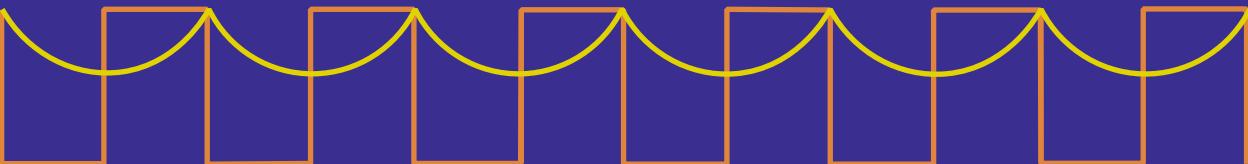


# **UNVirtualLab**

## **Un laboratorio Virtual basado en OpenModelica**

**Oscar G. Duarte V.**







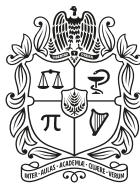
**UNVirtualLab**  
**Un laboratorio virtual basado**  
**en OpenModelica**



# **UNVirtualLab**

## **Un laboratorio virtual basado en OpenModelica**

Oscar G. Duarte V.



UNIVERSIDAD  
**NACIONAL**  
DE COLOMBIA

BOGOTÁ, D. C., 2018

© Universidad Nacional de Colombia - Sede Bogotá  
Facultad de Ingeniería, Departamento de Ingeniería Eléctrica y Electrónica  
© Vicerrectoría de Sede Bogotá  
Dirección de Investigación y Extensión – DIEB  
© Vicerrectoría de Investigación  
Editorial Universidad Nacional de Colombia  
© Oscar G. Duarte V.

Primera edición, septiembre de 2018

ISBN 978-958-783-521-2 (papel)  
ISBN 978-958-783-523-6 (IBD)  
ISBN 978-958-783-522-9 (digital)

Colección Ingenio Propio  
Serie Investigación  
Facultad de Ingeniería

Edición  
Editorial Universidad Nacional de Colombia  
[direditorial@unal.edu.co](mailto:direditorial@unal.edu.co)  
[www.editorial.unal.edu.co](http://www.editorial.unal.edu.co)

Coordinación editorial: María Carolina Suárez S.  
Corrección de estilo: Pablo Castellanos  
Diseño de la colección: Ángela Pilone Herrera  
Diagramación en L<sup>A</sup>T<sub>E</sub>X: Ana Patricia Chávez

Salvo cuando se especifica lo contrario, las figuras y tablas del presente volumen son propiedad del autor

Bogotá, D. C., Colombia, 2018

Prohibida la reproducción total o parcial por cualquier medio sin la autorización escrita del titular de los derechos patrimoniales

Impreso y hecho en Bogotá, D. C., Colombia

---

Catalogación en la publicación Universidad Nacional de Colombia

Duarte Velasco, Oscar Germán-  
UNVirtualLab : un laboratorio virtual basado en OpenModelica / Óscar G. Duarte V. . – Primera edición. – Bogotá : Universidad Nacional de Colombia. Facultad de Ingeniería. Departamento de Ingeniería Eléctrica y Electrónica, Vicerrectoría de Sede. Dirección de Investigación y Extensión (DIEB), Vicerrectoría de Investigación. Editorial, 2018.

1 CD-ROM (554 páginas): ilustraciones (principalmente a color), diagramas, figuras, fotografías. – (Colección Ingenio Propio. Serie Investigación)

Incluye referencias bibliográficas  
ISBN 978-958-783-522-9 (e-book)

1. OpenModelica (Programa para computador) 2. Software libre y de código abierto 3. Simulación por computadores – Educación 4. Simulación (Matemáticas) – Métodos 5. Desarrollo de programas para computador I. Título II. Serie

# CONTENIDO

<b>INTRODUCCIÓN</b>	30
<b>I MODELADO Y SIMULACIÓN CON OPENMODELICA</b>	39
<b>1 MODELADO DE SISTEMAS DINÁMICOS</b>	41
1.1 Sistemas, modelos, simulación	43
1.1.1 Construcción de modelos matemáticos	44
1.2 Técnicas de modelado matemático con fines de simulación	46
1.2.1 Técnicas analíticas específicas de cada dominio	46
1.2.2 Técnicas de espacio de estado	53
1.2.3 Técnicas variacionales	56
1.2.4 Diagramas de bloques y diagramas de flujo de señal	60
1.2.5 Grafos de enlaces de potencia	64
1.2.6 Modelado orientado a objetos	69
1.3 Comparación de las técnicas de modelado	75
1.4 El modelado para un laboratorio virtual	80
<b>2 SIMULACIÓN NUMÉRICA DE ECUACIONES DIFERENCIALES ALGEBRAICAS</b>	83
2.1 Análisis de DAEs	85
2.1.1 Condiciones iniciales	86
2.1.2 Índices	89
2.2 El proceso de simulación	91
2.3 Cálculo de condiciones iniciales	94
2.4 Algoritmos de tratamiento simbólico de DAEs	96
2.4.1 Ordenamiento y asignación de causalidad	96
2.4.2 El algoritmo de Tarjan	99
2.4.3 Algoritmos de rasgadura	107
2.4.4 Algoritmo de relajación	110
2.4.5 El algoritmo de Pantelides para la reducción del índice	112
2.5 Simulación de eventos discretos	114
2.6 Selección del método numérico	116
2.6.1 Algunas consideraciones sobre los métodos de simulación	123

<b>3 EL LENGUAJE MODELICA</b>	<b>127</b>
3.1 Características	132
3.1.1 Modelo de componentes y conectores	132
3.1.2 Orientación	135
3.1.3 Multidominio	140
3.1.4 Acausalidad	142
3.1.5 Modelo del tiempo	147
3.1.6 Interoperabilidad con lenguajes de programación	149
3.1.7 Documentación de los modelos	151
3.1.8 Reusabilidad	152
3.2 Algunos elementos del lenguaje	154
3.3 La Modelica Standard Library	156
<b>4 LA SUITE OPENMODELICA</b>	<b>163</b>
4.1 Arquitectura	165
4.2 La consola OMShell	167
4.3 El proceso de compilación	171
4.4 El compilador omc	175
4.4.1 Estructura interna del compilador	175
4.4.2 Uso del compilador	177
4.5 Ejecución del modelo compilado	179
4.5.1 Archivo de entrada	180
4.5.2 Archivo de resultados	181
4.6 El editor de modelos y documentos OMNotebook	183
4.7 El editor gráfico OmEdit	184
4.8 El <i>plugin</i> para MDT Eclipse	188
4.9 Otros módulos y desarrollos	189
<b>II UNVIRTUALLAB</b>	<b>191</b>
<b>5 LABORATORIOS VIRTUALES. ESTADO DEL ARTE</b>	<b>193</b>
5.1 Características de los laboratorios virtuales	195
5.1.1 Orientación a la web	195
5.1.2 Naturaleza de los modelos	197
5.1.3 Dominio de los experimentos	198
5.1.4 Interactividad	198
5.1.5 Función del laboratorio	199
5.1.6 Disponibilidad de los modelos	200
5.1.7 Integración con LMS y compatibilidad SCORM	200

5.1.8	Licenciamiento y acceso	201
5.2	Clasificación y ejemplos	201
5.3	Laboratorios y aprendizaje de la ingeniería	203
5.3.1	Importancia de la experimentación	204
5.3.2	Importancia del modelado y de la simulación	205
5.3.3	La experimentación como estrategia de aprendizaje	207
5.4	Consideraciones en el diseño de laboratorios virtuales	210
<b>6</b>	<b>UNVIRTUALLAB. ARQUITECTURA</b>	215
6.1	Especificaciones	217
6.1.1	Características	217
6.1.2	Concepto de diseño	218
6.1.3	Especificaciones funcionales	222
6.1.4	Casos de uso	225
6.2	Modelo Entidad-Relación	228
6.3	Componentes	230
6.4	Interfaz	234
6.4.1	Estructura de páginas	234
6.5	Implementación en php	239
6.5.1	Clases enfocadas en el experimentador	239
6.5.2	Clases enfocadas en el administrador	244
6.6	Configuración de la interfaz	250
6.6.1	Internacionalización - I18n	250
6.6.2	Apariencia	251
6.6.3	Estilo de las figuras	254
<b>7</b>	<b>MANUALES</b>	257
7.1	Manual del administrador	259
7.1.1	Instalación del servidor	259
7.1.2	Copias de seguridad	268
7.1.3	Exportación e importación de modelos	268
7.2	Manual del modelador	270
7.2.1	Objetivos del OVA	271
7.2.2	Especificaciones funcionales	272
7.2.3	Modelo	272
7.2.4	Archivos comprimidos	274
7.2.5	Plantas de experimentación	276
7.2.6	Animaciones	282
7.2.7	Experimentos sugeridos	290

7.2.8 Documentación	292
7.2.9 Ayudas para la documentación	294
7.3 Manual del experimentador	296
<b>III PLANTAS EXPERIMENTALES, MODELOS Y EXPERIMENTOS</b>	<b>301</b>
<b>8 MOTOR ELÉCTRICO DE CORRIENTE CONTINUA E IMANES PERMANENTES</b>	<b>303</b>
8.1 El modelo	304
8.1.1 Principios físicos	304
8.1.2 Forma constructiva y principio de operación	305
8.1.3 Modelo matemático	309
8.2 Plantas de experimentación y experimentos sugeridos	312
8.3 La implementación	316
8.3.1 Listado de archivos	317
<b>9 CALENTAMIENTO DE CABLES AÉREOS</b>	<b>319</b>
9.1 El modelo	320
9.1.1 Modelo térmico estático	320
9.1.2 Modelo térmico dinámico	321
9.1.3 Modelo mecánico	323
9.2 Plantas de experimentación y experimentos sugeridos	327
9.3 La implementación	340
9.3.1 Modelo térmico	341
9.3.2 Modelos mecánico y geométrico	342
9.3.3 Listado de archivos	344
<b>10 EVOLUCIÓN DE ENFERMEDADES</b>	<b>363</b>
10.1 El modelo	364
10.1.1 Modelo SIR	364
10.1.2 Modelo SIR con nacimientos y muertes	366
10.1.3 Modelo SIS	366
10.1.4 Modelo SIR-SI	367
10.1.5 Modelo SIR-SI con dos serotipos	368
10.1.6 Modelo del dengue	370
10.2 Plantas de experimentación y experimentos sugeridos	373
10.3 La implementación	385
10.3.1 Ejemplo: implementación del modelo SIR	387
10.3.2 Listado de archivos	387

<b>11 TRÁNSITO DE ESTUDIANTES A TRAVÉS DE UN PLAN DE ESTUDIOS</b>	395
11.1 El modelo	396
11.1.1 Modelo reducido	402
11.1.2 Análisis del modelo reducido	402
11.1.3 Evolución del número de estudiantes	403
11.1.4 Número esperado de estudiantes	405
11.1.5 Efecto de los parámetros sobre el número esperado de estudiantes	405
11.1.6 Tiempos de graduación	408
11.1.7 Efecto de los parámetros sobre los tiempos de graduación	411
11.2 Plantas de experimentación y experimentos sugeridos	413
11.3 La implementación	419
11.3.1 Listado de archivos	420
<b>12 CENTRAL DE GENERACIÓN HIDROELÉCTRICA</b>	423
12.1 El modelo	424
12.1.1 Estructura de una central hidroeléctrica y transformaciones de energía	424
12.1.2 El embalse	425
12.1.3 El ducto	427
12.1.4 La turbina	427
12.1.5 El generador	428
12.1.6 Los lazos de control	431
12.1.7 La carga eléctrica	433
12.2 Plantas de experimentación y experimentos sugeridos	435
12.3 La implementación	439
12.3.1 Listado de archivos	442
<b>13 DINÁMICA DE SISTEMAS. MODELO DEL MUNDO</b>	445
13.1 El modelo	446
13.1.1 Dinámica de sistemas	446
13.1.2 Modelo del mundo	449
13.2 Plantas de experimentación y experimentos sugeridos	456
13.3 La implementación	461
13.3.1 Listado de archivos	463
<b>14 IDENTIFICACIÓN ALGEBRAICA. MASA DESLIZANTE CON FRICCIÓN</b>	467
14.1 El modelo	468
14.1.1 Primera integración	468

14.1.2 Segunda integración	469
14.1.3 Realizaciones	471
14.2 Plantas de experimentación y experimentos sugeridos	472
14.3 La implementación	476
14.3.1 Listado de archivos	477
<b>A LICENCIAMIENTO</b>	483
A.1 Traducción no oficial	490
<b>B ESTRUCTURA DE TABLAS EN LA BASE DE DATOS</b>	499
<b>C VALORES DE LAS PROPIEDADES DE LAS CLASES DE ADMINISTRACIÓN</b>	505
<b>D PARÁMETROS DEL TEMA unvlabasic</b>	517
<b>E VERSIÓN 1.0</b>	533
<b>BIBLIOGRAFÍA</b>	535
<b>LISTA DE ENLACES</b>	551

## LISTA DE FIGURAS

Figura 1.1	Círculo de los ejemplos de modelado mediante diversas técnicas	47
Figura 1.2	Círculo de los ejemplos 1.5 y 1.7	63
Figura 1.3	Elementos de un grafo de enlace de potencia	66
Figura 2.1	Estructura del método de simulación de Euler hacia adelante	92
Figura 2.2	Estructura de un simulador numérico para DAEs	93
Figura 2.3	Aplicación del algoritmo de Tarjan al ejemplo 2.6, visualizado con grafos	104
Figura 2.4	Aplicación del algoritmo de Tarjan al ejemplo 2.6, visualizado matricialmente	105
Figura 2.5	Interacción entre variables de sistemas híbridos	115
Figura 2.6	Diagrama de flujo básico de un simulador híbrido	116
Figura 2.7	Simulación de algunos modelos de la parte III	123
Figura 3.1	Diagrama y resultados de simulación para el modelo del ejemplo 3.1	130
Figura 3.2	Conexión de dos componentes Modelica	132
Figura 3.3	Esquema de conexión de varios componentes Modelica	133
Figura 4.1	Interacción de los módulos de la <i>suite</i> OpenModelica	166
Figura 4.2	Captura de pantalla de la consola OMShell	167
Figura 4.3	Proceso de compilación, ejecución y visualización con OpenModelica	172
Figura 4.4	Interacción de algunos archivos en una simulación de OpenModelica	179
Figura 4.5	Apariencia de OMNotebook con una implementación del ejemplo 4.2	184
Figura 4.6	Ventanas de OMEdit	185
Figura 4.7	Apariencia de OMEdit con una implementación del ejemplo 4.4	187

Figura 4.8	Apariencia de eclipse con una implementación del modelo del ejemplo 4.2	189
Figura 5.1	Orientación de laboratorios virtuales	196
Figura 5.2	Modelo del aprendizaje de Kolb	212
Figura 6.1	UNVirtualLab como herramienta de aprendizaje	219
Figura 6.2	Archivos en una simulación de UNVirtualLab	220
Figura 6.3	Diagrama UML de actividades de una simulación en UNVirtualLab	221
Figura 6.4	Diagrama UML de casos de uso para el experimentador	226
Figura 6.5	Diagrama UML de casos de uso para el administrador	226
Figura 6.6	Diagrama UML de casos de uso para el administrador. Edición de experimentos	227
Figura 6.7	Diagrama Entidad-Relación de la base de datos	228
Figura 6.8	Módulos para el experimentador	231
Figura 6.9	Módulos para el experimentador avanzado	232
Figura 6.10	Módulos del servidor mínimo	233
Figura 6.11	Módulos del servidor con compilación	233
Figura 6.12	Diagrama UML de implementación. Módulos para el modelador	234
Figura 6.13	Página <i>about</i> . Estructura de bloques y código XML asociado	235
Figura 6.14	Estructura de la interfaz de experimentación	236
Figura 6.15	Estructura de las páginas de administración	238
Figura 6.16	Estructura de la interfaz para embeber el modelo y la documentación	239
Figura 6.17	Diagrama UML de clases	240
Figura 6.18	Diagrama UML de clases de administración	245
Figura 6.19	Diagrama UML de clases de manejo de sesiones	246
Figura 6.20	Anidación CSS	252
Figura 6.21	Anidación CSS para admimistración	252
Figura 6.22	Bloques <i>div</i> principales	253
Figura 6.23	Bloques <i>div</i> principales de administración	253
Figura 6.24	Regiones de las figuras asociadas a un modelo	255
Figura 7.1	Proceso de instalación de UNVirtualLab	259
Figura 7.2	Pantalla de UNVirtualLab para el usuario experimentador	263

Figura 7.3	Pantalla de UNVirtualLab para inicio de sesión de administrador	264
Figura 7.4	Pantalla de UNVirtualLab para el usuario administrador con sesión abierta	264
Figura 7.5	Pantalla de UNVirtualLab, ejemplo de Motor DC	268
Figura 7.6	Proceso iterativo de modelado	270
Figura 7.7	Un ciclo de modelado	271
Figura 7.8	Instrumentos asociados a un experimento	276
Figura 7.9	Gráfico SVG para la animación del ejemplo	290
Figura 8.1	Rotor de un motor DC de imanes permanentes.	303
Figura 8.2	Esquema de una forma constructiva de un motor DC de imanes permanentes	305
Figura 8.3	Líneas de campo magnético en un motor DC de imanes permanentes	306
Figura 8.4	Geometría de una espira	307
Figura 8.5	Fuerzas y torques en una espira	308
Figura 8.6	Fuerza electromotriz inducida en una espira	309
Figura 8.7	Diagrama de un motor DC de imanes permanentes acoplado a una carga mecánica	310
Figura 9.1	Línea de transmisión	319
Figura 9.2	Geometría de la catenaria	324
Figura 9.3	Modelo <i>Catenary</i>	343
Figura 10.1	<i>Aedes aegypti</i>	363
Figura 10.2	Diagrama del modelo SIR	366
Figura 10.3	Diagrama del modelo SIR con nacimientos y muertes	366
Figura 10.4	Diagrama del modelo SI	367
Figura 10.5	Diagrama del modelo SIR-SI	367
Figura 10.6	Diagrama del modelo SIR-SI con dos serotipos de virus	370
Figura 10.8	Diagrama del dengue	372
Figura 11.1	Ceremonia de graduación de ingenieros. Año 1964	395
Figura 11.2	Plan de estudios del modelo mínimo	396
Figura 11.3	Diagrama de flujo para el estado $x_j(k)$	397
Figura 11.4	Respuesta al escalón del modelo. $\alpha = 0,85$ $\beta = 0,03$ $\sigma = 0,01$	404

Figura 11.5	Efecto de la variación de los parámetros sobre el número de estudiantes	407
Figura 11.6	Superficie de población de referencia	408
Figura 11.7	Respuesta al impulso del modelo. $\alpha = 0,85$ $\beta = 0,03$ $\sigma = 0,01$	410
Figura 11.8	Efecto de la variación de los parámetros sobre el tiempo de graduación	412
Figura 12.1	Central hidroeléctrica de Cheoah	423
Figura 12.2	Diagrama esquemático de una central hidroeléctrica	425
Figura 12.3	Casa de máquinas y rotor de la central Hoover	429
Figura 12.4	Diagrama de una máquina sincrónica	430
Figura 12.5	Sistema trifásico de tensiones inducidas en el circuito de armadura	430
Figura 12.6	Círculo equivalente de una máquina sincrónica	432
Figura 12.7	Estrategia de control de generación	433
Figura 12.8	Implementación del modelo de una central hidroeléctrica	440
Figura 12.9	Curva de demanda de potencia	441
Figura 13.1	<i>Model of the World and Gdańsk society</i> de Anton Möller	445
Figura 13.2	Símbolos empleados en un diagrama de estructuras	448
Figura 13.3	Dinámica general de la población	450
Figura 13.4	Dinámica general del capital	451
Figura 14.1	Masa deslizante	467
Figura 14.2	Diagrama de bloques del proceso de estimación básico	471
Figura 14.3	Diagrama de bloques del proceso de estimación con filtrado de señal	471
Figura 14.4	Esquema de la implementación	479

## LISTA DE TABLAS

Tabla 1.1	Variables del modelo del ejemplo 1.7	77
Tabla 1.2	Parámetros del modelo del ejemplo 1.7	78
Tabla 2.1	Numeración de ecuaciones y variables en el ejemplo 2.6	102
Tabla 2.2	Ecuaciones y variables del ejemplo 2.6, una vez ordenadas con el algoritmo de Tarjan	103
Tabla 2.3	Ecuaciones de asignación en el ejemplo 2.6	106
Tabla 2.4	Modelos y variables simulados en el ejemplo 2.11	119
Tabla 2.5	Comparación experimental del desempeño de algunos métodos de simulación numérica	120
Tabla 2.6	Comparación experimental del desempeño del método <code>dassl</code>	124
Tabla 2.7	Comparación experimental del desempeño del método <code>euler</code>	125
Tabla 2.8	Comparación experimental del desempeño del método <code>rungekutta</code>	125
Tabla 2.9	Comparación experimental del desempeño del método <code>inline-rungekutta</code>	126
Tabla 2.10	Comparación experimental del desempeño del método <code>dasslwort</code>	126
Tabla 3.1	Paquetes y subpaquetes de la Modelica Standard Library	157
Tabla 4.1	Comandos disponibles más usuales en una sesión interactiva	168
Tabla 4.2	Principales argumentos del comando <code>simulate</code>	169
Tabla 4.3	Métodos de integración disponibles en <code>omc</code>	173
Tabla 5.1	Subtópicos CDIO relacionados directamente con habilidades de experimentación	205
Tabla 5.2	Listado de habilidades CDIO relacionadas con el modelado y la simulación	206

Tabla 5.3	Características de diseño de un ambiente virtual de aprendizaje	210
Tabla 5.4	Niveles de autonomía y tipos de actividad de laboratorio	213
Tabla 6.1	Características de los módulos de la versión 2.0 de UNVirtualLab	232
Tabla 6.2	Funcionalidad de los bloques de UNVirtualLab	237
Tabla 6.3	Clases orientadas al experimentador	242
Tabla 6.3	Clases orientadas al experimentador	243
Tabla 6.3	Clases orientadas al experimentador	244
Tabla 6.4	Tipo de datos válidos para la clase <code>adminblock</code>	247
Tabla 6.5	Clases orientadas al administrador	249
Tabla 7.1	VARIABLES DE CONFIGURACIÓN DE UNVIRTUALLAB EN EL ARCHIVO <code>config/unvlconfig.txt</code>	267
Tabla 7.2	PROPIEDADES DE LOS CONTROLES DE CADA EXPERIMENTO	276
Tabla 7.3	PROPIEDADES DE LOS CONTROLES EN EL EJEMPLO	280
Tabla 7.4	EJEMPLO DE IMPLEMENTACIÓN DE ALGUNOS TIPOS DE ANIMACIÓN SVG	283
Tabla 7.5	USOS Y CONSIDERACIONES DE DISEÑO PARA ACTIVIDADES DE LABORATORIO	291
Tabla 7.6	EXPERIMENTOS SUGERIDOS PARA EL EJEMPLO DE LA SECCIÓN 7.2	293
Tabla 7.7	LISTADO DE PREGUNTAS FRECUENTES RELATIVAS A UNVIRTUALLAB	296
Tabla 8.1	PARÁMETROS DEL EXPERIMENTO 1	314
Tabla 8.2	FIGURAS DEL EXPERIMENTO 1	315
Tabla 8.3	ANIMACIONES DEL EXPERIMENTO 1	316
Tabla 8.4	VARIABLES DE RESULTADOS DEL EXPERIMENTO 1	316
Tabla 8.5	ARCHIVOS DEL MODELO	317
Tabla 9.1	PARÁMETROS DEL EXPERIMENTO 2	329
Tabla 9.2	FIGURAS DEL EXPERIMENTO 2	330
Tabla 9.3	VARIABLES DE RESULTADOS DEL EXPERIMENTO 2	330
Tabla 9.4	PARÁMETROS DEL EXPERIMENTO 3	333
Tabla 9.5	FIGURAS DEL EXPERIMENTO 3	333
Tabla 9.6	VARIABLES DE RESULTADOS DEL EXPERIMENTO 3	333
Tabla 9.7	PARÁMETROS DEL EXPERIMENTO 4	336
Tabla 9.8	FIGURAS DEL EXPERIMENTO 4	336
Tabla 9.9	VARIABLES DE RESULTADOS DEL EXPERIMENTO 4	337

