
| IAS | © Com Pro-Hard&Software Vertriebs GmbH | V:3.11 Win3.1/Win95 | 682 US$ |
| Snap-Master | © HEM Data Corporation | V:3.1 Win3.1/Win95 | -------- |
| FLOWCHART TRAINER | © Com Tec GmbH | V:3.0 MS-DOS/Win3.1 | 60000ptas. |

Tabla 1

| OTRAS HERRAMIENTAS ORIENTADAS A APLICACIONESPROFESIONALES DE INVESTIGACIÓN O UNIVERSIDADES | | | |
|---|---|---|---|
| Nombre | Fabricante | Versión/Plataforma | Precio Ver.Educ. |
| LabVIEW | © National Instruments | V:5.0 Multiplataforma | Varios Precios |
| LabWindows/CVI | © National Instruments | V:5.0 Multiplataforma | Varios Precios |
| HP VEE | © Hewlett Packard | V:4.0 Multiplataforma | |
| VisSim | © Visual Solutions Inc. | V:3.0 Multiplataforma | . |
| MATRIXx | © Integrated Systems, Inc. | V:6.0 | |
| MATLAB | © The Math Works, Inc. | V:4.0 | Varios Precios |

Tabla 2

La gran mayoría de estas herramientas están concebidas con la misma filosofía. Se organizan mediante un entorno gráfico cuyo núcleo es un editor que permite realizar el cableado o conexionado de distintos bloques funcionales que posteriormente podremos parametrizar a nuestro gusto acercándonos poco a poco el modelo matemático o físico que

deseamos. Poseen además, en su mayoría, una serie de módulos que permiten la conexión con el mundo exterior, convirtiendo de este modo el ordenador en un sistema de adquisición de datos con la posibilidad de crear una seri de instrumentos virtueles y pantallas gráficas que nos permitan un mayor acercamiento a nuestro modelo real.

Con la proliferación de los lenguajes de programación orientados a objetos y los entornos tipo Visual Basic, Delphi o últimamente Java, estas herramientas permiten la integración de bloques de código en nuestra aplicación con lo cual se multiplica su potencia. Algo parecido ocurre con las modernas técnicas de control Fuzzy y Redes Neuronales. Por ejemplo mediante las herramientas WinFAC 96 o DIAdem podemos implementar controladores proporcionales con algoritmos difusos o crear una red neuronal con VisSim, entrenarla y enlazarla mediante el módulo de Adquisición de datos con un proceso como por ejemplo el control de la temperatura de un horno.

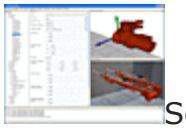
En muchas de estas herramientas, como en el caso de Visual Designer o de DasyLab el usuario haciendo uso de un módulo especial puede diseñar sus propios bloques funcionales tipo Librería e incorporarlos al entorno.

La herramienta IAS es muy apropiada para desarrollar aplicaciones de control mediante PLC o Sistemas de Adquisición de Datos ya que se incorpora la posibilidad de implementar un algoritmo de control haciendo uso de un editor de Organigramas, un editor de esquema funcional, un editor de pantallas de visualización y un potente trazador gráfico.

## HERRAMIENTAS ESPECIFICAS DE USO PROFESIONAL

Son numerosas las herramientas software que el mercado nos ofrece para poder desarrollar estrategias de simulación. A continuación y a modo de ejemplo se muestra una serie de herramientas especificas de simulación.

1º

 Software de modelización de sistema dinámico

MeVEA Modeller

La plataforma de la simulación de MeVEA consta de dos software que se pueden combinar: modelizador y simulador. El software se puede modificar para ser utilizado para probar y para desarrollar casi cualquier clase de máquinas y de componentes mecánicos.

2º

 Software para cálculo de estructura

ANSYS Structural

**3º**

Software de modelización 3D

Edgecam Part Modeler

Modelado rápido para la fabricación.

El Modeledo de la pieza de Edgecam es un 3D que modela la herramienta diseñada específicamente para la construcción y/o la modificación rápidas y simples de modelos sólidos.

4º

Software de simulación de campo electromagnético en 2D

características:

- Elegir a un método de elemento de límite (BEM), al método de elemento finito (FEM) o a disolvente híbrido del método de BE-FE que combina las fuerzas del análisis de BEM y del mercado de cambios
- Enlaces directos para major los paquetes del cad para la representación verdadera de formas geométricas complejas
- Asignar las distribuciones de carga constantes o no uniformes a las superficies
- Simular la conductividad y la permitividad no lineales
- Analizar los campos eléctricos estáticos, la conducción eléctrica y la cuasi-estática de los dieléctricos del lossy
- Exportar los datos a la hoja de balance y otros programas del software para el análisis adicional
- El visualizador resulta con los gráficos, animaciones, y el contorno, la flecha, la línea aerodinámica, o los lugares geométricos del vector traza