Tabla 9.10	Parámetros del experimento 5	339
Tabla 9.11	Figuras del experimento 5	340
Tabla 9.12	Variables de resultados del experimento 5	340
Tabla 9.13	Parámetros en el registro ConductorData	340
Tabla 9.14	Parámetros en el registro SpanData	341
Tabla 9.15	Parámetros en el registro TimeData	341
Tabla 9.16	Funciones para el modelo térmico	341
Tabla 9.17	Funciones para los modelos mecánico y geométrico	342
Tabla 9.18	Parámetros para la implementación por defecto	344
Tabla 9.19	Archivos del modelo	344
Tabla 10.1	Parámetros del experimento 6	374
Tabla 10.2	Figuras del experimento 6	375
Tabla 10.3	Variables de resultados del experimento 6	376
Tabla 10.4	Parámetros del experimento 7	377
Tabla 10.5	Figuras del experimento 7	377
Tabla 10.6	Variables de resultados del experimento 7	378
Tabla 10.7	Parámetros del experimento 8	380
Tabla 10.8	Figuras del experimento 8	380
Tabla 10.9	Variables de resultados del experimento 8	381
Tabla 10.10	Parámetros del experimento 9	383
Tabla 10.11	Figuras del experimento 9	384
Tabla 10.12	Variables de resultados del experimento 9	385
Tabla 10.13	Archivos del modelo	387
Tabla 11.1	Parámetros del experimento 10	414
Tabla 11.2	Figuras del experimento 10	415
Tabla 11.3	Variables de resultados del experimento 10	416
Tabla 11.4	Parámetros del experimento 11	418
Tabla 11.5	Figuras del experimento 11	418
Tabla 11.6	Variables de resultados del experimento 11	419
Tabla 11.7	Archivos del modelo	420
Tabla 12.1	Parámetros del experimento 12	437
Tabla 12.2	Figuras del experimento 12	438
Tabla 12.3	Animaciones del experimento 12	439
Tabla 12.4	Variables de resultados del experimento 12	439
Tabla 12.5	Archivos del modelo	442

Tabla 13.1	Parámetros del experimento 13	458
Tabla 13.2	Figuras del experimento 13	459
Tabla 13.3	Variables de resultados del experimento 13	460
Tabla 13.4	Obtención de los escenarios de Meadows	462
Tabla 13.5	Archivos del modelo	463
Tabla 14.1	Parámetros del experimento 14	474
Tabla 14.2	Figuras del experimento 14	475
Tabla 14.3	Variables de resultados del experimento 14	475
Tabla 14.4	Archivos del modelo	477
Tabla B.1	Tablas y campos en la base de datos	499
Tabla C.1	Contenido de la propiedad table para las clases de administración	506
Tabla C.2	Contenido del arreglo Fields para las clases de administración	506
Tabla C.3	Contenido del arreglo Relations1N para las clases de administración	511
Tabla C.4	Contenido del arreglo Relations1N1N para las clases de administración	513
Tabla C.5	Contenido del arreglo Relations1NN1 para las clases de administración	515
Tabla D.1	Clases y parámetros CSS. Theme unvlabasic. Archivo style.css	518
Tabla D.2	Valores de los parámetros CSS en el theme unvlabasic en el archivo style.css	522
Tabla D.3	Clases y parámetros CSS. Theme unvlabasic. Archivo styleSVG.css	531
Tabla D.4	Valores de los parámetros CSS en el theme unvlabasic en el archivo styleSVG.css	532

## LISTA DE EJEMPLOS

Ejemplo 1.1	Modelado analítico con técnicas específicas de un dominio	49
Ejemplo 1.2	Modelado para simulación con técnicas de un dominio	51
Ejemplo 1.3	Modelado en espacio de estado	54
Ejemplo 1.4	Modelado basado en el lagrangiano	58
Ejemplo 1.5	Modelado con diagramas de bloques	61
Ejemplo 1.6	Modelado con grafos de enlace de potencia	67
Ejemplo 1.7	Modelado orientado a objetos	72
Ejemplo 2.1	Restricciones ocultas	87
Ejemplo 2.2	Índice de diferenciación	89
Ejemplo 2.3	Método de Euler hacia adelante	92
Ejemplo 2.4	Condiciones iniciales consistentes	95
Ejemplo 2.5	Matriz de incidencia	98
Ejemplo 2.6	Algoritmo de Tarjan	101
Ejemplo 2.7	Algoritmo de rasgadura	108
Ejemplo 2.8	Algoritmo de relajación	111
Ejemplo 2.9	Algoritmo de Pantelides	113
Ejemplo 2.10	Ecuaciones <i>stiff</i>	118
Ejemplo 2.11	Desempeño de los métodos numéricos de simulación	118
Ejemplo 3.1	Simulación de un modelo Modelica	130
Ejemplo 3.2	Ejemplo mínimo	131
Ejemplo 3.3	Conexión de componentes	133
Ejemplo 3.4	Conectores	134
Ejemplo 3.5	Clases e instancias	136
Ejemplo 3.6	Jerarquía de clases	139
Ejemplo 3.7	Multidominio	141
Ejemplo 3.8	Acausalidad	143
Ejemplo 3.9	Causalidad	145
Ejemplo 3.10	Modelo discreto	147
Ejemplo 3.11	Modelo híbrido	148
Ejemplo 3.12	Funciones externas	150
Ejemplo 3.13	Documentación textual y gráfica	152
Ejemplo 3.14	Librerías	153
Ejemplo 4.1	Interpretación de comandos Modelica	168

Ejemplo 4.2	Definición en línea de modelos Modelica	169
Ejemplo 4.3	Uso de librerías y modelos externos	171
Ejemplo 4.4	Traducción de código Modelica	173
Ejemplo 4.5	Generación de código C a partir de un archivo simple	177
Ejemplo 4.6	Generación de código C a partir de un archivo que usa la librería Modelica	177
Ejemplo 4.7	Instrucciones Modelica	178
Ejemplo 4.8	Archivo de entrada	180
Ejemplo 4.9	Archivo de resultados en formato csv	182
Ejemplo 4.10	Archivo de resultados en formato plt	182
Ejemplo 4.11	OMNotebook	183
Ejemplo 4.12	OMEedit	186
Ejemplo 4.13	El <i>plugin</i> MDT para Eclipse	188
Ejemplo 7.1	Objetivos de aprendizaje	271
Ejemplo 7.2	Especificaciones funcionales	272
Ejemplo 7.3	Modelado	273
Ejemplo 7.4	Preparación de archivos	275
Ejemplo 7.5	Planta de experimentación	279
Ejemplo 7.6	Animación 2D	289
Ejemplo 7.7	Experimentos sugeridos	292
Ejemplo 7.8	Documentación	294
Ejemplo 7.9	Ayuda para la generación de documentación	296

## LISTA DE ARCHIVOS

1.1	LLC/LLC.mo	73
1.2	LLC/LLCRextends.mo	75
2.1	rc.mo	101
3.1	primerOrden.mo	131
3.2	rc.mo	134
3.3	LossyGearDemo1.mo	138
3.4	bloquesMath.mo	140
3.5	duffing1.mo	144
3.6	duffing2.mo	144
3.7	duffing3.mo	145
3.8	gTest1.mo	146
3.9	gTest2.mo	146
3.10	discreto.mo	148
3.11	hibrido.mo	149
3.12	selector.mo	150
3.13	extSelector.c	151
3.14	rcOMedit.mo	152
3.15	osciladores/package.mo	154
3.16	oscTest.mo	154
4.1	rc.mo	174
4.2	rc.c	175
4.3	test.mo	177
4.4	rc.mo	178
4.5	rc.mos	178
7.1	DCmotor.mo	274
7.2	figs/ejeSvg/ejemploBase.svg	284
7.3	figs/ejeSvg/ejemploMueveX.svg	284
7.4	figs/ejeSvg/ejemploMueveY.svg	285
7.5	figs/ejeSvg/ejemploMotion.svg	285
7.6	figs/ejeSvg/ejemploHeight.svg	286
7.7	figs/ejeSvg/ejemploWidth.svg	286
7.8	figs/ejeSvg/ejemploRota.svg	287

7.9	figs/ejeSvg/ejemploRotaPunto.svg	288
7.10	figs/ejeSvg/ejemploColor.svg	288
7.11	figs/ejeSvg/ejemploTextoBase.svg	289
7.12	figs/ejeSvg/ejemploTexto.svg	289
8.1	DCmotor.mo	317
9.1	AirConductivity.mo	345
9.2	AirDensity.mo	345
9.3	AirViscosity.mo	346
9.4	AngleFactor.mo	346
9.5	Catenary.mo	347
9.6	CatenaryLenght.mo	348
9.7	CatenarySa.mo	348
9.8	CatenarySag.mo	348
9.9	CatenaryStateChange.mo	349
9.10	CatenaryX.mo	349
9.11	CatenaryXbar.mo	349
9.12	Conductor.mo	349
9.13	ConductorData.mo	350
9.14	ConvectionFlow.mo	350
9.15	ConvectionHeatFlow.mo	351
9.16	Curve.mo	351
9.17	DynamicCurve.mo	352
9.18	ElectricalCatenary.mo	353
9.19	FilmTemperature.mo	353
9.20	ForcedConvection.mo	353
9.21	ForcedConvectionHigh.mo	354
9.22	ForcedConvectionLow.mo	355
9.23	MyElectricalLine.mo	355
9.24	MyStandAloneLine.mo	356
9.25	NaturalConvection.mo	356
9.26	Rating.mo	357
9.27	SagAnalysis.mo	358
9.28	SolarAltitude.mo	358
9.29	SolarAzimuth.mo	359
9.30	SolarFlux.mo	360
9.31	SolarHeatFlow.mo	360
9.32	SpanData.mo	361

9.33	StandAloneCatenary.mo	361
9.34	StandAloneHeatingResistor.mo	361
9.35	TimeData.mo	361
9.36	asinh.mo	362
9.37	package.mo	362
10.1	SIR.mo	388
10.2	SIRn.mo	388
10.3	SIS.mo	389
10.4	compartimento.mo	389
10.5	dengue.mo	391
10.6	dosserotipos.mo	391
10.7	humano.mo	392
10.8	interaccion.mo	393
10.9	package.mo	393
10.10	vector.mo	394
11.1	asignatura.mo	420
11.2	ingresoConstante.mo	421
11.3	ingresoPulso.mo	421
11.4	package.mo	421
11.5	plan.mo	422
12.1	HidroPI.mo	444
13.1	MyWorld3.mo	466
14.1	identificador.mo	478
14.2	identificadorBase.mo	478
14.3	identificadorFriccion.mo	478
14.4	idmasa.mo	478
14.5	masa.mo	480
14.6	ext_initRandNormal.c	480
14.7	ext_RandNormal.c	481



## LISTADO DE EXPERIMENTOS Y PRÁCTICAS

Planta de experimentación 1. Motor eléctrico DC de imanes permanentes	312
Experimento 1.1 Segunda ley de Newton de la rotación	313
Experimento 1.2 Tiempo de asentamiento de la velocidad angular	313
Experimento 1.3 Velocidad angular estacionaria	313
Experimento 1.4 Efectos de la carga mecánica en el circuito eléctrico	314
Experimento 1.5 Simulación de dispositivos	314
Experimento 1.6 Suma de potencias	314
Planta de experimentación 2. Calentamiento de un cable aéreo desnudo	327
Experimento 2.1 Tiempos de respuesta	328
Experimento 2.2 Fuentes de calentamiento y enfriamiento	328
Experimento 2.3 Ubicación geográfica	328
Experimento 2.4 Estaciones	328
Experimento 2.5 Margen de cargabilidad	329
Planta de experimentación 3. Capacidad de carga	331
Experimento 3.1 Efecto de las condiciones ambientales	332
Experimento 3.2 Condición límite	332
Experimento 3.3 Capacidad horaria	332
Experimento 3.4 Capacidad de carga por flecha	332
Planta de experimentación 4. Análisis de flecha	334
Experimento 4.1 Flecha vs temperatura de conductor	334
Experimento 4.2 Análisis de sensibilidad	335
Experimento 4.3 Aproximaciones de linea recta	335
Experimento 4.4 Efecto del conductor	335
Experimento 4.5 Efecto del vano	335
Planta de experimentación 5. Catenaria	337
Experimento 5.1 Robustez del algoritmo	338
Experimento 5.2 Condiciones de tendido	338
Experimento 5.3 Longitudes	338
Experimento 5.4 Tipos de conductor	338
Experimento 5.5 Calibre	339

<b>Planta de experimentación 6. Modelo Susceptible-Infectado-Recuperado</b>	<b>373</b>
Experimento 6.1      Estado estacionario	373
Experimento 6.2      Tiempos de respuesta	373
Experimento 6.3      Infección máxima	374
Experimento 6.4      Tiempo de infección máxima	374
<b>Planta de experimentación 7. Modelo Susceptible-Infectado-Recuperado con nacimientos</b>	<b>376</b>
Experimento 7.1      Frecuencia	376
Experimento 7.2      Linealización S-R	376
Experimento 7.3      Infección mínima	377
<b>Planta de experimentación 8. Modelo Susceptible-Infectado-Susceptible</b>	<b>379</b>
Experimento 8.1      Estado estacionario	379
Experimento 8.2      Tiempos de respuesta	379
Experimento 8.3      Nacimientos y muertes	380
<b>Planta de experimentación 9. Modelo SIR para humanos y SI con dos cepas para mosquito</b>	<b>381</b>
Experimento 9.1      Tiempos de vida	381
Experimento 9.2      Poblaciones iniciales	382
Experimento 9.3      Infección máxima	382
Experimento 9.4      Entrada del serotipo 2	382
Experimento 9.5      Modelo del dengue	382
Experimento 9.6      Una única cepa	383
<b>Planta de experimentación 10. Respuesta al escalón</b>	<b>413</b>
Experimento 10.1     Número estable de estudiantes	413
Experimento 10.2     Tiempo de estabilización	413
Experimento 10.3     Condiciones equivalentes de referencia	414
Experimento 10.4     Ingresos vs traslados	414
Experimento 10.5     Poblaciones por semestre	414
<b>Planta de experimentación 11. Respuesta al impulso</b>	<b>416</b>
Experimento 11.1     Tiempo medio de graduación	417
Experimento 11.2     Dispersión del tiempo de graduación	417
Experimento 11.3     Tasa de graduación	417
Experimento 11.4     Traslados	417
Experimento 11.5     Condiciones equivalentes de referencia	418

<b>Planta de experimentación 12. Central hidroeléctrica con control PI de frecuencia</b>	<b>435</b>
Experimento 12.1     Control de frecuencia	435
Experimento 12.2     Efecto de la demanda	436
Experimento 12.3     Embalse	436
Experimento 12.4     Planta sin control de frecuencia	436
Experimento 12.5     Oscilaciones de frecuencia	436
Experimento 12.6     Planta con control proporcional	437
<b>Planta de experimentación 13. World3</b>	<b>456</b>
Experimento 13.1     Recursos disponibles	456
Experimento 13.2     Eficiencia en la obtención de recursos no renovables	456
Experimento 13.3     Control de la contaminación	457
Experimento 13.4     Obsolescencia tecnológica	457
Experimento 13.5     Escenarios de Meadows	457
Experimento 13.6     Sensibilidad y robustez	457
<b>Planta de experimentación 14. Masa con fricción</b>	<b>472</b>
Experimento 14.1     Ruido en las mediciones	472
Experimento 14.2     Estímulo	472
Experimento 14.3     Fricción	473
Experimento 14.4     Reinicio de la identificación	473
Experimento 14.5     Filtrado de señales	473
Experimento 14.6     Filtrado de mediciones	473
Experimento 14.7     Tiempo mínimo de identificación	473



## INTRODUCCIÓN

En la mayoría de las áreas del conocimiento la experimentación es fundamental para su avance. En particular, la ingeniería y las ciencias físicas dependen directamente de ella. Una simulación es un experimento realizado sobre un modelo de un sistema bajo estudio. Hoy en día es práctica usual de ingeniería realizar tales simulaciones sobre modelos de software con sofisticadas herramientas de cómputo.

Desde el punto de vista pedagógico, la simulación es una tarea integradora de conocimientos y habilidades: para la construcción de los modelos a simular es necesario tener un dominio suficiente de los principios que rigen los fenómenos a simular; para seleccionar las condiciones de simulación es necesario conocer los rangos de validez de los modelos desarrollados; para diseñar los experimentos es necesario interpretar el contexto en el que operan los sistemas a ser simulados; para analizar los resultados es necesario dominar las herramientas matemáticas y de software.

Este documento presenta el diseño y la implementación de un ambiente de simulación amplio y abierto. A la herramienta obtenida le he dado el nombre UNVirtualLab, en donde las dos primeras letras hacen alusión directa a la Universidad Nacional de Colombia, mi alma máter. Actualmente se gestiona su instalación como parte de los servicios de la Dirección Nacional de Innovación Académica de dicha universidad.

### ¿Qué es UNVirtualLab?

UNVirtualLab es un ambiente web de simulación de sistemas dinámicos enfocado al aprendizaje. En UNVirtualLab se pueden alojar plantas de experimentación de dominios muy variados. En la versión inicial se han incorporado modelos de Ingeniería, Biología y Economía para destacar su naturaleza multidominio. Estos ejemplos se han implementado a través de 14 plantas de experimentación, que en total tienen 961 variables internas y 4112 parámetros internos. Se ha elaborado una colección de 70 experimentos sugeridos sobre esas plantas.

Para actuar sobre esas plantas de experimentación se han diseñado 113 controles agrupados en 40 grupos. Los resultados de la simulación se muestran en 48 figuras que grafican el comportamiento de 85 variables seleccionadas, que también son tabuladas. Adicionalmente se han diseñado un par de animaciones 2D para ilustrar el comportamiento de 5 variables.

En mi opinión, este tipo de herramientas desarrolladas desde instituciones públicas deben promover la democratización del conocimiento. Por tanto, el acceso a los experimentos virtuales que allí se alojen no debe tener restricciones. Este convencimiento guió varias decisiones funcionales y de diseño, como por ejemplo:

- Los experimentos que se alojan en UNVirtualLab pueden ser embebidos en ambientes de aprendizaje. Esto significa que un curso administrado desde una plataforma como moodle puede utilizar internamente un experimento específico; para ello, puede usarse el formato SCORM o embeber un código HTML.
- Los modelos de software de las plantas de experimentación se construyen con un lenguaje abierto (Modelica) y se diseñan, prueban, simulan y ajustan en una herramienta de software abierto (OpenModelica).
- Tanto el código fuente de UNVirtualLab como el de los modelos que en él se alojan tienen licenciamiento libre (ver apéndice A). De esta forma, los usuarios pueden acceder al código de los modelos y modificarlos en OpenModelica.

## UNVirtualLab como un producto de investigación

UNVirtualLab es el resultado de un esfuerzo que satisface los cinco criterios establecidos por la OECD para ser considerado como *actividad de investigación* (ver [OEC15], numeral 2.7.):

- Es *novedoso*, en tanto se ha obtenido un laboratorio virtual en el que se simplifica notablemente el desarrollo de nuevos experimentos virtuales, en comparación con alternativas preexistentes. El enfoque multidominio tambien es novedoso en el ámbito de las simulaciones web<sup>1</sup>.

---

<sup>1</sup>Compárense los ejemplos presentados en la sección 5.2 con las especificaciones de UNVirtualLab disponibles en la sección 6.1.

- Es *creativo*, en tanto articula de forma original dos ámbitos de software relativamente distantes: la simulación de sistemas dinámicos complejos, por una parte, y la interacción web, por otra<sup>2</sup>.
- El proceso no estuvo exento de la *incertidumbre* propia del desarrollo de prototipos. UNVirtualLab es un prototipo que permite probar una estrategia específica para llevar a la web simulaciones de sistemas dinámicos basadas en modelos de escritorio. La incertidumbre queda explícita al destacar que la versión inicial de UNVirtualLab se reveló inoperativa, lo que obligó a un cambio importante en la arquitectura interna del sistema<sup>3</sup>.
- El proceso ha sido *sistemático*, en tanto se han empleado los procedimientos conocidos de desarrollo de software, tales como identificación de actores, requerimientos funcionales, casos de uso, etc.<sup>4</sup>.
- El resultado es *transferible*, en tanto su desarrollo y resultado se han documentado profusamente. El código fuente de UNVirtualLab y de los experimentos allí alojados tienen, además, licenciamiento libre, lo que asegura su reproducibilidad.

El tipo de investigación al que nos referimos aquí corresponde al *desarrollo experimental* (ver [OEC15], numeral 2.9.) que se dirige a la obtención de nuevos productos. En ese sentido, puede identificarse una cadena convencional de generación de conocimiento, que inicia en el estudio de las Ecuaciones Diferenciales Algebraicas (*investigación básica*), continua con el desarrollo del lenguaje Modelica y el ambiente de simulación OpenModelica (*investigación aplicada*) y enlaza con el desarrollo de UNVirtualLab<sup>5</sup>.

## El documento

Este documento se ha elaborado para explicar qué es UNVirtualLab, para qué puede utilizarse, cómo está diseñado, cómo se utiliza, y para presentar las plantas experimentales que se han incorporado. El documento se ha organizado en 14 capítulos agrupados en tres partes. El propósito de cada uno de estas partes y capítulos se explica a continuación:

---

<sup>2</sup>El concepto de diseño utiliza una estrategia sencilla que se explica en la sección 6.1.2.

<sup>3</sup>Los principales cambios se enumeran en el apéndice E.

<sup>4</sup>Ampliamente documentados en el capítulo 6.

<sup>5</sup>El proceso es, por supuesto, mucho más complejo dado que debería incluirse toda la red de generación de conocimiento que ha llevado al desarrollo de software, de la internet, del modelado y simulación de sistemas dinámicos, etc. Esta cadena simple solo pretende ilustrar la condición de UNVirtualLab como actividad de desarrollo experimental.

**Parte I:** aborda el problema de construcción de modelos y la simulación de los mismos en el ambiente OpenModelica. Esta compuesta por cuatro capítulos:

Capítulo 1: está dedicado a presentar de forma comparativa una colección de técnicas de modelado de sistemas dinámicos. Tiene como propósito explicar las razones por las que se ha seleccionado la técnica de modelado orientado a objetos en general y el ambiente OpenModelica en particular para la construcción de los modelos que residen en UNVirtualLab.

Capítulo 2: está orientado a mostrar las dificultades de la simulación numérica de Ecuaciones Diferenciales Algebraicas como las que resultan al emplear la técnica de modelado orientado a objetos. En este capítulo también se presentan los algoritmos y estrategias usadas para abordar esas dificultades en los ambientes de simulación.

Capítulo 3: está destinado a presentar las características principales del lenguaje de modelado Modelica. También se hace una breve presentación de la Modelica Standard Library.

Capítulo 4: se utiliza para presentar la suite OpenModelica. El enfoque utilizado combina una explicación de los procedimientos internos de simulación con una presentación del potencial de las principales herramientas que conforman la suite.

**Parte II:** dedicada a presentar el laboratorio virtual UNVirtualLab. Contiene tres capítulos:

Capítulo 5: está dedicado a mostrar el estado del arte relativo a los laboratorios virtuales. Inicialmente se realiza una caracterización y clasificación de los mismos, para luego evaluar su papel en el aprendizaje de la ingeniería. Este capítulo concluye con unas consideraciones generales útiles para el diseño de UNVirtualLab.

Capítulo 6: se presenta en detalle la arquitectura de UNVirtualLab. A partir de las especificaciones funcionales se presentan los

componentes, el modelo interno, algunos detalles de implementación y la interfaz. Este capítulo se complementa con información que se encuentra en los apéndices B, C y D.

**Capítulo 7:** este capítulo contiene los manuales de UNVirtualLab. Se han preparado tres manuales diferentes:

**Manual del administrador:** el administrador es la persona encargada de la instalación y mantenimiento del sistema. Este manual explica cómo llevar a cabo esas tareas.

**Manual del modelador:** el modelador es la persona encargada de construir las plantas de experimentación que se alojan en UNVirtualLab. Su rol va más allá de la construcción del modelo que se simulará, pues también debe reflexionar sobre el uso pedagógico de los modelos, y preparar la documentación asociada. Por esta razón, este manual presenta una guía para el diseño de las plantas de experimentación con un enfoque pedagógico.

**Manual del experimentador:** el experimentador es el usuario final de UNVirtualLab. En este manual se explica el uso de la interfaz gráfica disponible. Esta explicación se complementa con un conjunto de videos disponibles en la implementación web de UNVirtualLab.

**Parte III:** se presentan los modelos disponibles en la instalación web de UNVirtualLab a la fecha de elaboración de este documento. El contenido de cada capítulo es la misma documentación que está disponible en la implementación web para cada una de las plantas de experimentación. Los capítulos que aparecen en esta parte tienen todos la misma estructura: se presenta una explicación teórica de los modelos subyacentes, una descripción de cada planta de experimentación, un listado de experimentos sugeridos y una breve explicación de la estrategia de implementación. Las plantas de experimentación se han agrupado en 7 capítulos cuyos modelos corresponden a:

**Capítulo 8:** el modelo básico de un motor eléctrico DC acoplado a una carga mecánica. Este modelo es el ‘Hola Mundo’ en el modelado orientado a objetos, y se incluye en los instaladores de UNVirtualLab.

**Capítulo 9:** un modelo del calentamiento de cables aéreos en líneas de transmisión de energía eléctrica. El modelo incorpora problemas eléctricos, térmicos y mecánicos. Este modelo tiene unas ecuaciones algebraicas implícitas, que son resueltas por OpenModelica de forma transparente para el modelador.

**Capítulo 10:** un grupo de modelos de propagación de enfermedades. Estos modelos están basados en la compilación e implementación desarrollada en [Cam12].

**Capítulo 11:** un modelo del tránsito de estudiantes a través de un plan de estudios. Este es un modelo original presentado en [Dua13] como parte de las actividades desarrolladas en la Vicedecanatura Académica de la Facultad de Ingeniería en la Universidad Nacional de Colombia.

**Capítulo 12:** un modelo de una central de generación hidroeléctrica que emplea la librería PowerSystems<sup>6</sup>. El modelo combina sistemas hidráulicos, mecánicos, eléctricos y de control. La planta utilizada incluye un lazo de control de frecuencia con un controlador PI.

**Capítulo 13:** un modelo de la evolución del mundo propuesto por [MIM<sup>+</sup>74] y basado en dinámica de sistemas. La planta de experimentación es una adaptación simple de la librería desarrollada por Cellier en [Cel08].

**Capítulo 14:** un modelo de la técnica de identificación algebraica propuesto por Sira. El ejemplo seleccionado es el más sencillo que aparece en [SRGRCRLJ14]: la identificación de una masa sobre la que se ejerce una fuerza.

Los estilos de redacción utilizados a lo largo del documento son variados. Los capítulos 1 a 4 se han redactado a manera de Texto Guía, y pueden ser utilizados en un módulo de algún curso de posgrado dedicado al modelado orientado a objetos. Estos capítulos presuponen que el lector está familiarizado con los conceptos de análisis y modelado de sistemas dinámicos, métodos numéricos de solución de ecuaciones, métodos numéricos de integración y simulación, ecuaciones diferenciales ordinarias y programación orientada a objetos. El capítulo 5 es un estado del arte, y como tal se ha redactado. Los

---

<sup>6</sup> [link:1].

capítulos 6 y 7 tienen el estilo de reportes técnicos y manuales de usuario respectivamente. Por su parte, los capítulos 8 a 14 tienen una combinación de estilos: texto guía para la presentación de los modelos y reporte técnico para la presentación de las plantas y la explicación de la implementación.

**Oscar Duarte**

Departamento de Ingeniería Eléctrica y Electrónica  
Universidad Nacional de Colombia



## **Parte I**

# **Modelado y simulación con OpenModelica**



# 1

## MODELADO DE SISTEMAS DINÁMICOS



En este capítulo se comparan algunas técnicas de modelado de sistemas dinámicos con el propósito de presentar las razones por las que se ha optado por una de ellas, el modelado orientado a objetos, para el desarrollo de los modelos alojados en UNVirtualLab. Las alternativas comparadas son: técnicas analíticas específicas de cada dominio (sección 1.2.1), técnicas de espacio de estado (sección 1.2.2), técnicas variacionales (sección 1.2.3), técnicas de diagramas de bloques (sección 1.2.4), grafos de enlace de potencia (sección 1.2.5) y modelado orientado a objetos (sección 1.2.6). Como preámbulo, en la sección 1.1 se presentan algunos elementos conceptuales básicos asociados a las tareas de modelado y simulación. La comparación de las técnicas se resume en la sección 1.3, mientras que en la sección 1.4 se justifica la elección de la técnica de modelado orientado a objetos.

## 1.1 SISTEMAS, MODELOS, SIMULACIÓN

En el contexto del estudio de los sistemas dinámicos es usual adoptar una definición muy amplia del término *sistema* como por ejemplo: “el objeto o colección de objetos cuyas propiedades queremos estudiar” ([LG94]) o, de forma más escueta, “aquello que se desea estudiar”. ([Dua05]). Estas amplias definiciones permiten abarcar objetos de estudio de disciplinas tan variadas como la biología, la astronomía, la mecánica, etc.

El énfasis de esas definiciones está en el propósito de estudiar el sistema. Para hacerlo, construimos representaciones de ese sistema que reciben el nombre de *modelos*. En términos generales, un modelo debe facilitarnos estudiar y, por tanto, entender el sistema.

Algunos de los modelos están construidos de tal manera que permiten realizar experimentos sobre ellos. El objetivo es experimentar sobre el modelo, en lugar de hacerlo sobre el sistema. Tales experimentos se denominan *simulaciones*. Si el modelo representa adecuadamente al sistema, entonces los resultados de una simulación reflejarán el comportamiento que habría tenido el sistema si el experimento se hubiese realizado directamente sobre él.

Hay una amplia variedad de tipos de modelo: gráficos, mentales, verbales, matemáticos, de *software* etc. En este capítulo nos interesa abordar el problema de construir modelos que permitan alimentar un laboratorio virtual, es decir, *modelos matemáticos a partir de los cuales se puedan obtener modelos de software* para realizar simulaciones en computador del sistema bajo estudio.

Un mismo sistema puede representarse de formas muy variadas; en otras palabras, podemos construir modelos diferentes de un mismo sistema. Para juzgar cuál de los modelos disponibles de un cierto sistema es el adecuado, deben usarse los criterios de *utilidad* y *simplicidad* del modelo. Se debe seleccionar un modelo que cumpla las funciones que esperamos de él, y entre todos los modelos que lo hagan, el más adecuado será el más sencillo. No existe, entonces, el modelo de un sistema dado. Por el contrario, la elección del modelo está condicionada por el tipo de estudio que querramos realizar sobre ese sistema<sup>1</sup>. El criterio de utilidad tiene al menos dos dimensiones:

- El modelo debe ser capaz de reflejar de forma adecuada los fenómenos que se quieren estudiar.
- El modelo debe poder utilizarse con las herramientas de experimentación (de simulación) disponibles.

Por esta razón, las técnicas de construcción de modelos están estrechamente relacionadas con las herramientas de simulación disponibles. La sofisticación de las unas van de la mano de las otras. En el caso particular de los modelos de *software*, la evolución de las técnicas de construcción ha tenido además otros condicionantes, tales como la evolución del *hardware* y el desarrollo de métodos numéricos computacionales adecuados para la simulación de sistemas cada vez más complejos (véase [OC95]).

### 1.1.1 Construcción de modelos matemáticos

Es usual clasificar los modelos matemáticos según la estrategia usada para su construcción. En [OR94] se distinguen dos casos:

**Modelos Empíricos:** son modelos obtenidos como ajuste de funciones matemáticas, generalmente simples, a partir de información sobre el comportamiento externo del modelo. También se denominan modelos *de entrada-salida* o *de caja negra*. El procedimiento empleado para la obtención de estos modelos se denomina *Identificación de sistemas*.

---

<sup>1</sup>Ljung lo expresa de forma diáfana: “El sistema de la vida real es un objeto de una clase diferente a nuestros modelos matemáticos. En cierto sentido, existe una pantalla impenetrable pero transparente entre nuestro mundo de las descripciones matemáticas y el mundo real. Podemos mirar a través de esa ventana y comparar ciertos aspectos del sistema físico con su descripción matemática, pero no podremos nunca establecer ninguna conexión exacta entre ellos. La pregunta de si la naturaleza es susceptible de ser descrita matemáticamente tiene algunos aspectos filosóficos profundos, y en términos prácticos debemos tener una visión más pragmática sobre los modelos. Nuestra aceptación de los modelos debe estar guiada por su ‘utilidad’ en lugar de su ‘verdad’”[Lju87].

Modelos Teóricos: son modelos obtenidos a partir del conocimiento previo sobre los fenómenos que se presentan en el sistema y que son relevantes para el estudio que se desea realizar, así como de las leyes que los rigen<sup>2</sup>. Debido a que estos modelos se apoyan en una descripción de lo que está sucediendo dentro del sistema, también se denominan modelos *internos* o *de caja blanca*. El procedimiento empleado para la obtención de estos modelos se denomina *Modelado teórico de sistemas* o simplemente *Modelado de sistemas*.

Por supuesto, también es posible combinar las estrategias de identificación y modelado: se pueden utilizar las técnicas de identificación para encontrar los valores numéricos más adecuados para los parámetros de un modelo teórico; así mismo, se puede emplear el conocimiento teórico previo que se tenga del sistema para seleccionar la función matemática que se debe ajustar en una identificación.

El enfoque usual de los laboratorios virtuales (véase capítulo 5) hace que los modelos empíricos no sean los más adecuados. Por ejemplo, UNVirtualLab tiene un enfoque hacia el aprendizaje (véase capítulo 6) y la descripción interna de los sistemas es fundamental en el proceso de aprendizaje; por tanto, es preferible emplear modelos internos<sup>3</sup>.

En [OR94] se propone un procedimiento general para el desarrollo de modelos teóricos que incluye una iteración sobre cuatro fases:

- Md.1. Definición del problema: en esta fase se definen aspectos generales tales como el propósito del modelo, los aspectos relevantes a modelar, el nivel de complejidad, el grado de conocimiento de los fenómenos involucrados, los recursos que se pueden emplear en la tarea de modelado, etc.
- Md.2. Formulación del modelo: en esta fase se construyen las relaciones matemáticas que describen el fenómeno.
- Md.3. Estimación de parámetros: en esta fase se encuentran los valores numéricos de los parámetros que aparecen en el modelo matemático. Es

---

<sup>2</sup>En [LG94] a los modelos de este tipo se les denomina modelos *físicos*. En este documento se ha preferido el término *teórico*, porque los modelos sociales y económicos construidos a partir de las leyes de sus respectivos dominios, difícilmente pueden calificarse como *físicos*.

<sup>3</sup>Los métodos de identificación también pueden ser objeto de estudio en un laboratorio virtual. Por ejemplo, en el capítulo 14 se presenta un modelo interno de un proceso de identificación de sistemas.

usual que se requiera algún tipo de experimentación física sobre la planta, o al menos el registro de su comportamiento natural para encontrar estos valores.

- Md.4. Validación del modelo: en esta fase se evalúa el desempeño del modelo para determinar si es lo suficientemente útil para nuestras necesidades. El *Rango de validez* del modelo es el conjunto de condiciones dentro de las cuales el modelo satisface esas necesidades.

En las siguientes secciones, el foco de atención se centra en la segunda de estas fases: la formulación del modelo; específicamente, en la formulación de modelos matemáticos a partir de los cuales se puedan obtener modelos de *software* para ser simulados en computador. En la sección 1.2 se presentan las técnicas más conocidas, cuyas características se comparan en la sección 1.3, mientras en la sección 1.4 se explica por qué se ha seleccionado la técnicas de modelado orientado a objetos para la construcción de los modelos simulados en UNVirtualLab.

## **1.2 TÉCNICAS DE MODELADO MATEMÁTICO CON FINES DE SIMULACIÓN**

A continuación se presenta un conjunto de técnicas de construcción de modelos matemáticos mediante las cuales se pueden obtener ecuaciones susceptibles de ser simuladas por computador. El énfasis está en la obtención de modelos dinámicos, es decir, modelos cuyo comportamiento actual depende del pasado. No es de extrañar, entonces, que las ecuaciones resultantes incluyan derivadas, integrales o diferencias finitas.

Algunas de las técnicas son más susceptibles de generalizar y sistematizar que otras. Las técnicas que se presentan en esta sección se han organizado de menor a mayor grado de sistematización. La presentación consiste en una somera explicación de los principios de cada una de las técnicas y no pretende ser un tratado sobre ninguna de ellas. Los conceptos se ilustran con ejemplos de construcción de modelos sobre un mismo sistema, que corresponde al circuito lineal de la figura 1.1 con  $L_1 = L_2 = 1H$ ,  $C = 1F$ , estimulado con un escalón unitario  $E(t) = \mu(t)$ .

### **1.2.1 Técnicas analíticas específicas de cada dominio**

En contexto de este documento, se utiliza el término *dominio* para referirse a un área delimitada del conocimiento, tal como la biología, la electricidad, la

termodinámica, etc. Al interior de cada uno de estos dominios se han desarrollado técnicas específicas para la construcción de los modelos matemáticos de los fenómenos que allí se estudia, de forma más o menos aislada respecto a otros dominios. En las aplicaciones típicas de control, se analizan modelos de los dominios eléctrico, magnético, mecánico rotacional, mecánico traslacional, fluidos, intercambios de calor e intercambios de masa<sup>4</sup>.

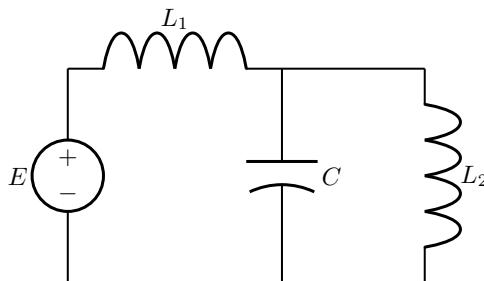


Figura 1.1 Circuito de los ejemplos de modelado mediante diversas técnicas

En el desarrollo de modelos específicos de cada dominio la estrategia básica es la misma:

1. Descomponer (abstractamente) el sistema en subsistemas más simples.
2. Obtener los modelos de los subsistemas mediante la aplicación de leyes o principios elementales.
3. Buscar las relaciones que existen entre los modelos obtenidos (de nuevo, mediante la aplicación de leyes elementales) e interconectarlos para obtener el modelo del sistema original.

Al interior de cada dominio se han desarrollado técnicas específicas que facilitan la obtención de los modelos analíticos. La aplicabilidad de esas técnicas está, por tanto, restringida a su propio dominio. Ejemplos de estas técnicas específicas son: análisis de mallas y nodos para circuitos eléctricos, análisis de cuerpos libres para sistemas mecánicos, técnicas basadas en el volumen de control para fenómenos termodinámicos, entre otros casos.

La similitud entre las expresiones matemáticas que aparecen en diferentes dominios, como modelos explicativos de fenómenos distintos, ha permitido la

---

<sup>4</sup>Véanse, por ejemplo el capítulo 5 de [LG94], o los capítulos 3 y 5 de [Wel00].

aparición de analogías tales como: el modelado de fenómenos magnéticos como si fuesen circuitos eléctricos resitivos, el modelado de fenómenos de transferencia de calor como circuitos eléctricos resitivos y capacitivos, etc. Estas similitudes también han permitido la aparición de técnicas generalizadas de modelado. Se trata de técnicas que utilizan variables generalizadas en conjunto con métodos de redes y grafos, que permiten entender los fenómenos de cada dominio (y sus técnicas de modelado) como casos particulares de fenómenos (y técnicas) más generales<sup>5</sup>. Sin embargo, estas técnicas generalizadas siguen siendo específicas de un dominio en el sentido de que son aplicables *a cada dominio por separado*.

Las técnicas de modelado básicas para cada dominio suelen tener un enfoque analítico, queriendo decir con esto que los modelos matemáticos suelen ser desarrollados con el propósito de facilitar el análisis de los fenómenos estudiados. A medida que los fenómenos van ganando complejidad, el análisis se dificulta y por tanto es necesario utilizar modelos enfocados hacia la simulación numérica. Para ilustrar la diferencia entre estos dos enfoques, consideremos el modelado de circuitos eléctricos mediante dos técnicas, que se ilustran en los ejemplos 1.1 y 1.2:

Análisis de mallas: esta es una de las técnicas elementales de modelado de circuitos<sup>6</sup>. El modelador debe identificar las mallas existentes en el circuito y asignar una corriente a cada una de ellas. Las ecuaciones se construyen a partir de la aplicación de la ley de tensiones de Kirchhoff, en conjunto con las características tensión-corriente de cada uno de los elementos del circuito.

Para circuitos simples, el procedimiento genera un conjunto reducido de ecuaciones susceptibles de ser analizadas manualmente, razón por la cual es la técnica favorita de los modeladores nóveles. Sin embargo, cuando la complejidad del circuito aumenta, aparecen algunas dificultades:

- Las fuentes de corriente, tanto dependientes como independientes, obligan a modificar el procedimiento básico.
- Si el grafo del circuito no es plano, el conjunto resultante de ecuaciones es redundante.

---

<sup>5</sup>Véase, por ejemplo, el texto clásico de Wellstead [Wel79], ahora disponible en formato digital abierto: [Wel00].

<sup>6</sup>La explicación se puede encontrar en cualquier texto básico de teoría de circuitos, como por ejemplo en [HDK07].

- Si el circuito contiene elementos controlados por tensión, es necesario modificar el procedimiento básico.
- Si el número de mallas es grande, el análisis manual puede ser difícil o ineficiente, cuando no imposible.

*Tableau analysis:* esta es una técnica orientada a la obtención de modelos de circuitos para ser simulados en computador (véase [CDK87]). El circuito debe representarse por un dígrafo, a partir del cual se escriben de forma sistemática: 1) un conjunto de ecuaciones provenientes de la aplicación de la ley de corrientes de Kirchhoff, y 2) las ecuaciones de cada rama a parir de las relaciones tensión-corriente de cada elemento.

Con esta técnica, el conjunto de ecuaciones resultante es mayor que al aplicar el análisis de mallas. Además, está formado por una mezcla de ecuaciones diferenciales y ecuaciones algebraicas. No obstante, estas ecuaciones están organizadas de tal manera que se pueden aplicar algoritmos generales de solución y simulación previamente implementados en *software*<sup>7</sup>.

---

**Ejemplo 1.1: Modelado analítico con técnicas específicas de un dominio.**

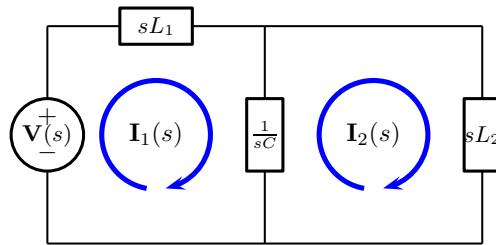
---

El sistema de la figura 1.1 corresponde a un circuito dinámico lineal. Como tal, puede modelarse empleando la técnica de análisis en el dominio de la frecuencia compleja: los elementos dinámicos se representan por sus impedancias complejas, y posteriormente se realiza un análisis convencional de mallas.

Los inductores se representan por la impedancia compleja  $sL$ , y los capacitores por  $1/sC$ , en donde  $s$  es la variable compleja. Esta técnica de análisis no es otra cosa que la utilización directa de la transformada de Laplace. En este ejemplo suponemos que las condiciones iniciales de los elementos dinámicos son cero, es decir, que no hay energía almacenada en el instante de tiempo  $t = 0$ . En estas condiciones, el sistema se puede representar por el siguiente circuito:

---

<sup>7</sup>Los simuladores de circuitos eléctricos más populares emplean métodos de Newton para solución de ecuaciones algebraicas, y métodos de Euler para simulación de sistemas dinámicos. No obstante, en la sección 2.6 se muestra las dificultades que este último método puede tener aún en casos simples.



Al emplear un análisis convencional de mallas se obtienen dos ecuaciones:

$$\begin{aligned} sL_1\mathbf{I}_1(s) + \frac{1}{sC}(\mathbf{I}_1(s) - \mathbf{I}_2(s)) &= \mathbf{V}(s) \\ \frac{1}{sC}(\mathbf{I}_2(s) - \mathbf{I}_1(s)) + sL_2\mathbf{I}_2(s) &= 0 \end{aligned} \quad (1.1)$$

Podemos despejar  $\mathbf{I}_1(s)$  de la segunda ecuación y remplazar en la primera:

$$\begin{aligned} \mathbf{I}_1(s) &= (s^2L_2C + 1)\mathbf{I}_2(s) \\ \mathbf{V}(s) &= \left(sL_1 + \frac{1}{sC}\right)(s^2L_2C + 1)\mathbf{I}_2(s) - \frac{\mathbf{I}_2(s)}{sC} \\ \mathbf{V}(s) &= (s^3L_1L_2C + sL_1 + sL_2)\mathbf{I}_2(s) \\ \mathbf{I}_2(s) &= \frac{1}{s^3L_1L_2C + sL_1 + sL_2}\mathbf{V}(s) \\ \mathbf{I}_1(s) &= \frac{s^2L_2C + 1}{s^3L_1L_2C + sL_1 + sL_2}\mathbf{V}(s) \end{aligned} \quad (1.2)$$

Para obtener los valores de las corrientes en el tiempo, usamos los valores  $L_1 = L_2 = 1H$ ,  $C = 1F$ ,  $E(t) = \mu(t)$ , aplicamos fracciones parciales y luego calculamos la transformada inversa de Laplace para  $t \geq 0$ :

$$\begin{aligned} \mathbf{I}_1(s) &= \frac{s^2 + 1}{s(s^2 + 2)} \frac{1}{s} \\ \mathbf{I}_2(s) &= \frac{1}{s(s^2 + 2)} \frac{1}{s} \end{aligned} \quad (1.3)$$

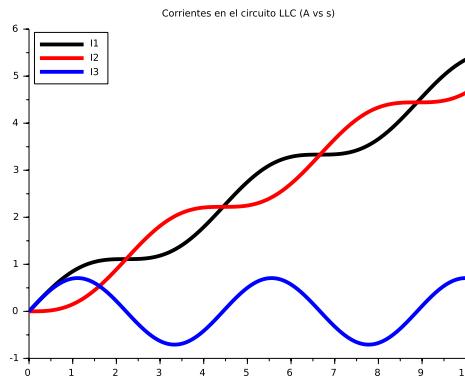
$$\begin{aligned} \mathbf{I}_1(s) &= \frac{0,5}{s^2} + \frac{0,5}{\sqrt{2}} \frac{\sqrt{2}}{s^2 + 2} \\ \mathbf{I}_2(s) &= \frac{0,5}{s^2} - \frac{0,5}{\sqrt{2}} \frac{\sqrt{2}}{s^2 + 2} \end{aligned} \quad (1.4)$$

$$\begin{aligned} i_1(t) &= 0,5t + \frac{0,5}{\sqrt{2}} \sin(\sqrt{2}t) \\ i_2(t) &= 0,5t - \frac{0,5}{\sqrt{2}} \sin(\sqrt{2}t) \end{aligned} \quad (1.5)$$

También es posible calcular la corriente en el capacitor,  $i_3(t)$

$$i_3(t) = i_1(t) - i_2(t) = \frac{1}{\sqrt{2}} \sin(\sqrt{2}t) \quad (1.6)$$

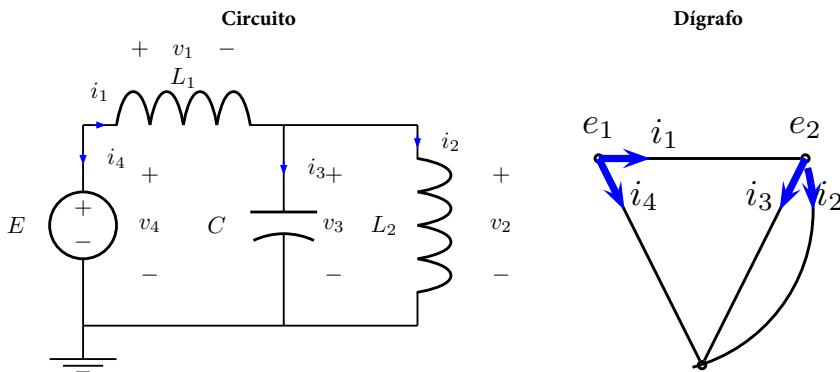
Las tres corrientes  $i_1(t)$ ,  $i_2(t)$ ,  $i_3(t)$  se han graficado utilizando Scilab (ver [Ent16]):



■

### Ejemplo 1.2: Modelado para simulación con técnicas de un dominio.

Considérese el circuito de la figura 1.1. Para aplicar la técnica del tableau, marcamos todas las corrientes y tensiones del circuito y construimos el dígrafo.



A partir del dígrafo construimos la matriz de incidencia reducida, que permite escribir de forma compacta las ecuaciones de corrientes de Kirchhoff:

$$\mathbf{A} = \begin{pmatrix} -1 & 0 & 0 & -1 \\ 1 & -1 & -1 & 0 \end{pmatrix} \quad \mathbf{A} \begin{pmatrix} i_1 \\ i_2 \\ i_3 \\ i_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad (1.7)$$

Las ecuaciones de rama se construyen a partir de las relaciones tensión corriente de cada elemento:

$$v_1 = L_1 \frac{di_1}{dt} \quad v_2 = L_2 \frac{di_2}{dt} \quad i_3 = C \frac{dv_3}{dt} \quad v_4 = E \quad (1.8)$$

Las ecuaciones del circuito son entonces

$$\mathbf{M}_0 \dot{\mathbf{v}} + \mathbf{M}_1 \mathbf{v} + \mathbf{N}_0 \dot{\mathbf{i}} + \mathbf{N}_1 \mathbf{i} = \mathbf{u}_s \quad (1.9)$$

en donde

$$\mathbf{M}_0 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & C & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad \mathbf{M}_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & C & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \mathbf{v} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix} \quad \mathbf{e} = \begin{pmatrix} e_1 \\ e_2 \end{pmatrix} \quad (1.10)$$

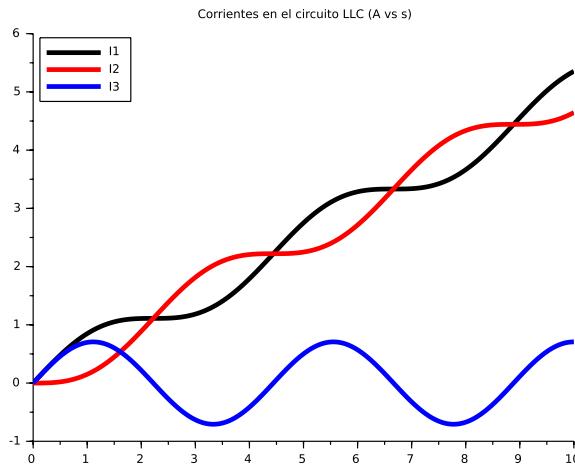
$$\mathbf{N}_0 = \begin{pmatrix} -L_1 & 0 & 0 & 0 \\ 0 & -L_2 & 0 & 0 \\ 0 & 0 & C & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad \mathbf{N}_1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad (1.11)$$

$$\mathbf{i} = \begin{pmatrix} i_1 \\ i_2 \\ i_3 \\ i_4 \end{pmatrix} \quad \mathbf{u}_s = \begin{pmatrix} 0 \\ 0 \\ 0 \\ E \end{pmatrix}$$

lo que permite escribir el sistema de ecuaciones matriciales (el tableau)

$$\begin{aligned} \mathbf{A}\mathbf{i} &= \mathbf{0} \\ -\mathbf{A}^T \mathbf{e} - \mathbf{v} &= \mathbf{0} \\ \mathbf{M}_0 \dot{\mathbf{v}} + \mathbf{M}_1 \mathbf{v} + \mathbf{N}_0 \dot{\mathbf{i}} + \mathbf{N}_1 \mathbf{i} - \mathbf{u}_s &= \mathbf{0} \end{aligned} \quad (1.12)$$

La ecuación 1.12 es de la forma  $\mathbf{g}(t, \mathbf{y}, \dot{\mathbf{y}}) = \mathbf{0}$ , y por tanto puede ser simulada con el algoritmo dassl. Se ha utilizado la implementación disponible en Scilab para la simulación, con  $L_1 = L_2 = 1H$ ,  $C = 1F$ ,  $E = 1V$ . La figura siguiente muestra el resultado para  $i_1$ ,  $i_2$  e  $i_3$ :



■

### 1.2.2 Técnicas de espacio de estado

Los modelos en espacio de estado, o simplemente modelos de estado, se caracterizan por tener la siguiente estructura matemática:

$$\begin{aligned}\dot{\mathbf{x}}(t) &= \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \\ \mathbf{y}(t) &= \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t), t)\end{aligned}\quad (1.13)$$

en donde  $\mathbf{u}(t)$ ,  $\mathbf{y}(t)$  son vectores que contienen las variables de entrada y de salida del sistema, respectivamente. Por su parte,  $\mathbf{x}(t)$  es un vector de *variables de estado* que pueden asimilarse a variables auxiliares seleccionadas para representar la dinámica del sistema en la forma dada por la ecuación 1.13.

Son varias las ventajas de lograr una representación de estado. Desde el punto de vista de la simulación, la ecuación diferencial  $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t)$  tiene la forma adecuada para ser simulada por métodos numéricos. Desde el punto de vista analítico, existe una enorme cantidad de resultados matemáticos y procedimientos bien conocidos para el análisis, la identificación y el control de modelos de estado, en especial para aquellos que además son lineales e invariantes en el tiempo y que, por tanto, pueden representarse como<sup>8</sup>:

$$\begin{aligned}\dot{\mathbf{x}}(t) &= \mathbf{Ax}(t) + \mathbf{Bu}(t) \\ \mathbf{y}(t) &= \mathbf{Cx}(t) + \mathbf{Du}(t)\end{aligned}\quad (1.14)$$

---

<sup>8</sup>Véanse por ejemplo [Che95], [Che93], [OM96], [Kat05].

Las variables de estado pueden carecer de sentido físico directo. No obstante, para la construcción del modelo matemático es conveniente destacar que las variables de estado son las portadoras de la información sobre la dinámica del sistema. Debido a que en los sistemas físicos esta dinámica se relaciona con la acumulación de energía, una buena idea consiste en seleccionar como variables de estado candidatas a las variables asociadas con este fenómeno.

Otra de las grandes ventajas de los modelos de estado consiste en la similitud matemática con la que se pueden representar fenómenos donde la variable tiempo se representa de forma discreta:  $k = 0, 1, 2, \dots$ . En estos casos, el modelo general resulta ser

$$\begin{aligned}\mathbf{x}(k+1) &= \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k), k) \\ \mathbf{y}(k) &= \mathbf{g}(\mathbf{x}(k), \mathbf{u}(k), k)\end{aligned}\tag{1.15}$$

mientras el modelo lineal e invariante en el tiempo corresponde a

$$\begin{aligned}\mathbf{x}(k+1) &= \mathbf{Ax}(k) + \mathbf{Bu}(k) \\ \mathbf{y}(k) &= \mathbf{Cx}(k) + \mathbf{Du}(k)\end{aligned}\tag{1.16}$$

---

### *Ejemplo 1.3: Modelado en espacio de estado.*

Para el circuito de la figura 1.1 los fenómenos de acumulación de energía suceden en los inductores en forma de energía magnética y en el capacitor en forma de energía eléctrica. Para los inductores, la variable directamente asociada con la acumulación de energía ( $w$ ) es la corriente ( $w = Li^2/2$ ), mientras que para el capacitor es la tensión ( $w = Cv^2/2$ ). Por esta razón, seleccionamos las siguientes variables de estado:

$$x_1 = i_{L_1} \quad x_2 = i_{L_2} \quad x_3 = v_c\tag{1.17}$$

La dinámica del circuito puede entonces escribirse como

$$\begin{aligned}E - L_1 \dot{x}_1 - x_3 &= 0 \\ x_3 - L_2 \dot{x}_2 &= 0 \\ -x_1 + x_2 + C \dot{x}_3 &= 0\end{aligned}\tag{1.18}$$

Al despejar las derivadas de las variables de estado se obtiene:

$$\begin{aligned}\dot{x}_1 &= -\frac{1}{L_1}x_3 + \frac{1}{L_1}E \\ \dot{x}_2 &= \frac{1}{L_2}x_3 \\ \dot{x}_3 &= \frac{1}{C}x_1 - \frac{1}{C}x_2\end{aligned}\tag{1.19}$$

Que en forma matricial es

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} 0 & 0 & -1/L_1 \\ 0 & 0 & 1/L_2 \\ 1/C & -1/C & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} 1/L_1 \\ 0 \\ 0 \end{pmatrix} E\tag{1.20}$$

Tomando  $\mathbf{y} = [i_1 \ i_2 \ i_3]^T$  como la salida del sistema, entonces

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & -1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}\tag{1.21}$$

las ecuaciones 1.20 y 1.21 tienen la forma general de la ecuación 1.14. Además, para el caso en que  $L_1 = L_2 = 1H$   $C = 1F$ , la matriz  $\mathbf{A}$  resulta ser:

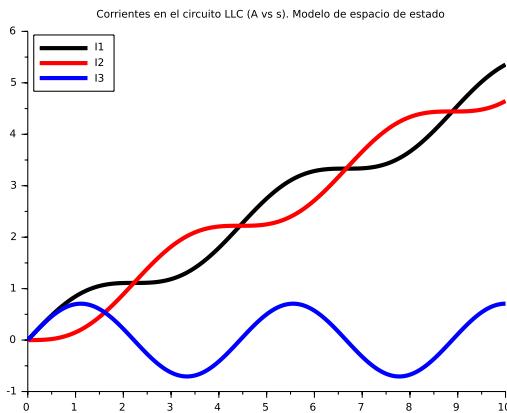
$$\mathbf{A} = \begin{pmatrix} 0 & 0 & -1 \\ 0 & 0 & 1 \\ 1 & -1 & 0 \end{pmatrix}\tag{1.22}$$

cuyos valores propios son  $e_1 = 0$ ,  $e_2 = j\sqrt{2}$ ,  $e_3 = -j\sqrt{2}$ . En tales condiciones, podemos asegurar (véanse, por ejemplo [Che95, Dua05]) que las respuestas de entrada cero del sistema serán de la forma

$$r_{ec}(t) = a_1 t + a_2 \sin(\sqrt{2}t + a_3)\tag{1.23}$$

que, por otra parte, concuerda con el resultado analítico obtenido en las ecuaciones 1.5 y 1.6.

Las ecuaciones 1.20 y 1.21 se han simulado en Scilab para  $L_1 = L_2 = 1H$ ,  $C = 1F$ ,  $E = \mu(t)$ . Para ello se utilizó la instrucción `ode`, que implementa por defecto el método de simulación de Adams para Ecuaciones Diferenciales Ordinarias. La siguiente figura muestra el resultado de la simulación numérica para las corrientes del circuito, que coincide con las simulaciones de los ejemplos 1.1 y 1.2:



■

### 1.2.3 Técnicas variacionales

Ciertos fenómenos físicos pueden explicarse como la condición mediante la cual la naturaleza minimiza una cantidad específica. El principio de Hamilton de mínima acción es quizás el ejemplo más conocido. Este hecho permite formular el problema de modelado matemático como la búsqueda de la función matemática que minimiza esa cantidad específica, expresada como un funcional. Debido a que los problemas de minimización de funcionales son objeto de estudio del cálculo de variaciones, estas técnicas de modelado se denominan de forma genérica *técnicas variacionales*.

Un funcional es una función que opera sobre funciones, asignándole un número real a cada función. En particular, para una función genérica  $y(t)$  podemos definir el funcional  $J : U \subset V \rightarrow \mathbb{R}$  así:

$$J(y) = \int_{t_0}^{t_1} L(t, y(t), \dot{y}(t)) dt \quad (1.24)$$

en donde  $V$  es el conjunto de funciones diferenciables con continuidad  $C^1([t_0, t_1])$ . En esas condiciones, la función  $y(t)$  que minimiza el funcional  $J$  debe satisfacer las condiciones de Euler (véanse [Gal03, Vil09, San03]):

$$\frac{\partial L}{\partial y}(t, y(t), \dot{y}(t)) - \frac{d}{dt} \frac{\partial L}{\partial \dot{y}}(y, y(t), \dot{y}(t)) = 0 \quad \forall t \in (t_0, t_1) \quad (1.25)$$

En términos generales, las técnicas variacionales de modelado consisten en construir el funcional que debe minimizarse y posteriormente aplicar

las condiciones de Euler para hallar las ecuaciones reales del fenómeno. Las técnicas variacionales buscan representar la dinámica del sistema a partir de un conjunto de  $N$  *coordenadas generalizadas* independientes  $q_1(t), q_2(t), \dots, q_N(t)$ , acompañadas de un conjunto de  $N$  *variables dinámicas*, que pueden ser las *velocidades generalizadas*,  $\dot{q}_1(t), \dot{q}_2(t), \dots, \dot{q}_N(t)$  o los *momentum generalizados*  $p_1(t), p_2(t), \dots, p_N(t)$ . El adjetivo *generalizado* hace referencia a la aplicación del método en fenómenos distintos a los de traslación mecánica, en los que fueron inicialmente utilizados.

Para la descripción de una gran variedad de fenómenos físicos en los que una partícula se mueve en un campo conservativo, el funcional que se utiliza es el *lagrangiano*:

$$\mathcal{L}(q, \dot{q}) = \mathcal{T}(q, \dot{q}) - \mathcal{V}(q) \quad (1.26)$$

en donde  $\mathcal{T}(q, \dot{q})$  y  $\mathcal{V}(q)$  representan las energías cinética y potencial respectivamente.

Sin embargo, cuando la partícula está en presencia de campos no conservativos, es necesario modificar el lagrangiano. En [OLNSR98] se propone el siguiente lagrangiano que incluye acciones de control  $\mathcal{U}$  y las perturbaciones  $\mathcal{Q}_\varsigma$ :

$$\mathcal{L}_{NC}(q, \dot{q}, t) = \mathcal{T}(q, \dot{q}) + \int_o^t \mathcal{F}(\dot{q}) dt - (\mathcal{V}(q) + \mathcal{V}_{NC}(q, t)) \quad (1.27)$$

en donde  $\mathcal{F}(\dot{q}) = \frac{1}{2} \sum_j k_j \dot{q}_j^2$  es la función de disipación de Rayleigh y

$$\mathcal{V}_{NC}(q, t) = -(\mathcal{U} + \mathcal{Q}_\varsigma)^T q \quad (1.28)$$

Al incorporar 1.26, 1.27 y 1.28 como la función  $L$  en la ecuación 1.25, se obtiene una versión de las ecuaciones de Euler-Lagrange:

$$\frac{d}{dt} \left( \frac{\partial \mathcal{L}}{\partial \dot{q}} \right) - \frac{\partial \mathcal{L}}{\partial q} = \mathcal{U} + \mathcal{Q}_\varsigma - \frac{\partial \mathcal{F}}{\partial \dot{q}} \quad (1.29)$$

Si, además, el sistema está sujeto a restricciones, entonces puede usarse la técnica de los multiplicadores de Lagrange para minimizar el funcional. Por ejemplo, en [SJK03] se utiliza una estrategia desarrollada para el análisis de circuitos eléctricos, donde las leyes de Kirchhoff se expresan como restricciones. El lagrangiano utilizado es la diferencia entre la coenergía magnética y la energía de campo eléctrico:

$$\mathcal{L}(q, \dot{q}) = \mathcal{T}_{\mathcal{M}}(q, \dot{q}) - \mathcal{V}_{\mathcal{E}}(q) \quad (1.30)$$

y la ecuación 1.29 se transforma en:

$$\frac{d}{dt} \left( \frac{\partial \mathcal{L}}{\partial \dot{q}}(q, \dot{q}) \right) - \frac{\partial \mathcal{L}}{\partial q}(q, \dot{q}) = -\frac{\partial \mathcal{F}}{\partial \dot{q}}(\dot{q}) + A(q)\lambda + \mathcal{U}_q \quad (1.31)$$

en donde  $\dot{q}$  es el vector de corrientes del circuito,  $\lambda$  es el multiplicador de Lagrange, y

$$A(q)^T(\dot{q}) = 0 \quad (1.32)$$

es la forma matricial de la aplicación de la Ley de Corrientes de Kirchhoff en los nodos del circuito.

#### **Ejemplo 1.4: Modelado basado en el lagrangiano.**

---

Con respecto al circuito de la figura 1.1, podemos aplicar el método variacional basado en el lagrangiano. Para ello definimos las coordenadas  $q$ :

$$\dot{q} = [\dot{q}_1 \ \dot{q}_2 \ \dot{q}_3]^T = [i_{L_1} \ i_{L_2} \ i_c]^T \quad (1.33)$$

La coenergía magnética es

$$\mathcal{T}_{\mathcal{M}}(q, \dot{q}) = \frac{1}{2}L_1 i_{L_1}^2 + \frac{1}{2}L_2 i_{L_2}^2 = \frac{L_1 \dot{q}_1^2}{2} + \frac{L_2 \dot{q}_2^2}{2} \quad (1.34)$$

La energía de campo eléctrico es

$$\mathcal{V}_{\mathcal{E}}(q) = \frac{q_3^2}{2C} \quad (1.35)$$

Al reemplazar en 1.26, se obtiene:

$$\mathcal{L}(q, \dot{q}) = \frac{L_1 \dot{q}_1^2}{2} + \frac{L_2 \dot{q}_2^2}{2} - \frac{q_3^2}{2C} \quad (1.36)$$

Para escribir la ecuación 1.31 calculamos primero:

$$\frac{\partial \mathcal{L}}{\partial q}(q, \dot{q}) = \begin{cases} \frac{\partial \mathcal{L}}{\partial q_1}(q, \dot{q}) = 0 \\ \frac{\partial \mathcal{L}}{\partial q_2}(q, \dot{q}) = 0 \\ \frac{\partial \mathcal{L}}{\partial q_3}(q, \dot{q}) = -\frac{q_3}{C} \end{cases} \quad (1.37)$$

También calculamos:

$$\frac{\partial \mathcal{L}}{\partial \dot{q}}(q, \dot{q}) = \begin{cases} \frac{\partial \mathcal{L}}{\partial \dot{q}_1}(q, \dot{q}) = L_1 \dot{q}_1 \\ \frac{\partial \mathcal{L}}{\partial \dot{q}_2}(q, \dot{q}) = L_2 \dot{q}_2 \\ \frac{\partial \mathcal{L}}{\partial \dot{q}_3}(q, \dot{q}) = 0 \end{cases} \therefore \frac{d}{dt} \left( \frac{\partial \mathcal{L}}{\partial \dot{q}}(q, \dot{q}) \right) = \begin{cases} \frac{d}{dt} \frac{\partial \mathcal{L}}{\partial \dot{q}_1}(q, \dot{q}) = L_1 \ddot{q}_1 \\ \frac{d}{dt} \frac{\partial \mathcal{L}}{\partial \dot{q}_2}(q, \dot{q}) = L_2 \ddot{q}_2 \\ \frac{d}{dt} \frac{\partial \mathcal{L}}{\partial \dot{q}_3}(q, \dot{q}) = 0 \end{cases} \quad (1.38)$$

Como no hay disipación en el circuito, entonces  $\mathcal{F}(q) = 0$ . Por otra parte, no hay perturbaciones y solo aparece una fuerza generalizada, dada por la fuente de tensión:

$$\mathcal{U}_{q_1} = E \quad \mathcal{U}_{q_2} = 0 \quad \mathcal{U}_{q_3} = 0 \quad (1.39)$$

mientras que las restricciones están dadas por las leyes de Kirchhoff:

$$i_{L_1} - i_{L_2} - i_c = 0 \quad \therefore \quad \dot{q}_1 - \dot{q}_2 - \dot{q}_3 = 0 \quad (1.40)$$

La ecuación de restricciones 1.32 se define entonces con

$$A(q) = [1 \ -1 \ -1]^T \quad (1.41)$$

y la ecuación 1.31 resulta ser:

$$\begin{pmatrix} L_1 \ddot{q}_1 \\ L_2 \ddot{q}_2 \\ 0 \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ -q_3/C \end{pmatrix} = - \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ -1 \\ -1 \end{pmatrix} \lambda + \begin{pmatrix} E \\ 0 \\ 0 \end{pmatrix} \quad (1.42)$$

Es decir, las ecuaciones 1.31 y 1.32 se expresan mediante el siguiente sistema de ecuaciones:

$$\begin{cases} L_1 \ddot{q}_1 = \lambda + E \\ L_2 \ddot{q}_2 = -\lambda \\ \frac{q_3}{C} = -\lambda \\ \dot{q}_1 - \dot{q}_2 - \dot{q}_3 = 0 \end{cases} \quad (1.43)$$

Para eliminar  $\lambda$  lo despejamos de la tercera ecuación, y el valor obtenido lo remplazamos en las dos primeras:

$$\begin{cases} L_1 \ddot{q}_1 = -\frac{q_3}{C} + E \\ L_2 \ddot{q}_2 = \frac{q_3}{C} \\ \dot{q}_1 - \dot{q}_2 - \dot{q}_3 = 0 \end{cases} \quad (1.44)$$

Si definimos  $x_1 = \dot{q}_1$ ,  $x_2 = \dot{q}_2$ ,  $x_3 = q_3/C$  el sistema se convierte en

$$\begin{cases} \dot{x}_1 = -\frac{x_3}{L_1} + \frac{E}{L_1} \\ \dot{x}_2 = \frac{x_3}{L_2} \\ \dot{x}_3 = \frac{x_1}{C} - \frac{x_2}{C} \end{cases} \quad (1.45)$$

que en forma matricial resulta ser la misma ecuación 1.20:

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} 0 & 0 & -1/L_1 \\ 0 & 0 & 1/L_2 \\ 1/C & -1/C & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} 1/L_1 \\ 0 \\ 0 \end{pmatrix} E \quad (1.46)$$

■

#### 1.2.4 Diagramas de bloques y diagramas de flujo de señal

En [MB12] se clasifican las técnicas de modelado de sistemas dinámicos en dos tipos:

- Técnicas de flujo de señal.
- Técnicas de interacción física.

Los diagramas de bloques y los diagramas de flujo de señal corresponden al primer tipo. La característica fundamental de esta técnica consiste en modelar la interacción del sistema con su entorno a través de *señales*. El sistema es estimulado por el entorno a través de *señales de entrada*, reacciona a ese estímulo y genera *señales de salida*.

En general, si se denotan con  $\mathbf{y}$  las señales de salida de un sistema y con  $\mathbf{u}$  las de entrada, el modelo matemático del sistema será:

$$\mathbf{y} = \mathcal{F}(\mathbf{u}) \quad (1.47)$$

en donde  $\mathcal{F}$  es un operador genérico.

Un aspecto para resaltar: la señal de salida del sistema es la misma sin importar qué elemento del entorno reciba esa señal. Empleando el lenguaje del análisis de circuitos, los modelos de flujo de señal son modelos de impedancias ideales: la impedancia vista por el sistema desde sus entradas es cero, mientras que la impedancia vista por el sistema desde sus salidas es infinita.

¿El mundo físico se comporta así? La respuesta a esta pregunta no compete al modelado. En su lugar deberíamos formular otra más adecuada, como por ejemplo: ¿bajo qué condiciones un modelo de flujo de señal es útil? En particular, varios dispositivos de procesamiento de señales se diseñan con el propósito explícito de que su respuesta (su salida) no dependa del elemento que se conecte a sus salidas. En tales condiciones, un modelo de flujo de señal puede ser adecuado. El ejemplo 1.5 ilustra este aspecto de los modelos de flujo de señal.

Si el modelo de un sistema dinámico es lineal e invariante en el tiempo, entonces la ecuación 1.47, con condiciones iniciales nulas, puede convertirse en:

$$\mathbf{y}(s) = \mathbf{F}(s)\mathbf{u}(s) \quad (1.48)$$

en donde  $\mathbf{F}(s)$  es la función de transferencia del sistema, y los vectores  $\mathbf{u}(s)$ ,  $\mathbf{y}(s)$  son los vectores de las transformadas de Laplace de  $\mathbf{u}(t)$ ,  $\mathbf{y}(t)$ . En estas circunstancias, además, es posible utilizar herramientas tales como el álgebra de bloques y la regla de Mason para analizar diagramas complejos (véanse [Che93, Dua05], por ejemplo).

### *Ejemplo 1.5: Modelado con diagramas de bloques.*

---

Los resultados de los ejemplos 1.1, 1.2 y 1.3 nos permiten interpretar el comportamiento del circuito de la figura 1.1 como un dispositivo que convierte una tensión constante  $E$  en señales eléctricas sinusoidales. En particular, la tensión entre los terminales del capacitor  $v_c$  puede obtenerse a partir de

$$v_c(t) = \frac{1}{C} \int_0^t i_c(t) + k \quad (1.49)$$

$i_c$  es la misma  $i_3$  en la ecuación 1.6:

$$v_c(t) = -\frac{1}{2} \cos(\sqrt{2}t) + k \quad (1.50)$$

El valor de  $k$  se obtiene al evaluar la condición inicial:

$$v_c(0) = 0 = -\frac{1}{2} + k \quad k = \frac{1}{2} \quad (1.51)$$

Por lo tanto, la tensión entre los terminales del condensador esta dada por

$$v_c(t) = \frac{1}{2} \left( \cos(\sqrt{2}t) + 1 \right) \quad (1.52)$$

Se puede definir una función de transferencia para la salida  $v_c$ , empleando las ecuaciones de estado 1.20 y 1.21:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{Ax}(t) + \mathbf{Bu}(t) \\ \mathbf{y}(t) &= \mathbf{Cx}(t) + \mathbf{Du}(t) \end{aligned} \quad (1.53)$$

La función de transferencia se calcula como:

$$\mathbf{F}(s) = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D} \quad (1.54)$$

Calculamos inicialmente:

$$s\mathbf{I} - \mathbf{A} = \begin{pmatrix} s & 0 & 1/L_1 \\ 0 & s & -1/L_2 \\ -1/C & 1/C & s \end{pmatrix} \det(s\mathbf{I} - \mathbf{A}) = s \left( s^2 + \frac{1}{L_1 C} + \frac{1}{L_2 C} \right) \quad (1.55)$$

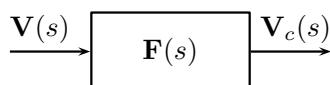
y por tanto:

$$(s\mathbf{I} - \mathbf{A})^{-1} = \frac{1}{s \left( s^2 + \frac{1}{L_1 C} + \frac{1}{L_2 C} \right)} \begin{pmatrix} s^2 + 1/L_2 C & 1/L_1 C & -s/L_1 \\ 1/L_2 C & s^2 + 1/L_1 C & s/L_2 \\ s/C & -s/C & s^2 \end{pmatrix} \quad (1.56)$$

Al utilizar  $\mathbf{y} = v_c$  la matriz  $\mathbf{C}$  debe ser  $\mathbf{C} = [0 \ 0 \ 1]$  y por tanto:

$$\mathbf{F}(s) = \frac{1/L_1 C}{s^2 + \frac{1}{L_1 C} + \frac{1}{L_2 C}} \quad (1.57)$$

En estas condiciones, podemos visualizar el circuito de la figura 1.1 mediante el siguiente bloque



Se ha empleado la herramienta xcos (de scicos) para modelar este diagrama de bloques. El montaje en xcos y el resultado de la simulación son los siguientes:

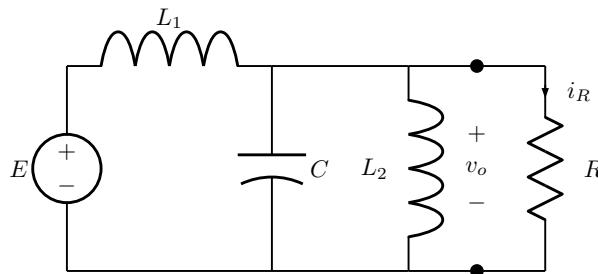
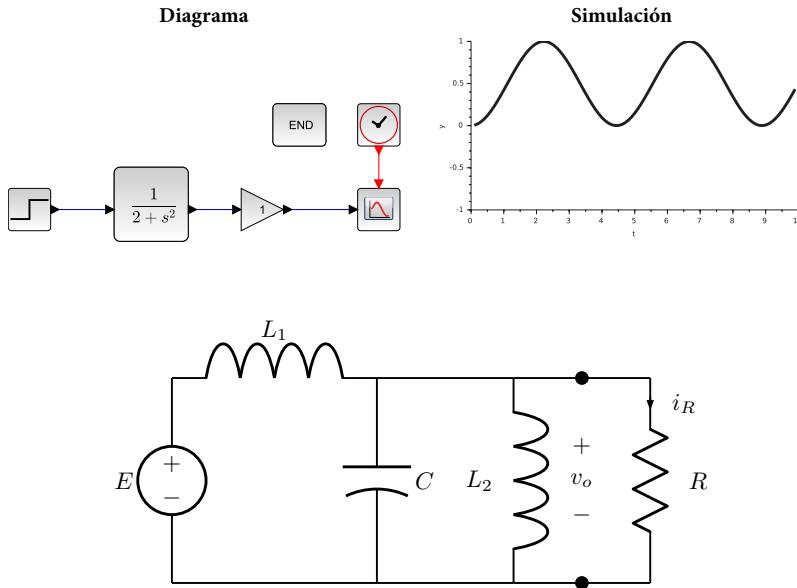
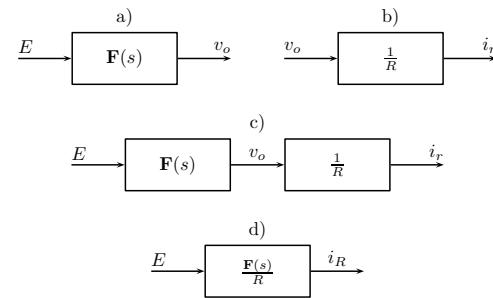
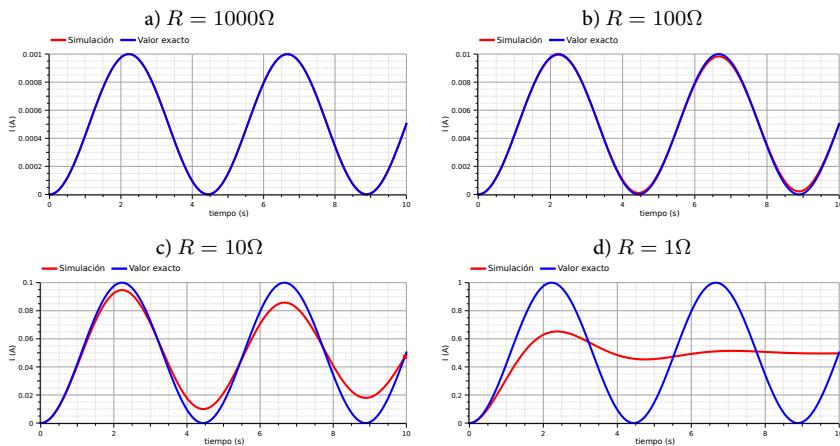


Figura 1.2 Circuito de los ejemplos 1.5 y 1.7

¿Qué sucede si ahora conectamos un resistor entre los terminales del capacitor, tal como muestra la figura 1.2? Para conocer la corriente que circula por el resistor podemos modelar este mediante  $i_R = R/v_R$ , e interconectar los dos sistemas como se muestra a continuación:



Se ha graficado el resultado de la simulación (en azul) junto con el resultado de un análisis exacto del circuito (en rojo), para cuatro valores diferentes de  $R$ :



Puede observarse que para valores “grandes” de  $R$  el modelo de flujo de señal es bastante exacto, mientras que para valores “pequeños” de  $R$  el error cometido es inadmisible. Al variar  $R$ , se varía el valor de la impedancia vista por el circuito original: para valores “grandes”, la aproximación de “impedancia infinita” funciona bastante bien.

■

### 1.2.5 Grafos de enlaces de potencia

Un grafo de enlace de potencia es una representación gráfica del flujo de potencia en un sistema. El formalismo con el que se construyen los grafos permite extraer un modelo matemático del sistema a partir del grafo. Este formalismo también permite sistematizar el proceso de construcción del modelo, y por tanto es posible desarrollar herramientas de *software* que, a partir del grafo, construyan un modelo matemático y lo simulen directamente. En [Bor10] se hace una presentación detallada y extensa de la técnica de grafos de enlace de potencia. En la presente sección solo se presentan los fundamentos mínimos de ella.

En cada uno de los elementos del grafo se definen dos variables: *esfuerzo*  $e$  y *flujo*  $f$ . Para el modelado de sistemas físicos, la selección de estas variables debe ser tal, que su producto sea la potencia  $p$  desarrollada en ese elemento:

$$p(t) = e(t)f(t) \quad (1.58)$$

Los elementos con los que se construye uno grafo de enlace de potencia son de tres tipos (véase la figura 1.3):

GF.1. Elementos de un puerto: intervienen una variable de esfuerzo y una variable de flujo. Los principales casos son:

- Elemento  $R$ : el flujo es directamente proporcional al esfuerzo:

$$e(t) = Rf(t) \quad f(t) = \frac{e(t)}{R} \quad (1.59)$$

- Elemento  $I$ : el esfuerzo es directamente proporcional a la variación de flujo:

$$e(t) = I \frac{df(t)}{dt} \quad f(t) = \frac{1}{I} \int_0^t e(t) dt + f(0) \quad (1.60)$$

- Elemento  $C$ : el flujo es directamente proporcional a la variación de esfuerzo:

$$f(t) = C \frac{de(t)}{dt} \quad e(t) = \frac{1}{C} \int_0^t f(t) dt + e(0) \quad (1.61)$$

- Fuente de esfuerzo  $S_e$ : el esfuerzo es conocido e independiente del flujo:

$$e(t) = S_e \quad (1.62)$$

- Fuente de flujo  $S_f$ : el flujo es conocido e independiente del esfuerzo:

$$f(t) = S_f \quad (1.63)$$

GF.2. Elementos de dos puerto: elementos que relacionan dos parejas de variables esfuerzo-flujo de igual potencia. Existen dos casos:

- Transformadores: el esfuerzo de un puerto es directamente proporcional al esfuerzo del otro puerto:

$$e_1(t) = k e_2(t) \quad f_1(t) = \frac{f_2(t)}{k} \quad p_1(t) = p_2(t) \quad (1.64)$$

- Giradores: el esfuerzo de un puerto es directamente proporcional al flujo del otro puerto:

$$e_1(t) = k f_2(t) \quad f_1(t) = \frac{e_2(t)}{k} \quad p_1(t) = p_2(t) \quad (1.65)$$

GF.3. Uniones (elementos multipuerto): elementos de acoplos entre otros elementos del grafo. Algunos de los acoplos pueden considerarse de entrada (suministran potencia) y otros de salida. La relación básica es la continuidad de potencia:

$$p_{\text{entrada}}(t) = p_{\text{salida}}(t) \quad \sum_{\text{entradas}} e_i(t)f_i(t) = \sum_{\text{salidas}} e_i(t)f_i(t) \quad (1.66)$$

Existen dos casos especiales:

- Uniones tipo 0: los elementos acoplados comparten el mismo esfuerzo:

$$e_1(t) = e_2(t) = \dots = e_n(t) \quad \sum_{\text{entradas}} f_i(t) = \sum_{\text{salidas}} f_i(t) \quad (1.67)$$

- Uniones tipo 1: los elementos acoplados comparten el mismo flujo:

$$f_1(t) = f_2(t) = \dots = f_n(t) \quad \sum_{\text{entradas}} e_i(t) = \sum_{\text{salidas}} e_i(t) \quad (1.68)$$

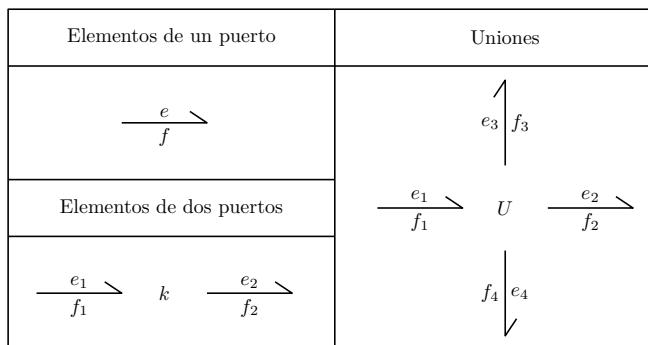


Figura 1.3 Elementos de un grafo de enlace de potencia

En términos generales, el proceso de modelado con grafos de enlace contempla tres etapas:

BG.1. Construcción del grafo: la estrategia básica consiste en descomponer el sistema en sus elementos más simples, y representarlos por alguno

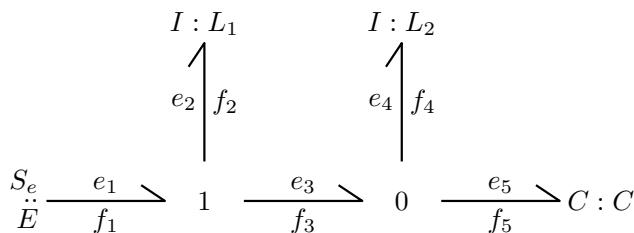
de los elementos básicos anteriormente enumerados. Además, existen procedimientos específicos para ciertos dominios físicos. En [Bor10] (sección 2.7) se explican algunos de estos procedimientos para sistemas mecánicos, eléctricos, magnéticos, hidráulicos y térmicos.

- BG.2. Asignación de causalidades: consiste en determinar para cada elemento del grafo cuál de las dos variables (esfuerzo y flujo) será calculada a partir de la otra. Se trata, entonces, de una causalidad computacional que no está relacionada con la causalidad física del sistema. Existen procedimientos explícitos para “causalizar” el grafo (véase el capítulo 4 de [Bor10]).
- BG.3. Extracción de ecuaciones: una vez asignadas las causalidades, es elemental extraer de cada enlace las relaciones matemáticas individuales. Es posible procesar manualmente estas relaciones para obtener modelos de espacio de estado del sistema. También es posible entregar el listado extenso de ecuaciones a algoritmos que las procesen y simulen (véase capítulo 2).

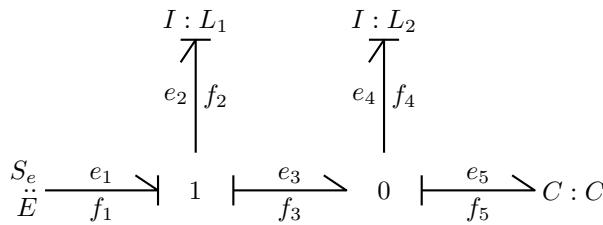
#### *Ejemplo 1.6: Modelado con grafos de enlace de potencia.*

---

En sistemas eléctricos, las variables de esfuerzo y flujo que se utilizan son la diferencia de potencial  $v(t)$  y la corriente  $i(t)$ , que satisfacen  $p(t) = v(t)i(t)$ . El circuito de la figura 1.1 puede representarse por el siguiente grafo de enlace de potencia:



Nótese que la conexión serie ha sido modelada por una unión tipo 1, en tanto que la conexión paralelo lo ha sido por una unión tipo 0. El árbol causalizado se muestra a continuación:



Las pequeñas marcas perpendiculares distinguen los dos tipos de causalidad:

- Salida de esfuerzo: la marca está en la cabeza del enlace (como en los elementos tipo  $I$  de este ejemplo) y significa que la variable que será calculada es el esfuerzo.
- Salida de flujo: la marca está en la cola del enlace (como en el elemento tipo  $C$  de este ejemplo) y significa que la variable que será calculada es el flujo.

Es posible entonces obtener las siguientes ecuaciones básicas:

- Ecuaciones de los puertos:

$$e_1 = 0 \quad e_2 = L_1 \dot{f}_2 \quad e_4 = L_2 \dot{f}_4 \quad f_5 = C \dot{e}_5 \quad (1.69)$$

- Ecuaciones de las uniones:

$$f_1 = f_2 = f_3 \quad e_1 = e_2 + e_3 \quad e_3 = e_4 = e_5 \quad f_3 = f_4 + f_5 \quad (1.70)$$

Si seleccionamos las variables que están derivadas ( $f_2, f_4, e_5$ ) como variables de estado, podemos procesar las relaciones simples para desarrollar un modelo de estado. Para ello, buscamos las ecuaciones correspondientes a las derivadas de las variables de estado:

- La ecuación diferencial para la derivada de  $f_2$ :

$$\dot{f}_2 = \frac{e_2}{L_1} = \frac{e_1 - e_3}{L_1} = \frac{E - e_5}{L_1} \quad (1.71)$$

- La ecuación diferencial para la derivada de  $f_4$ :

$$\dot{f}_4 = \frac{e_4}{L_2} = \frac{e_5}{L_2} \quad (1.72)$$

- La ecuación diferencial para la derivada de  $e_5$ :

$$\dot{e}_5 = \frac{f_5}{C} = \frac{f_3 - f_4}{L_2} = \frac{f_2 - f_4}{L_2} \quad (1.73)$$

Estas tres ecuaciones se pueden escribir en forma matricial así:

$$\begin{pmatrix} \dot{f}_2 \\ \dot{f}_4 \\ \dot{e}_5 \end{pmatrix} = \begin{pmatrix} 0 & 0 & -1/L_1 \\ 0 & 0 & 1/L_2 \\ 1/C & -1/C & 0 \end{pmatrix} \begin{pmatrix} f_2 \\ f_4 \\ e_5 \end{pmatrix} + \begin{pmatrix} 1/L_1 \\ 0 \\ 0 \end{pmatrix} E \quad (1.74)$$

Esta ecuación coincide con la ecuación 1.20, haciendo  $x_1 = f_2 = i_{L_1}$ ,  $x_2 = f_4 = i_{L_2}$ ,  $x_3 = e_5 = v_c$ .

■

### 1.2.6 Modelado orientado a objetos

El enfoque de la programación orientada a objetos (OOP, sigla según el nombre en inglés) tiene sus raíces en los problemas de simulación<sup>9</sup>. No obstante, el desarrollo de un paradigma de *modelado* orientado a objetos (OOM) es posterior al paradigma de *programación* orientada a objetos. En [Elm78], Elmqvist construye las bases del modelado orientado a objetos, como una forma de superar las limitaciones de los lenguajes de simulación basados en los diagramas de bloques, y en particular para evitar los errores de modelado que pueden surgir con ese enfoque (véase el ejemplo 1.5).

En [CEOL95] y en [Bor10] se presentan las características mínimas del paradigma OOM, la gran mayoría de las cuales también están presentes en el paradigma OOP:

**OOM.1.** Instanciación de objetos: una *clase* es una definición genérica de un modelo. Esta definición se vuelve específica cuando se crea un *objeto* de esa clase. En otras palabras, un objeto es cada *instancia* que se crea de la clase. De esta forma, es posible reutilizar el modelo que reside en la clase en múltiples instancias.

---

<sup>9</sup>El primer lenguaje de programación orientado a objetos fue Simula 67. En su presentación se argumentaba la necesidad de un nuevo enfoque que permitiera la “organización e implementación de programas muy complejos y altamente interactivos, tales como los grandes programas de simulación” [DMN70]. De hecho, uno de los primeros ejemplos en ese mismo documento consiste en la implementación de una clase dedicada a la simulación de eventos discretos. Una revisión histórica del paradigma OOP se encuentra en [Cap03].

- OOM.2. Encapsulamiento del conocimiento: la información de cada objeto está encapsulada, lo que significa que un objeto guarda su información internamente sin mezclarse con la información de otros objetos. Esta característica protege la información propia de cada objeto, de tal manera que en el diseño de las clases no es necesario preocuparse por los efectos que otros objetos (quizás aún no diseñados) puedan llegar a tener accidentalmente.
- OOM.3. Capacidades de interconexión topológicas: los objetos pueden interconectarse para ensamblar modelos más sofisticados. Los mecanismos de conexión deben ser tales que permitan un diseño tipo *plug and play* a nivel de *software*.
- OOM.4. Capacidades de construcción de redes generalizadas: uno de los mecanismos de interconexión disponibles son *nodos* que permiten conectar un número arbitrario de elementos y, de esta forma, construir redes complejas de objetos.
- OOM.5. Modelado jerárquico: un modelo sofisticado puede contener modelos más simples en su interior. Esta característica busca que el modelo pueda reflejar las estructuras reales que existen en el sistema modelado.
- OOM.6. Herencia de clases: mediante este mecanismo es posible definir *clases especializadas* a partir de una o más *clases base*. La herencia facilita la organización, reutilización y mantenimiento del código, así como la construcción de librerías estructuradas.
- OOM.7. Polimorfismo: esta característica permite la definición de interfaces genéricas para conectar elementos de diferentes tipos, lo que facilita la construcción de clases genéricas, cuyo comportamiento se hace explícito en diferentes clases especializadas.

La utilización de estas características con propósitos de modelado de sistemas dinámicos se resume en [Fri14] así:

- “La orientación a objetos se usa principalmente como un concepto *estructurador*, enfatizando la estructura declarativa y la reutilización de modelos matemáticos. Nuestras tres formas de estructurar son las jerarquías, la conexión de componentes y la herencia”.
- “Las propiedades de los modelos dinámicos se expresan de forma declarativa a través de *ecuaciones*”.

- “Un objeto es una colección de variables de instancia y ecuaciones que comparten un conjunto de datos.”

También en [Fri14] se propone una metodología de construcción de modelos orientados a objetos mediante la siguiente estrategia:

- OO.1. Definir brevemente el sistema.
- OO.2. Descomponer el sistema en sus componentes principales y diagramar las clases del modelo o utilizar clases predefinidas en librerías disponibles.
- OO.3. Definir la comunicación, es decir, las vías de interacción entre los componentes.
- OO.4. Definir las interfaces, es decir, los conectores externos de cada componente. Mediante estos conectores, cada componente se comunicará con otros componentes. Si es necesario, se deben definir nuevas clases de conectores que permitan un alto grado de reutilización y conectividad, pero permitiendo un grado de revisión de tipo adecuado.
- OO.5. Descomponer recursivamente los componentes del modelo. En cada iteración se aplica la metodología desde el paso OO.2, lo que conlleva la disminución progresiva de la complejidad de los componentes y conectores. El proceso se lleva a cabo hasta que todos los modelos sean definidos, o bien como instancias de clases o tipos predefinidos, o bien por nuevas clases definidas por el usuario.
- OO.6. Formular nuevas clases de modelo donde sea necesario, tanto clases base como derivadas:
  - Declarar nuevas clases de modelo para todos los componentes que no sean instancias de clases predefinidas.
  - Declarar las posibles clases base para incrementar la reutilización y la mantenibilidad de las clases.

En este procedimiento, el paso OO.4 es el más difícil, sin lugar a dudas. La definición de los mecanismos de interconexión requiere no solo un conocimiento profundo de los fenómenos que se modelarán, sino también una buena experiencia como modelador. Un mecanismo de interconexión bien diseñado debe:

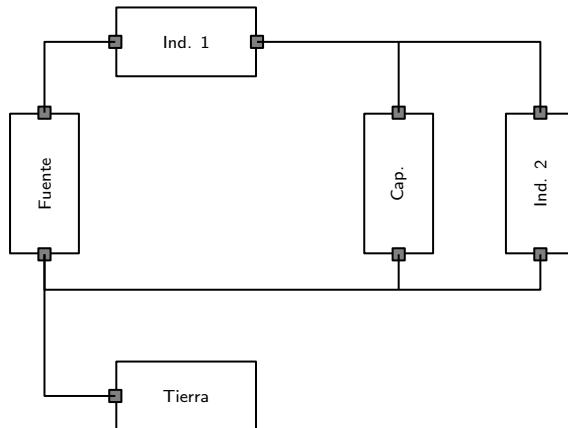
- Permitir una conexión fácil y natural de los objetos. En particular, para sistemas físicos, debe reflejar la forma como estos sistemas se conectan realmente.
- Facilitar la reutilización del modelo en condiciones muy generales. Debe evitarse el diseño de conectores *ad hoc*, es decir, aquellos que solo funcionan bajo condiciones muy específicas.

Es claro que el OOM por sí solo no es suficiente: se necesitan herramientas específicas de *software* que procesen los modelos para generar y simular los modelos matemáticos explícitos. En estos ambientes de modelado y simulación, tales tareas pueden ser más o menos transparentes para el modelador. Además, como consecuencia de las características del OOM, dichos ambientes suelen ofrecer librerías de modelos multidominio prediseñadas, que facilitan la tarea del modelador. Estas librerías pueden incorporar tanto modelos continuos, como discretos e híbridos.

### **Ejemplo 1.7: Modelado orientado a objetos.**

---

Para obtener un modelo orientado a objetos del circuito de la figura 1.1, primero se identifican los componentes interconectados, tal como se muestra a continuación:



Para construir el modelo se ha utilizado el lenguaje Modelica (véase capítulo 3), apoyados en las definiciones disponibles en la Modelica Standard Library (véase sección 3.3). El código fuente del modelo es el siguiente:

## Archivo 1.1 LLC/LLC.mo

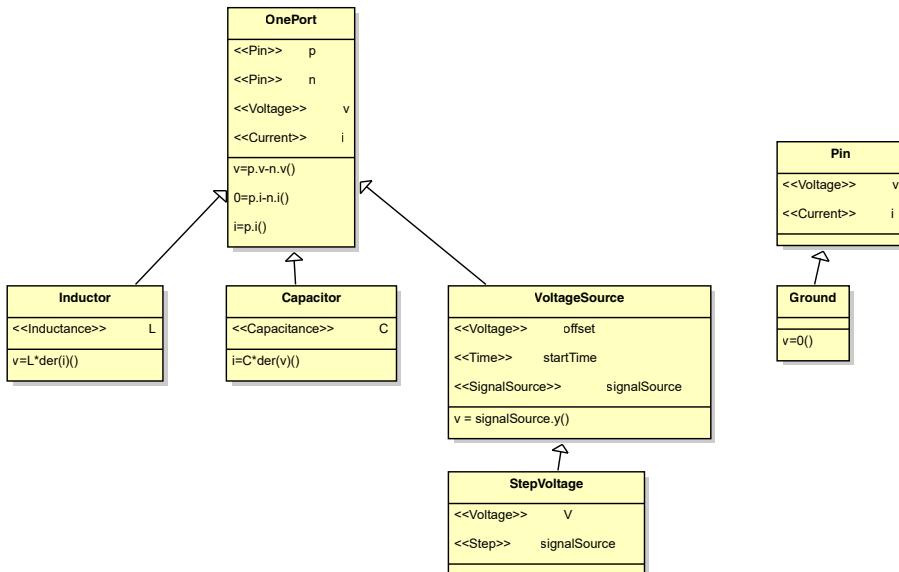
```

model LLC
  Modelica.Electrical.Analog.Basic.Inductor L1;
  Modelica.Electrical.Analog.Basic.Inductor L2;
  Modelica.Electrical.Analog.Basic.Capacitor C;
  Modelica.Electrical.Analog.Basic.Ground G;
  Modelica.Electrical.Analog.Sources.StepVoltage E;

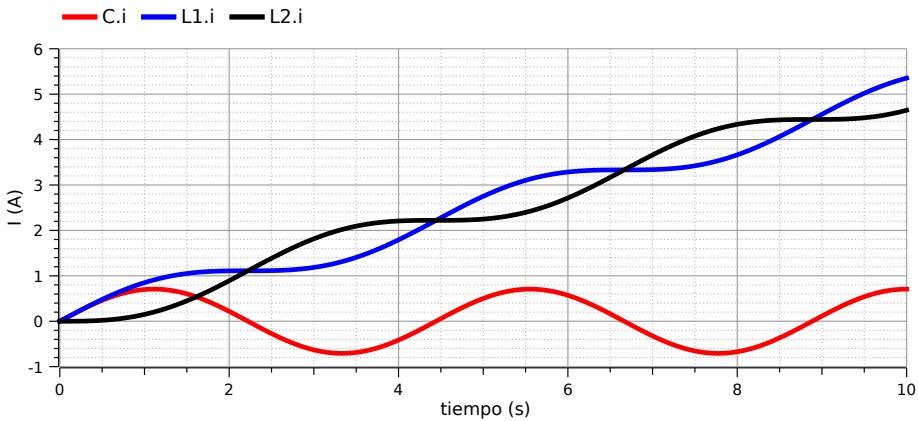
equation
  connect(E.p,L1.p);
  connect(L1.n,C.p);
  connect(C.p,L2.p);
  connect(C.n,L2.n);
  connect(E.n,G.p);
end LLC;

```

El siguiente diagrama muestra las relaciones jerárquicas entre las clases pre-definidas que se han usado en la construcción del modelo. Puede observarse que la clase OnePort funciona como clase base para las clases especializadas Inductor, Capacitor y Voltage Source:



Con la ayuda del ambiente de simulación OMEedit, que forma parte de la suite OpenModelica (ver capítulo 4 y sección 4.7) se ha simulado el modelo. El resultado se presenta a continuación:



Para lograr esa simulación, la herramienta de software genera en un paso intermedio un conjunto de relaciones matemáticas. Estas relaciones se construyen sobre las definiciones de variables y parámetros predefinidos en la librería, que para este ejemplo resultan ser los que se muestran en las tablas 1.1 y 1.2. Las relaciones matemáticas construidas por la herramienta de software son las siguientes:

```

L1.L * der(L1.i) = L1.v;
L1.v = L1.p.v - L1.n.v;
0.0 = L1.p.i + L1.n.i;
L1.i = L1.p.i;
L2.L * der(L2.i) = L2.v;
L2.v = L2.p.v - L2.n.v;
0.0 = L2.p.i + L2.n.i;
L2.i = L2.p.i;
C.i = C.C * der(C.v);
C.v = C.p.v - C.n.v;
0.0 = C.p.i + C.n.i;
C.i = C.p.i;
G.p.v = 0.0;
E.signalSource.y = E.signalSource.offset +
(if time < E.signalSource.startTime then 0.0
else E.signalSource.height);
E.v = E.signalSource.y;
E.v = E.p.v - E.n.v;
0.0 = E.p.i + E.n.i;
E.i = E.p.i;
L1.p.i + E.p.i = 0.0;
L1.n.i + L2.p.i + C.p.i = 0.0;
L2.n.i + C.n.i + G.p.i + E.n.i = 0.0;
E.p.v = L1.p.v;
C.p.v = L1.n.v;

```

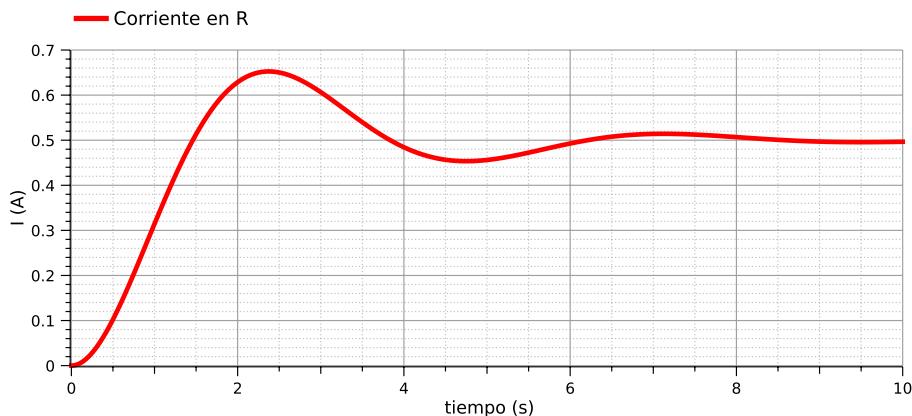
```
C.p.v = L2.p.v;
C.n.v = E.n.v;
C.n.v = G.p.v;
C.n.v = L2.n.v;
```

Para simular el circuito de la figura 1.2 puede hacerse uso de la herencia de clases, con el siguiente código:

Archivo 1.2 LLC/LLCReextends.mo

```
model LLCR
  extends LLC;
  Modelica.Electrical.Analog.Basic.Resistor R(R=1);
equation
  connect(C.p,R.p);
  connect(C.n,R.n);
end LLCR;
```

El resultado de la simulación para  $R = 1\Omega$  se grafica a continuación



### 1.3 COMPARACIÓN DE LAS TÉCNICAS DE MODELADO

Los ejemplos 1.1 a 1.7 muestran la existencia de importantes diferencias entre una técnica de modelado y otra. De esas diferencias podemos destacar:

- a. La robustez de la técnica de modelado, que consiste en su capacidad para generar modelos *correctos*, es decir *sin errores*. Las técnicas más robustas son las diseñadas para dominios específicos, debido a que se han construido dentro de marcos de referencia cerrados que dan un escaso margen de error. En [CEOL95] se identifican tres fuentes de error conceptuales

comunes que pueden generar modelos erróneos. Los errores surgen de la aplicación incorrecta de ciertas técnicas de modelado, frente a lo cual se identifican como las más robustas las técnicas variacionales, los enlaces de potencia y el modelado orientado a objetos. Los errores referidos son<sup>10</sup>:

- Considerar que los modelos de espacio de estado son la base de la física. Debido a la popularidad y utilidad de los modelos de estado, particularmente en el área de la ingeniería de control, es usual intentar descripciones de los fenómenos físicos tan cercanas a los modelos de estado como sea posible. En este intento, es posible obviar inadvertidamente algunos fenómenos físicos relevantes para el modelo.
  - Considerar que las señales capturan el fenómeno físico. La popularidad de los diagramas de bloques, y su gran aplicabilidad en áreas como el procesamiento de señales hace que en ocasiones se olviden los efectos de la interconexión de sistemas, tal como se muestra en el ejemplo 1.5.
  - **Considerar que la causalidad es la base de la física.** La causalidad es en realidad la metáfora que utilizamos para interpretar los fenómenos físicos. Es frecuente confundir la *causalidad física* con la *causalidad computacional*. La naturaleza es en esencia *acausal*, en el sentido de que los fenómenos *suceden* sin que podamos asegurar que uno sea la causa de otro, aunque usualmente los interpretamos como si así fuera. Por su parte, la causalidad computacional determina el orden en que podemos calcular las variables del modelo; este concepto es particularmente relevante en los modelos de grafos de enlace de potencia y en los modelos orientados a objetos.
- b. La eficiencia del modelo generado, que es su capacidad para representar el sistema con pocos recursos. Los modelos generados con técnicas analíticas de dominios específicos son los más eficientes, mientras que los modelos orientados a objetos son los que menos. Esto se ve claramente al comparar los modelos obtenidos en los ejemplos 1.1 y 1.7, ambos sobre el mismo sistema de la figura 1.1: en el primer caso, el modelo consiste en tres ecuaciones diferenciales sobre tres variables y tres parámetros; en el segundo se trata de un conjunto de veintisiete ecuaciones sobre veintisiete variables y quince parámetros.

---

<sup>10</sup>En [CEOL95] se muestran unos ejemplos muy ilustrativos sobre cada uno de estos errores.

Tabla 1.1 Variables del modelo del ejemplo 1.7

Componente						
Nivel 1	Nivel 2	Nivel 3	Nombre	Tipo	Cantidad	Unidad
L1	v	L1.v	Real	Electric Potential	V	Voltage drop between the two pins (p.v - n.v)
	i	L1.i	Real	Electric Current	A	Current flowing from pin p to pin n
	p	L1.pv	Real	Electric Potential	V	Potential at the pin
	i	L1.pi	Real	Electric Current	A	Current flowing into the pin
L2	n	L1.nv	Real	Electric Potential	V	Potential at the pin
	i	L1.ni	Real	Electric Current	A	Current flowing into the pin
	v	L2.v	Real	Electric Potential	V	Voltage drop between the two pins (p.v - n.v)
	i	L2.i	Real	Electric Current	A	Current flowing from pin p to pin n
C	p	L2.pv	Real	Electric Potential	V	Potential at the pin
	i	L2.pi	Real	Electric Current	A	Current flowing into the pin
	n	L2.nv	Real	Electric Potential	V	Potential at the pin
	v	L2.ni	Real	Electric Current	A	Current flowing into the pin
G	i	C.v	Real	Electric Potential	V	Voltage drop between the two pins (p.v - n.v)
	c	C.i	Real	Electric Current	A	Current flowing from pin p to pin n
	p	C.pv	Real	Electric Potential	V	Potential at the pin
	i	C.pi	Real	Electric Current	A	Current flowing into the pin
E	n	C.nv	Real	Electric Potential	V	Potential at the pin
	i	C.ni	Real	Electric Current	A	Current flowing into the pin
	v	G.pv	Real	Electric Potential	V	Potential at the pin
	p	G.pi	Real	Electric Current	A	Current flowing into the pin
signalSource	v	E.v	Real	Electric Potential	V	Voltage drop between the two pins (p.v - n.v)
	i	E.i	Real	Electric Current	A	Current flowing from pin p to pin n
	p	E.pv	Real	Electric Potential	V	Potential at the pin
	i	E.pi	Real	Electric Current	A	Current flowing into the pin
n	v	E.nv	Real	Electric Potential	V	Potential at the pin
	i	E.ni	Real	Electric Current	A	Current flowing into the pin
signalSource		y	E.signalSource.y	Real		Connector of Real output signal

Tabla 1.2 Parámetros del modelo del ejemplo 1.7

Componente		Nombre	Tipo	Cantidad	Unidad	Descripción
Nivel 1	Nivel 2					
L	L	L.L	Real	Inductance	H	Inductance
	IC	L1.IC	Real	Electric Current	A	Initial Value
L1	UIC	L1.UIC	Boolean			Use initial condition value
	L	L2.L	Real	Inductance	H	Inductance
L2	IC	L2.IC	Real	Electric Current	A	Initial Value
	UIC	L2.UIC	Boolean			Use initial condition value
C	C	C.C	Real	Capacitance	F	Capacitance
	IC	C.IC	Real	Electric Potential	V	Initial Value
E	UIC	C.UIC	Boolean			Use initial condition value
	offset	E.offset	Real	Electric Potential	V	Voltage offset
V	startTime	E.startTime	Real	Time	s	Time offset
	V	E.V	Real	Electric Potential	V	Height of step
E	offset	E.signalSource.offset	Real			Offset of output signal y
	signalSource	E.signalSource.startTime	Real	Time	s	Output y = offset for time < startTime
	height	E.signalSource.height	Real			Height of step

La eficiencia de los modelos también puede analizarse desde la facilidad para ser simulados. Las técnicas analíticas, las técnicas de espacio de estado y las técnicas variacionales generan Ecuaciones Diferenciales Ordinarias (ODEs u ODE, sigla según el nombre en inglés) compactas, en contraste con las técnicas orientadas a objetos, que generan Ecuaciones Diferenciales Algebraicas (DAEs o DAE, sigla según el nombre en inglés) dispersas. La simulación de las primeras es más sencilla que la de las segundas. En otras palabras, las exigencias para los métodos numéricos de simulación son mayores en el segundo caso (véase capítulo 2).

- c. La escalabilidad de la técnica, que es su capacidad para generar modelos cuando el tamaño del sistema a modelar, es decir su complejidad, aumenta. Las técnicas analíticas son las menos escalables, debido a que suelen requerir una cantidad importante de trabajo manual. En contraste, el modelado orientado a objetos tiene una de sus fortalezas en este aspecto, lo cual no es de extrañar, ya que justamente ese es uno de sus propósitos: facilitar el modelado de sistemas complejos.
- d. La interpretabilidad de los modelos generados, que está relacionada con nuestra capacidad de análisis de los mismos. Existen dos aspectos que afectan directamente la interpretabilidad: 1) el tamaño del modelo, y 2) la utilización de variables con sentido físico directo. Las técnicas analíticas son, por supuesto, las que generan modelos de mayor interpretabilidad. Los modelos de estado están en un puesto intermedio, mientras que los modelos orientados a objetos son los menos interpretables, sobre todo por su tamaño y el encapsulamiento de la información.
- e. La universalidad de la técnica, que es la capacidad de una técnica para construir modelos de todo tipo. Las técnicas menos universales son, como es de esperar, aquellas definidas para un dominio específico, precisamente porque tienen su campo de acción restringido. En el otro extremo se encuentra la técnica orientada a objetos, no solo por su capacidad para abordar modelos multidominio (capacidad compartida con la técnica de grafos de enlace), sino por la capacidad de modelar eventos discretos, discontinuidades, no-linealidades, y de incorporar el efecto de las condiciones iniciales.

En particular, la habilidad para modelar eventos discretos es uno de los aspectos relevantes de la técnica orientada a objetos. La Especificación de Sistemas de Eventos Discretos (DEVS, sigla según el nombre en inglés) suele emplearse para describir formalmente eventos discretos dentro de

ambientes de simulación (véase [ZPK00], [CK06]). La técnica orientada a objetos permite utilizar ese formalismo de manera transparante para el modelador, a través de las especificaciones del lenguaje de modelado.

La universalidad de una técnica también puede analizarse como su capacidad para incorporar otras técnicas. Por ejemplo, en [Bor10] se muestra cómo obtener modelos variacionales a partir de grafos de enlaces; en este sentido, los grafos de enlaces son más universales que las técnicas variacionales. En [Zim06] se desarrolla una librería orientada a objetos para el diseño de grafos de enlace de potencia que emplean múltiples enlaces simultáneamente, lo que muestra que la orientación a objetos es más universal aún.

- f. La versatilidad de la técnica, entendida como su capacidad para adaptarse a situaciones nuevas. Esta característica va de la mano de la universalidad: cuanto más universal una técnica, más versátil.

#### 1.4 EL MODELADO PARA UN LABORATORIO VIRTUAL

Las diferencias reseñadas en la sección 1.3 hacen previsible que cada técnica de modelado tenga un nicho de aplicación en donde es más adecuada que otras técnicas. Es conveniente considerar la triada técnica-uso-herramienta (T-U-H) para encontrar esos nichos. Algunos de estos son:

- T) Técnicas analíticas, U) interpretación de fenómenos físicos relativamente simples y de un único dominio, y H) rutinas de *software* que implementen métodos de simulación de ODEs genéricos.
- T) Técnicas analíticas de dominios específicos orientadas a la simulación (como la del ejemplo 1.2), U) diseño y simulación de sistemas de un único dominio, y H) en ambientes de simulación propios de ese dominio que implementen métodos de simulación numérica.
- T) Técnicas variacionales en ciertos dominios específicos<sup>11</sup>, U) análisis y simulación, y H) herramientas de *software* específicas.
- T) Técnicas orientadas a objetos, U) análisis, diseño y optimización de sistemas multidominio de gran escala, y H) en ambientes completos de diseño, simulación y optimización tales como el que se muestra en el capítulo 4.

---

<sup>11</sup> Algunos casos son: robótica; modelos de dispersión de gases en la atmósfera; modelos del estado del tiempo.

En este contexto, ¿cuál es la técnica de modelado más adecuada para nuestro laboratorio virtual? En la sección 6.1 se listan las especificaciones funcionales de UNVirtualLab. Entre ellas, destacamos: propósito general, enfoque multidominio, apoyo al aprendizaje, disponibilidad de modelos y licenciamiento abierto.

El propósito general y el enfoque multidominio causan que las técnicas de un dominio específico no sean buenas candidatas. Por su parte, el apoyo al aprendizaje inclina la balanza hacia las técnicas que generan modelos cuya construcción refleja de forma más cercana el sistema real, tales como los grafos de enlace y las técnicas orientadas a objetos. La disponibilidad de modelos y el licenciamiento abierto nos lleva a la búsqueda de herramientas de *software* y librerías de modelos abiertas, punto en que las técnicas de enlace de grafos son particularmente débiles, en tanto que el modelado orientado a objetos es muy fuerte.

Por estas razones, para la construcción de los modelos de UNVirtualLab seleccionamos las técnicas de modelado orientado a objetos; seleccionamos también el uso de lenguajes abiertos, así como de herramientas de software libre y sus librerías también libres (véanse capítulos 3 y 4).



# 2

## SIMULACIÓN NUMÉRICA DE ECUACIONES DIFERENCIALES ALGEBRAICAS



En general, el resultado del proceso de modelado orientado a objetos es un conjunto de ecuaciones simples, algunas de las cuales son diferenciales y otras son algebraicas. El concepto de *Simulación numérica* corresponde a la obtención de la solución de ese conjunto de ecuaciones simultáneas para un conjunto finito de instantes de tiempo. En este capítulo se presentan algunos conceptos básicos para comprender cómo se obtiene dicha simulación. La sección 2.1 destaca algunos aspectos conocidos sobre el comportamiento de las Ecuaciones Diferenciales Algebraicas (DAEs, por su sigla en inglés) que dificultan su simulación. La sección 2.2 muestra el proceso general de simulación. Por su parte, las secciones 2.3, 2.4 y 2.5 profundizan algunos aspectos de dicho proceso. En la sección 2.6 se reseñan algunos problemas relativos al método numérico de simulación empleado.

## 2.1 ANÁLISIS DE DAEs

En términos generales, las DAE son sistemas de ecuaciones que combinan ecuaciones diferenciales con ecuaciones algebraicas. En su expresión más general, tienen la forma

$$\mathbf{F}(t, \mathbf{y}, \dot{\mathbf{y}}) = \mathbf{0} \quad (2.1)$$

en donde  $\mathbf{F}$  es el conjunto de ecuaciones,  $\mathbf{y}$  es el conjunto de variables y  $t$  representa el tiempo. En el contexto de la simulación de sistemas dinámicos, se plantea la restricción en la que el número de ecuaciones es igual al número de variables. En general, algunas de las ecuaciones serán diferenciales y otras serán algebraicas.

Existen algunos casos especiales de DAEs que adquieren ciertas formas definidas en la literatura:

- Forma implícita: corresponde a la ecuación 2.1.
- Forma explícita: el vector  $\mathbf{y}$  se separa para diferenciar las variables que aparecen derivadas ( $x$ ) de las que no lo están ( $y$ ); además, es posible despejar todas las derivadas y todas las variables que no están derivadas:

$$\begin{aligned} \frac{dx}{dt} &= f(x, y, t) \\ y &= g(x, t) \end{aligned} \quad (2.2)$$

- Forma semiexplícita: el vector  $y$  se separa para diferenciar las variables que aparecen derivadas ( $x$ ) de las que no lo están ( $y$ ); además, es posible despejar todas las derivadas:

$$\begin{aligned}\frac{dx}{dt} &= f(x, y, t) \\ 0 &= g(x, y, t)\end{aligned}\tag{2.3}$$

- Forma semiexplícita autónoma: forma semiexplícita en la que la variable tiempo ( $t$ ) no aparece de forma explícita en las funciones:

$$\begin{aligned}\frac{dx}{dt} &= f(x, y) \\ 0 &= g(x, y)\end{aligned}\tag{2.4}$$

- Forma semiexplícita de Hessenberg: forma semiexplícita autónoma en la que las variables derivadas no aparecen en las ecuaciones algebraicas de forma directa:

$$\begin{aligned}\frac{dx}{dt} &= f(x, y) \\ 0 &= g(y)\end{aligned}\tag{2.5}$$

El problema de valor inicial consiste en encontrar una solución de 2.1 que también satisfaga  $\mathbf{y}(t_0) = \mathbf{y}_0$ . En la sección 2.1.1 se muestra que el problema de valor inicial en las DAEs es más sofisticado que su equivalente para Ecuaciones Diferenciales Ordinarias (*Ordinary Differential Equations - ODEs*). En la sección 2.1.2 se estudia la complejidad estructural de las DAEs, mediante el concepto de *índice*.

### 2.1.1 Condiciones iniciales

Para las ODEs, el teorema de existencia y unicidad de la solución establece que es necesario conocer los valores iniciales de todas las variables para poder determinar una única solución. En ese sentido, se pueden escoger libremente las condiciones iniciales. El *problema de valor inicial* consiste en encontrar la solución de la ecuación que satisface las condiciones iniciales.

En el caso de las DAEs, la situación es diferente. Consideraremos una ecuación semiexplícita de la forma:

$$\begin{aligned}\frac{dx}{dt} &= f(x, y, t) \\ 0 &= g(x, y, t)\end{aligned}\tag{2.6}$$

en la que  $x, y$  son vectores de tamaño  $N_x, N_y$  respectivamente. En total hay  $N = N_x + N_y$  ecuaciones, de las cuales  $N_x$  son diferenciales y  $N_y$  son algebraicas. Debido a la presencia de las ecuaciones algebraicas, no todo punto

en  $\mathcal{R}^N$  formará parte de alguna trayectoria solución de 2.6. Pueden identificarse dos tipos de *restricciones* en las posibles soluciones de 2.6:

- Restricciones explícitas: que aparecen de forma evidente en las ecuaciones algebraicas, y corresponden a los puntos de  $\mathcal{R}^N$  que las satisfacen.
- Restricciones ocultas (o implícitas): que pueden aparecer por las relaciones entre las ecuaciones algebraicas y las diferenciales.

Aquellos puntos que satisfacen las restricciones, tanto las explícitas como las ocultas, se denominan *valores iniciales consistentes*. Solo es posible encontrar solución al problema de valor inicial de una DAE, si su valor inicial es consistente.

### **Ejemplo 2.1: Restricciones ocultas.**

---

Se presentan a continuación tres casos de comparación:

1. En primer lugar, tomemos la ODE simple:

$$\dot{x}_1 - 1 = 0 \quad (2.7)$$

cuya solución está determinada de manera única cuando se establece la condición inicial  $x_1(t_0) = x_{10}$  por  $x_1(t) = x_{10}e^t$ . Por esta razón no hay restricción alguna para establecer el valor de  $x_{10}$ .

2. En segundo lugar, considérese la DAE de dos variables:

$$\begin{aligned} \dot{x}_1 &= x_1 \\ 1 &= x_1^2 + x_2^2 \end{aligned} \quad (2.8)$$

La segunda de las ecuaciones establece una *restricción* sobre los valores que pueden tomar  $x_1(t)$  y  $x_2(t)$  en todo instante de tiempo, y en particular para el tiempo inicial  $t_0$ : las condiciones iniciales deben ser un punto de la circunferencia unitaria.

3. En tercer lugar, considérese la DAE con tres variables (tomada de [Sch00])

$$\begin{aligned} \dot{x}_1 &= x_1 \\ \dot{x}_2 &= \frac{x_3}{x_2} \\ 1 &= x_1^2 + x_2^2 \end{aligned} \quad (2.9)$$

La tercera de las ecuaciones implica una restricción semejante a la del numeral anterior. No obstante, como el número de ecuaciones es 3, deberíamos decir que las condiciones iniciales deben estar en el cilindro de radio unitario y eje definido por el vector  $[0, 0, 1]$ . Podemos entonces construir un subconjunto de  $\mathcal{R}^3$  en el que se satisface esta restricción:

$$M_1 = \{(x_1, x_2, x_3) \in \mathcal{R}^3 \mid x_1^2 + x_2^2 = 1\} \quad (2.10)$$

Sin embargo, hay una *restricción oculta* adicional. Para hacerla evidente, solucionamos el sistema:

$$\begin{aligned} x_1(t) &= x_{10}e^t \\ x_2(t) &= (1 - x_{10}^2e^{2t})^{1/2} \\ \dot{x}_2 &= \frac{-2x_{10}^2e^{2t}}{2(1-x_{10}e^{2t})^{1/2}} = \frac{-x_1^2(t)}{x_2(t)} \\ \frac{-x_1^2(t)}{x_2(t)} &= \frac{x_3(t)}{x_2(t)} \end{aligned} \quad (2.11)$$

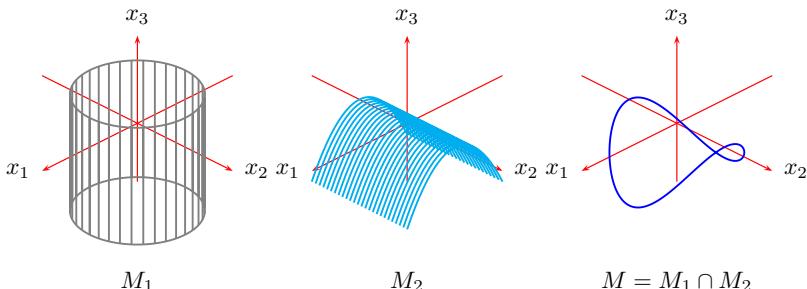
La última de las ecuaciones establece la restricción adicional  $x_3(t) = -x_1^2(t)$  válida para todo instante de tiempo y, por tanto, también para  $t_0$ . El subconjunto de  $\mathcal{R}^3$  en el que se satisface esta restricción es:

$$M_2 = \{(x_1, x_2, x_3) \in \mathcal{R}^3 \mid x_3 = -x_1^2\} \quad (2.12)$$

Las trayectorias de 2.9 deben estar contenidas tanto en  $M_1$  como en  $M_2$ . Por tanto, el subconjunto de puntos  $M$  en el que se alojan las trayectorias está dado por

$$M = M_1 \cap M_2 \quad (2.13)$$

La siguiente figura muestra los subconjuntos  $M_1$ ,  $M_2$  y  $M$ . Destacamos que las condiciones iniciales de 2.9 solo pueden estar en  $M$ .



## 2.1.2 Índices

Un *índice* de una DAE es una medida de la complejidad estructural de la ecuación, o, en otras palabras, es una medida de qué tan diferente es una DAE a una ODE. Existen varias definiciones en la literatura<sup>1</sup> en función de los aspectos que se quieran enfatizar de la estructura de la DAE. En el contexto de este documento utilizaremos la definición de *índice de diferenciación* que aparece en [Sch03], que se enfoca en determinar qué tan difícil puede resultar la simulación numérica de la DAE para los simuladores numéricos de ODEs:

*La DAE 2.1 tiene índice (de diferenciación)  $\mu$  si  $\mu$  es el número mínimo de diferenciaciones*

$$\left\{ \begin{array}{l} \mathbf{F}(t, \mathbf{y}, \dot{\mathbf{y}}) = \mathbf{0} \\ \frac{d\mathbf{F}(t, \mathbf{y}, \dot{\mathbf{y}})}{dt} = \mathbf{0} \\ \vdots \\ \frac{d^\mu \mathbf{F}(t, \mathbf{y}, \dot{\mathbf{y}})}{dt^\mu} = \mathbf{0} \end{array} \right. \quad (2.14)$$

*tales que las ecuaciones 2.14 permiten extraer un sistema de ecuaciones diferenciales explícitas usando sólo manipulaciones algebraicas.*

La importancia del índice de diferenciación radica en que da información acerca de qué tipo de tratamiento simbólico es conveniente realizar sobre la DAE para facilitar la labor del método de simulación seleccionado. De esta definición se desprende que una ODE es una DAE de índice 0.

### Ejemplo 2.2: Índice de diferenciación.

---

Para ilustrar el concepto de índice de diferenciación, supóngase el caso en que las variables y de 2.1 se organizan como  $\mathbf{y} = [x \ y]^T$ , en donde  $x$  corresponde a las variables que están derivadas mientras que  $y$  corresponde a las que no lo están. Además, considérese el caso semieexplícito y autónomo:

$$\begin{aligned} \frac{dx}{dt} &= f(x, y) \\ 0 &= g(x, y) \end{aligned} \quad (2.15)$$

---

<sup>1</sup>Además del índice de diferenciación, en [Sch03] se definen los índices: de Kronecker, de tratabilidad, de perturbación, de rareza y geométrico.

A continuación se analizan unos casos especiales de 2.15:

1. Considérese la situación en la que todas variables se derivan:

$$\frac{dx}{dt} = f(x, y) \quad (2.16)$$

Claramente el sistema corresponde a una ODE, y por tanto el índice es 0. El caso 1 del ejemplo 2.1 se ajusta a 2.16 con  $x = x_1$ .

2. Considérese ahora el sistema de la ecuación 2.15 con variables derivadas y no derivadas. Para intentar convertir el sistema en una ODE, derivamos una vez la segunda igualdad:

$$\begin{aligned} 0 &= \frac{dg(x,y)}{dt} \\ 0 &= \frac{\partial g(x,y)}{\partial x} \frac{dx}{dt} + \frac{\partial g(x,y)}{\partial y} \frac{dy}{dt} \\ 0 &= \frac{\partial g(x,y)}{\partial x} f(x, y) + \frac{\partial g(x,y)}{\partial y} \frac{dy}{dt} \end{aligned} \quad (2.17)$$

Para poder despejar  $\frac{dy}{dt}$  es necesario que el jacobiano  $\frac{\partial g(x,y)}{\partial y}$  sea no singular. En ese caso:

$$\frac{dy}{dt} = -\frac{\partial g(x,y)}{\partial y}^{-1} \frac{\partial g(x,y)}{\partial x} f(x, y) \quad (2.18)$$

En otras palabras, el sistema 2.15 es de índice 1 si  $\frac{\partial g(x,y)}{\partial y}$  es no singular.

El segundo caso del ejemplo 2.1 corresponde a esta situación con  $x = x_1$ ,  $y = x_2$ .

3. Considérese el caso en el que la ecuación algebraica es independiente de las variables no derivadas:

$$\begin{aligned} \frac{dx}{dt} &= f(x, y) \\ 0 &= g(x) \end{aligned} \quad (2.19)$$

En esta situación, el jacobiano  $\frac{\partial g(x,y)}{\partial y}$  es singular y por tanto el sistema no es de índice 1. Usando 2.17 tenemos la primera derivada:

$$0 = \frac{\partial g(x,y)}{\partial x} f(x, y) = h(x, y) \quad (2.20)$$

es decir, podemos construir el sistema auxiliar:

$$\begin{aligned} \frac{dx}{dt} &= f(x, y) \\ 0 &= h(x, y) \end{aligned} \quad (2.21)$$

Por el caso anterior, sabemos que si el jacobiano  $\frac{\partial h(x,y)}{\partial y}$  es no singular, 2.21 es de índice 1, y por tanto 2.19 es de índice 2. En esta situación, la ecuación 2.20 corresponde a la restricción oculta del sistema.

El caso 3 del ejemplo 2.1 corresponde a esta situación con  $x = [x_1 \ x_2]^T$ ,  $y = x_3$ . Por tanto:

$$f(x, y) = \begin{pmatrix} x_1 \\ \frac{x_3}{x_2} \\ x_2 \end{pmatrix} \quad g(x) = (x_1^2 + x_2^2 - 1) \quad \frac{\partial g(x)}{\partial x} = (2x_1 \quad 2x_2) \quad (2.22)$$

La restricción oculta 2.20 es:

$$h(x, y) = \frac{\partial g(x)}{\partial x} f(x, y) = (2x_1 \quad 2x_2) \begin{pmatrix} x_1 \\ \frac{x_3}{x_2} \\ x_2 \end{pmatrix} = 2x_1^2 + 2x_3 = 0 \quad (2.23)$$

es decir, la restricción oculta es  $x_3 = -x_1^2$ , tal como se había calculado en el ejemplo 2.1.

■

## 2.2 EL PROCESO DE SIMULACIÓN

Considérese inicialmente el problema de obtener la simulación numérica de la ODE

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \\ \mathbf{x}(t_0) = \mathbf{x}_0 \end{cases} \quad (2.24)$$

en la que  $\mathbf{x}$  es el vector de estado ( $x_i$  es la  $i$ -ésima variable de estado),  $\mathbf{u}$  es el vector de entradas,  $t$  es el tiempo,  $\mathbf{f}$  es un vector de funciones, y la segunda ecuación establece las condiciones iniciales. Los métodos de simulación numérica se basan en la expansión en series de Taylor de la solución:

$$x_i(\tau + h) = x_i(\tau) + \frac{dx_i(\tau)}{dt} \cdot h + \frac{d^2x_i(\tau)}{dt^2} \cdot \frac{h^2}{2!} + \dots \quad (2.25)$$

Utilizando la nomenclatura  $f_i$  para la  $i$ -ésima componente de  $\mathbf{f}$ , se tiene:

$$x_i(\tau + h) = x_i(\tau) + f_i(\tau) \cdot h + \frac{df_i(\tau)}{dt} \cdot \frac{h^2}{2!} + \dots \quad (2.26)$$

Los distintos métodos de simulación difieren en la forma de calcular los términos de orden  $h^2$  y superior. Por ejemplo, el método de Euler hacia adelante los desprecia:

$$x_i(\tau + h) \approx x_i(\tau) + f_i(\tau) \cdot h \quad (2.27)$$

$$\mathbf{x}(\tau + h) \approx \mathbf{x}(\tau) + \mathbf{f}(\mathbf{x}(\tau), \mathbf{u}(\tau), \mathbf{t}) \cdot h \quad (2.28)$$

El método de Euler hacia adelante tiene la estructura que se muestra en la figura 2.1. El método utiliza iterativamente las ecuaciones de estado para actualizar los valores de  $\dot{\mathbf{x}}$  y con estos, los valores de  $\mathbf{x}$ . Esta estructura, no obstante, es insuficiente para simular numéricamente las DAEs.

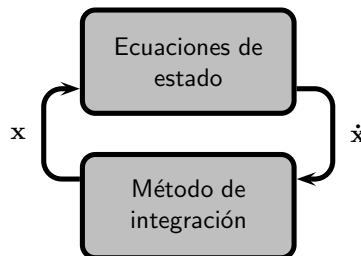


Figura 2.1 Estructura del método de simulación de Euler hacia adelante

### **Ejemplo 2.3: Método de Euler hacia adelante.**

---

Para ilustrar la estructura que se muestra en la figura 2.1, considérese la ODE simple

$$\dot{x}(t) = ax(t) \quad x(0) = x_0;$$

cuya solución analítica es  $x(t) = x_0 e^{at}$

$$u(t) = 0 \quad f(x(t), u(u), t) = ax(t)$$

$$x(\tau + h) \approx x(\tau) + ax(\tau)h$$

La aplicación sucesiva de 2.28 da como resultado:

$t$	$u(t)$	$x(t)$	$f(t) = ax(t)$	$x(t + h)$
0	0	$x_0$	$x_0 a$	$x_0(1 + ah)$
$h$	0	$x_0(1 + ah)$	$x_0 a(1 + ah)$	$x_0(1 + ah)^2$
$2h$	0	$x_0(1 + ah)^2$	$x_0 a(1 + ah)^2$	$x_0(1 + ah)^3$
$\vdots$				

En cada paso de iteración se utiliza la ecuación de estado  $\dot{x}(t) = f(t) = ax(t)$  para actualizar el valor de  $x(t + h)$ .



La figura 2.2 muestra la estructura general de un simulador numérico para DAEs. Sobre la estructura simple de la figura 2.1, se observan varias sofisticaciones:

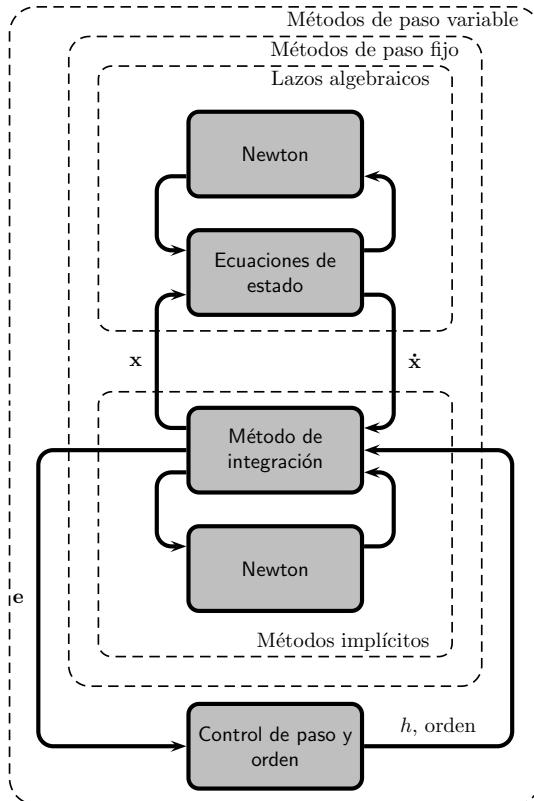


Figura 2.2 Estructura de un simulador numérico para DAEs

- La posible (y usual) presencia de lazos algebraicos no permite calcular de forma explícita  $\dot{x}$  a partir de  $x$ . Si esto sucede, se requiere la aplicación de un método numérico de solución de ecuaciones del tipo Newton.
- Si se utilizan métodos de simulación implícitos, la actualización de los valores de  $x$  en cada iteración requiere también de un método numérico de solución de ecuaciones del tipo Newton. Estos métodos tienen mejores condiciones de estabilidad que los explícitos.

- Si se utilizan métodos que automáticamente ajustan el tamaño de paso y el orden de integración, entonces es necesario incorporar el lazo de control de estas dos variables a partir de la estimación del error. Estos métodos son especialmente útiles para la simulación de problemas *stiff*, bien sea en ODEs o en DAEs.

La figura 2.2 muestra también que las ecuaciones de estado son utilizadas reiterativamente en el proceso de simulación. Por esta razón, es crítico tener una representación numérica eficiente de ellas; en particular, debe buscarse que el ciclo de iteraciones de Newton para la solución de lazos algebraicos y singularidades estructurales sea lo más eficiente posible, o, mejor aún, que no se requiera tal ciclo, es decir, que se tenga una representación explícita.

La sección 2.4 presenta una serie de algoritmos que utilizan representaciones simbólicas de las DAEs para disminuir la complejidad del ciclo de iteraciones de Newton directamente asociado a las ecuaciones de estado. Estos algoritmos son indispensables en la simulación de problemas de alta complejidad en los que el número de ecuaciones es considerable ( $> 100$ ).

Por otra parte, antes de iniciar el proceso de simulación es necesario encontrar un conjunto de condiciones iniciales consistente. La sección 2.3 ilustra este proceso.

### 2.3 CÁLCULO DE CONDICIONES INICIALES

En la sección 2.1.1 se muestra que no todo conjunto de valores iniciales es consistente. Esto significa que en el proceso de simulación de una DAE es necesario disponer de procedimientos para encontrar valores iniciales consistentes. Existen varias propuestas en la literatura. En [Sch00] y en [BHLRP98] se relaciona una buena cantidad de ellas.

No hay un algoritmo universal para la búsqueda de condiciones iniciales consistentes. Tanto la diversidad de DAEs existentes, como la variedad de algoritmos de simulación, hacen que unos de ellos sean más adecuados que otros para problemas específicos. A título ilustrativo, presentamos aquí el algoritmo propuesto en [BHLRP98] para encontrar los valores de  $\mathbf{v}(t_0) = \mathbf{v}_0$  a partir de  $\mathbf{u}(t_0) = \mathbf{u}_0$  en el sistema semiexplícito:

$$\begin{aligned} 0 &= \mathbf{f}(t, \dot{\mathbf{u}}, \mathbf{u}, \mathbf{v}) \\ 0 &= \mathbf{g}(t, \mathbf{u}, \mathbf{v}) \end{aligned} \tag{2.29}$$

Se trata de un esquema iterativo del tipo Newton, en el que se actualiza el valor de  $\bar{\mathbf{u}}_0, \bar{\mathbf{v}}_0$  de la siguiente forma:

$$\begin{pmatrix} \bar{\mathbf{u}}_0 \\ \bar{\mathbf{v}}_0 \end{pmatrix}_{k+1} = \begin{pmatrix} \bar{\mathbf{u}}_0 \\ \bar{\mathbf{v}}_0 \end{pmatrix}_k + \begin{pmatrix} \Delta\bar{\mathbf{u}}_0 \\ \Delta\bar{\mathbf{v}}_0 \end{pmatrix}_k \quad (2.30)$$

La corrección se establece en cada iteración como:

$$\begin{pmatrix} \Delta\bar{\mathbf{u}}_0 \\ \Delta\bar{\mathbf{v}}_0 \end{pmatrix}_k = -\bar{J}^{-1} \begin{pmatrix} \mathbf{f}(t_0, \mathbf{u}_0, \mathbf{v}_0) \\ \mathbf{g}(t_0, \mathbf{u}_0, \mathbf{v}_0) \end{pmatrix}_k \quad (2.31)$$

en donde  $\bar{J}$  es una matriz que se obtiene a partir de los jacobianos de  $\mathbf{f}, \mathbf{g}$ , y un paso de iteración  $h$  así:

$$\bar{J} = \begin{pmatrix} \mathbf{f}_{\dot{\mathbf{u}}} + h\mathbf{f}_{\mathbf{u}} & \mathbf{f}_{\mathbf{v}} \\ h\mathbf{g}_{\mathbf{u}} & \mathbf{g}_{\mathbf{v}} \end{pmatrix} \quad (2.32)$$

En caso de no contar con una expresión explícita de los jacobianos, es necesario obtener una aproximación numérica de ellos. En [BHLRP98] se argumenta que este método es particularmente adecuado en combinación con los métodos numéricos que, de por sí, requieren un jacobiano, dado que no implicaría un esfuerzo de codificación adicional.

#### *Ejemplo 2.4: Condiciones iniciales consistentes.*

---

Considérese la ecuación 2.8 del ejemplo 2.1, que puede reescribirse como:

$$\begin{aligned} 0 &= f(t, \dot{u}, u, v) = \dot{u} - u \\ 0 &= g(t, u, v) = u^2 + v^2 - 1 \end{aligned} \quad (2.33)$$

en donde hemos renombrado  $u = \mathbf{u} = x_1, v = \mathbf{v} = x_2$ . Los jacobianos resultan ser las derivadas parciales simples:

$$\begin{aligned} \mathbf{f}_{\dot{\mathbf{u}}} &= \frac{\partial f}{\partial \dot{u}} = 1 & \mathbf{g}_{\dot{\mathbf{u}}} &= \frac{\partial g}{\partial \dot{u}} = 0 \\ \mathbf{f}_{\mathbf{u}} &= \frac{\partial f}{\partial u} = -1 & \mathbf{g}_{\mathbf{u}} &= \frac{\partial g}{\partial u} = 2u \\ \mathbf{f}_{\mathbf{v}} &= \frac{\partial f}{\partial v} = 0 & \mathbf{g}_{\mathbf{v}} &= \frac{\partial g}{\partial v} = 2v \end{aligned} \quad (2.34)$$

Reemplazando en 2.32 se obtiene la matriz  $\bar{J}$ :

$$\bar{J} = \begin{pmatrix} 1 - h & 0 \\ 2h\bar{u}_0 & 2\bar{v}_0 \end{pmatrix}_k \quad (2.35)$$

Supongamos ahora que el algoritmo se inicia ( $k = 0$ ) con los valores  $\bar{u}_0 = \bar{v}_0 = 0,5$ . Estos valores no son una condición inicial consistente, dado que no satisfacen la restricción algebraica de 2.33, es decir, no está en la circunferencia unitaria. La siguiente tabla muestra la evolución de estos valores en las primeras iteraciones del algoritmo:

	$k = 0$	$k = 1$	$k = 2$	$k = 3$
$\bar{u}_0$	0,5	-0,00010001	0,00000001	$-1,0003 \times 10^{-12}$
$\bar{v}_0$	0,5	1,0	0,9999999995	1,0

El algoritmo converge hacia la pareja  $(u_0, v_0) = (0, 1)$  que sí satisface las restricciones, y por tanto es una condición inicial consistente.

■

## 2.4 ALGORITMOS DE TRATAMIENTO SIMBÓLICO DE DAEs

### 2.4.1 Ordenamiento y asignación de causalidad

Consideremos de nuevo el proceso de simulación de la ecuación 2.1. Independientemente de cuál sea el método de simulación utilizado, este necesitará actualizar en cada iteración unos nuevos valores para un cierto vector de variables  $\mathbf{x}$ , es decir, en cada iteración será necesario solucionar una ecuación de la forma  $\mathbf{G}(\mathbf{x}) = 0$  que en forma explícita se escribe como:

$$\left\{ \begin{array}{l} G_1(x_1, x_2, \dots, x_n) = 0 \\ G_2(x_1, x_2, \dots, x_n) = 0 \\ \vdots \\ G_n(x_1, x_2, \dots, x_n) = 0 \end{array} \right. \quad (2.36)$$

En cada iteración será necesario utilizar cada una de las relaciones  $G_i$  para calcular las variables  $x_i$ . Para resolver 2.36, hay dos posibles enfoques: utilizar un método numérico (Newton, por ejemplo) o intentar un análisis simbólico para despejar las  $x_i$  y calcularlas. El primer enfoque puede ser extremadamente costoso desde el punto de vista computacional, mientras que el segundo no siempre podrá ser exitoso.

El enfoque utilizado para resolver 2.36, en el contexto de la simulación de DAEs, puede resumirse como: *realizar un análisis simbólico hasta donde sea posible, y aplicar un método numérico donde no lo sea*. Es necesario entonces responder la pregunta: ¿hasta dónde será posible realizar un análisis simbólico? o en otras palabras ¿qué variables  $x_i$  podrán despejarse mediante un algoritmo?

Para ilustrar el problema, supóngase el caso en que  $\mathbf{G}(x) = 0$  es de la forma:

$$\begin{cases} G_1(x_1, x_2) = x_1 + x_2 = 0 \\ G_2(x_1, x_2) = x_2 - \sin(t) = 0 \end{cases} \quad (2.37)$$

En un cierto paso de iteración  $k$  será necesario actualizar los valores de  $x_1(k-1)$  y  $x_2(k-1)$ . Si se utilizan las ecuaciones en el mismo orden de la ecuación 2.37, la actualización será la siguiente:

$$\begin{cases} x_1(k) = -x_2(k-1) \\ x_2(k) = \sin(t_k) \end{cases} \quad (2.38)$$

Nótese que la actualización de  $x_1$  se haría con un valor “viejo” de  $x_2$ . En cambio, si alteramos el orden en que utilizamos las ecuaciones, tendremos disponible el valor actualizado de  $x_2$  cuando necesitemos calcular  $x_1$ :

$$\begin{cases} x_2(k) = \sin(t_k) \\ x_1(k) = -x_2(k) \end{cases} \quad (2.39)$$

Este ejemplo pone en evidencia, que es necesario determinar cómo se utilizará cada una de las  $n$  ecuaciones de 2.36 para actualizar los valores de las variables en cada paso de iteración. En forma concreta, es necesario:

1. *Ordenar* las ecuaciones.
2. *Asignar causalidades*, es decir, determinar la variable que se va a calcular con cada ecuación.

Las dos tareas no son independientes: el ordenamiento dependerá de la causalidad, y viceversa. Para abordar estas tareas, nótese que es de esperar que no todas las variables  $x_i$  estén presentes en todas las ecuaciones  $G_i$ . Hay dos formas equivalentes de representar la *estructura* de  $\mathbf{G}(x)$  (es decir, las variables que intervienen en cada ecuación):

- La *matriz de incidencia estructural*: se trata de una matriz  $n \times n$  en la que cada fila representa una ecuación y cada columna una variable. El contenido de cada celda  $i - j$  es un 1 o un 0 que representa si la variable  $j$  está presente o no en la ecuación  $i$ , respectivamente.
- El *dígrafo estructural*: se trata de un grafo con  $2n$  nodos, la mitad de los cuales representan las ecuaciones y la otra mitad las variables. Es usual dibujar el grafo como dos columnas de nodos una enfrente de la otra.

En el grafo se traza una conexión entre el nodo de la ecuación  $i$  y el nodo de la variable  $j$ , cuando la variable  $j$  está presente en la ecuación  $i$ .

Para efectos del proceso de simulación numérica, se busca reordenar las ecuaciones y variables de 2.36, de tal forma que su matriz de incidencia sea triangular inferior. Si se logra, el análisis simbólico habrá sido completamente exitoso.

En la sección 2.4.2 se muestra el algoritmo base para ordenamiento y asignación de causalidades. Este algoritmo logra construir, a partir de 2.36, un sistema de ecuaciones cuya matriz de incidencia es triangular inferior por bloques (BLT, sigla según su nombre en inglés). En la sección 2.4.3 se muestra un segundo algoritmo que realiza un análisis simbólico adicional sobre los bloques de la matriz BLT, para intentar evitar la necesidad de usar un método numérico; por su parte, la sección 2.4.4 muestra un tercer algoritmo útil que es útil para resolver los bloques de la matriz BLT cuando estos constituyen ecuaciones lineales.

### **Ejemplo 2.5: Matriz de incidencia.**

---

La matriz de incidencia del sistema de ecuaciones 2.37 es:

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \quad (2.40)$$

Sin embargo, si se reordenan las ecuaciones y variables, el sistema de ecuaciones se puede escribir como:

$$\begin{cases} \bar{G}_2(\bar{x}_1, \bar{x}_2) = \bar{x}_1 - \sin(t) = 0 \\ \bar{G}_1(\bar{x}_1, \bar{x}_2) = \bar{x}_1 + \bar{x}_2 = 0 \end{cases} \quad (2.41)$$

cuya matriz de incidencia resulta ser:

$$\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \quad (2.42)$$

Esta es una matriz triangular inferior.



### 2.4.2 El algoritmo de Tarjan

El algoritmo de Tarjan<sup>2</sup> opera sobre el dígrafo estructural de un sistema de ecuaciones. Las operaciones se representan mediante colores que se van asignando a los nodos y a sus conexiones. Se utilizan tres colores: negro, azul y rojo. Inicialmente, todas las conexiones tienen color negro, y al finalizar el algoritmo todas ellas tienen colores rojo o azul (ninguno es negro). Adicionalmente, al finalizar el algoritmo cada ecuación y cada variable tiene un número (un orden) asignado. El significado de los colores en un dígrafo ya coloreado es el siguiente:

- Conexión roja: si la conexión entre el nodo de la ecuación  $i$  y el nodo de la variable  $j$  es de color rojo, significa que la ecuación  $i$  se utilizará para actualizar el valor de la variable  $j$ .
- Conexión azul: si la conexión entre el nodo de la ecuación  $i$  y el nodo de la variable  $j$  es de color azul, significa que, al momento de utilizar la ecuación  $i$ , el valor de la variable  $j$  ya fue previamente actualizado.

Durante la ejecución del algoritmo es necesario distinguir los siguientes conceptos:

- a. Ecuación causal: ecuación cuyo nodo tiene exactamente una única conexión roja asociada.
- b. Ecuación acausal: ecuación cuyo nodo tiene solamente conexiones negras y azules asociadas.
- c. Variable conocida: variable cuyo nodo tiene exactamente una única conexión roja asociada.
- d. Variable desconocida: variable cuyo nodo tiene solamente conexiones negras y azules asociadas.

Desde el punto de vista del método de integración, las variables que deben integrarse son variables conocidas. Esta afirmación parece un contrasentido: ¿no se supone que el método de simulación debe, precisamente, calcular el valor de esas variables? ¿Por qué pueden entonces considerarse como conocidas?

La explicación puede visualizarse con el ejemplo 2.3. En cada paso del proceso de iteración se conoce un valor de  $x(t)$ , que se utiliza para calcular

---

<sup>2</sup>La presentación que se hace aquí del algoritmo se basa en la explicación que se encuentra en [CK06], sección 7.2.

el valor de  $x(t + h)$ . De hecho, para  $t = 0$  se conoce el valor de  $x(t) = x(0)$ , la condición inicial.

De igual forma, en cada iteración se conocen las variables que representan las variables exógenas, es decir, las entradas al sistema. El proceso de ordenamiento y asignación de causalidades se enfoca entonces en aquellas variables que no son conocidas al inicio de cada paso de iteración.

El algoritmo consiste en la aplicación iterativa de dos reglas:

Tr.1. Analizar todas las ecuaciones acausales así: si su nodo tiene únicamente una conexión negra asociada entonces:

- Colorear de rojo esa conexión.
- Identificar el nodo de variable que está al otro extremo de la conexión que se ha coloreado de rojo.
- Asignar a la ecuación y a la variable un número. Este será el número entero más bajo disponible, comenzando con el número 1 y aumentándolo sucesivamente.
- Colorear de azul todas las conexiones asociadas al nodo de la variable identificada.

Tr.2. Analizar todas las variables desconocidas así: si su nodo tiene únicamente una conexión negra asociada, entonces:

- Colorear de rojo esa conexión.
- Identificar el nodo de ecuación que está al otro extremo de la conexión que se ha coloreado de rojo.
- Asignar a la variable y la ecuación un número. Este será el número entero más alto disponible, comenzando con el número  $n$  y disminuyéndolo sucesivamente.  $n$  es el número de ecuaciones, que es igual al número de variables.
- Colorear todas las conexiones asociadas al nodo de la ecuación identificada con color azul.

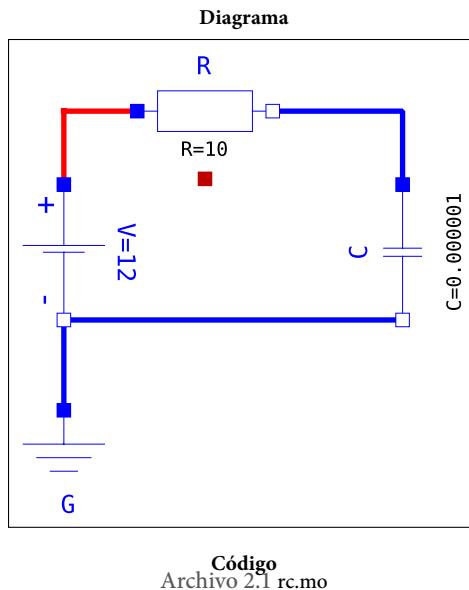
El algoritmo termina cuando todas las conexiones se han coloreado o cuando no sea posible colorear ninguna conexión más. En ese momento se utilizan los números asignados a las ecuaciones y variables, así:

- Las ecuaciones se ordenan de menor a mayor, utilizando el número asignado.

- La variable con el número asignado  $i$  será actualizada utilizando la ecuación cuyo número asociado sea  $i$ .

### Ejemplo 2.6: Algoritmo de Tarjan.

La tabla 2.1 muestra las ecuaciones que se generan al expandir el modelo del archivo rc.mo, que corresponde al modelo en lenguaje Modelica del circuito RC simple que se muestra en el diagrama siguiente (véase ejemplo 4.4):



Código  
Archivo 2.1 rc.mo

```
within Modelica.Electrical.Analog;
model rc
  Basic.Ground g;
  Basic.Resistor R(R=10);
  Basic.Capacitor C(C=1e-6);
  Sources.ConstantVoltage V;
equation
  connect(V.p,R.n);
  connect(R.p,C.n);
  connect(C.p,V.n);
  connect(V.n,g.p);
end rc;
```

Nótese que en el listado de variables de la tabla 2.1 no se han incluido los parámetros (tal como  $C.C$ , la capacitancia del condensador); tampoco se ha incluido la variable  $C.v$  (potencial en el condensador), ya que esta variable aparece derivada como  $der(C.v)$  y en consecuencia se considera conocida.

Tabla 2.1 Numeración de ecuaciones y variables en el ejemplo 2.6

No.	Ecuación	Variable
1	$g.p.v = 0.0;$	$V.n.i$
2	$R.R\_actual = R.R * (1.0 + R.alpha * (R.T\_heatPort - R.T\_ref));$	$V.n.v$
3	$R.v = R.R\_actual * R.i;$	$V.p.i$
4	$R.LossPower = R.v * R.i;$	$V.p.v$
5	$R.v = R.p.v - R.n.v;$	$V.i$
6	$0.0 = R.p.i + R.n.i;$	$V.v$
7	$R.i = R.p.i;$	$C.n.i$
8	$R.T\_heatPort = R.T;$	$C.n.v$
9	$C.i = C.C * der(C.v);$	$C.p.i$
10	$C.v = C.p.v - C.n.v;$	$C.p.v$
11	$0.0 = C.p.i + C.n.i;$	$C.i$
12	$C.i = C.p.i;$	$der(C.v)$
13	$V.v = V.V;$	$R.R\_actual$
14	$V.v = V.p.v - V.n.v;$	$R.T\_heatPort$
15	$0.0 = V.p.i + V.n.i;$	$R.LossPower$
16	$V.i = V.p.i;$	$R.n.i$
17	$g.p.i + C.p.i + V.n.i = 0.0;$	$R.n.v$
18	$R.p.i + C.n.i = 0.0;$	$R.p.i$
19	$R.n.i + V.p.i = 0.0;$	$R.p.v$
20	$R.n.v = V.p.v;$	$R.i$
21	$C.n.v = R.p.v;$	$R.v$
22	$C.p.v = V.n.v;$	$g.p.i$
23	$C.p.v = g.p.v;$	$g.p.v$

Las figuras 2.3 y 2.4 muestran los resultados intermedios y finales de la aplicación del algoritmo de Tarjan sobre las 23 ecuaciones y las 23 variables; la primera figura muestra los resultados en forma de grafos, y la segunda en forma de tablas.

El algoritmo concluye en la iteración número 4. La numeración resultante de ecuaciones y variables se emplea para reordenar tanto ecuaciones como variables. El resultado se muestra en la tabla 2.2. El resultado también se visualiza en las figuras 2.3 y 2.4; el efecto del algoritmo de Tarjan se ve especialmente claro en la última tabla de la figura 2.4, que muestra cómo se ha obtenido una matriz de incidencia triangular inferior.

Tabla 2.2 Ecuaciones y variables del ejemplo 2.6, una vez ordenadas con el algoritmo de Tarjan

No.	Ecuación	No.	Variable
1	$g.p.v = 0.0;$	23	$g.p.v$
8	$R.T\_heatPort = R.T;$	14	$R.T\_heatPort$
13	$V.v = V.V;$	6	$V.v$
23	$C.p.v = g.p.v;$	10	$C.p.v$
2	$R.R\_actual = R.R * (1.0 + R.alpha * (R.T\_heatPort - R.T\_ref));$	13	$R.R\_actual$
10	$C.v = C.p.v - C.n.v;$	8	$C.n.v$
21	$C.n.v = R.p.v;$	19	$R.p.v$
22	$C.p.v = V.n.v;$	2	$V.n.v$
14	$V.v = V.p.v - V.n.v;$	4	$V.p.v$
20	$R.n.v = V.p.v;$	17	$R.n.v$
5	$R.v = R.p.v - R.n.v;$	21	$R.v$
3	$R.v = R.R\_actual * R.i;$	20	$R.i$
7	$R.i = R.p.i;$	18	$R.p.i$
18	$R.p.i + C.n.i = 0.0;$	7	$C.n.i$
11	$0.0 = C.p.i + C.n.i;$	9	$C.p.i$
6	$0.0 = R.p.i + R.n.i;$	16	$R.n.i$
12	$C.i = C.p.i;$	11	$C.i$
19	$R.n.i + V.p.i = 0.0;$	3	$V.p.i$
15	$0.0 = V.p.i + V.n.i;$	1	$V.n.i$
17	$g.p.i + C.p.i + V.n.i = 0.0;$	22	$g.p.i$
4	$R.LossPower = R.v * R.i;$	15	$R.LossPower$
9	$C.i = C.C * der(C.v);$	12	$der(C.v)$
16	$V.i = V.p.i;$	5	$V.i$

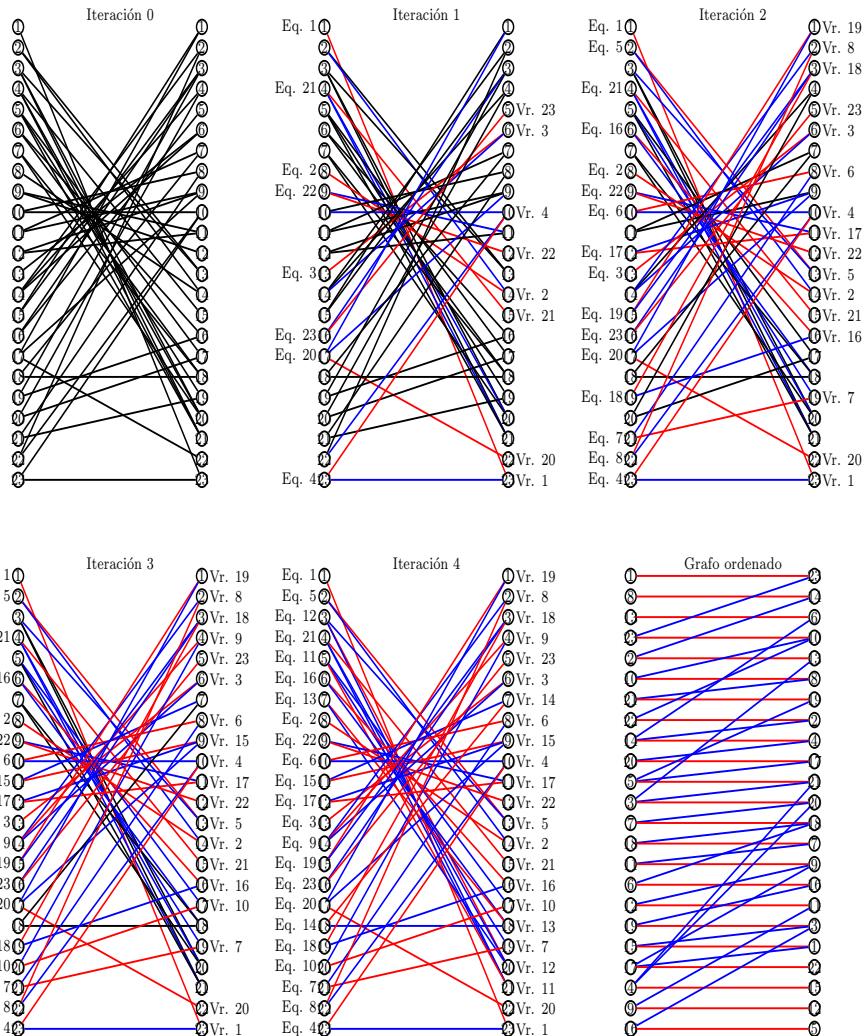


Figura 2.3 Aplicación del algoritmo de Tarjan al ejemplo 2.6, visualizado con grafos

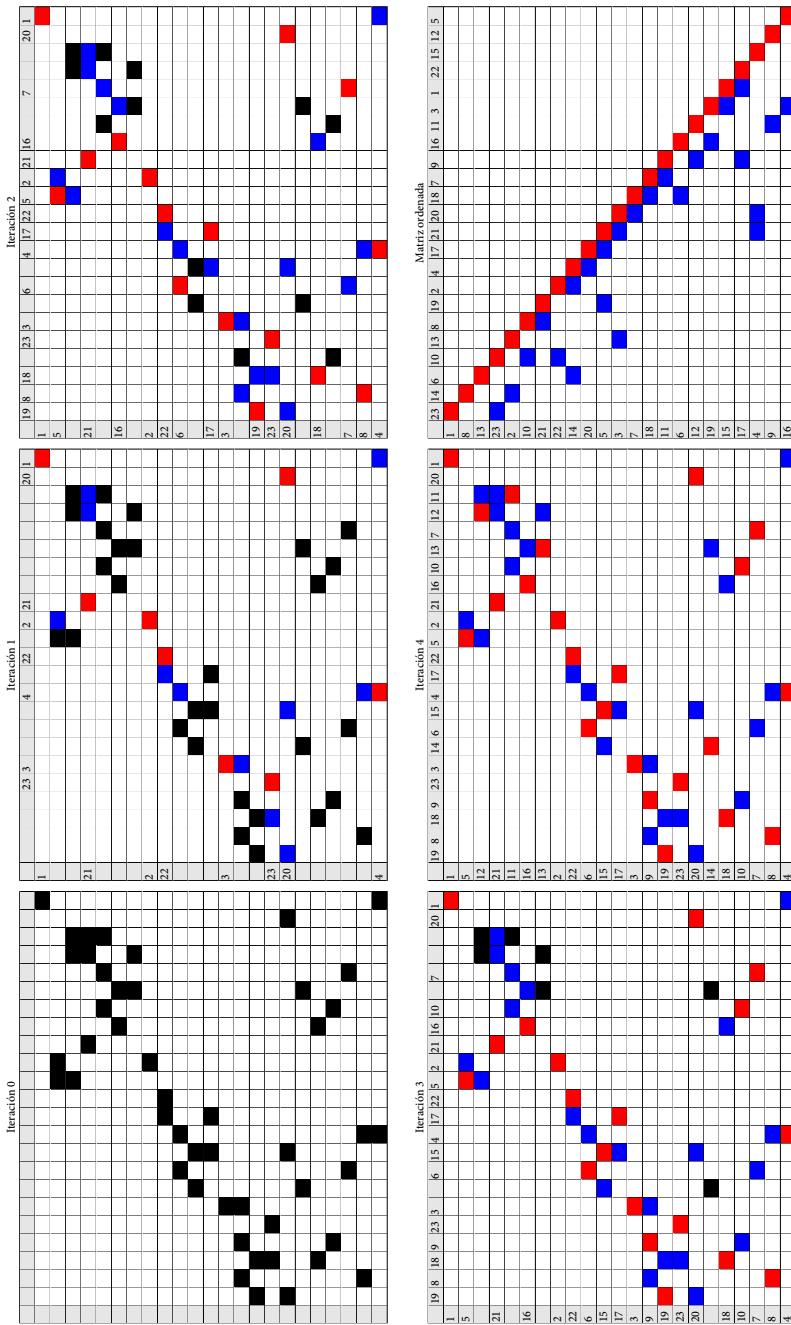


Figura 2.4 Aplicación del algoritmo de Tarjan al ejemplo 2.6, visualizado matricialmente

La variable número  $i$  debe ser obtenida a partir de la ecuación número  $i$ .

Como resultado, las ecuaciones de asignación de valores que resultan son las que se muestran en la tabla 2.3. Se ha empleado el símbolo  $:=$  del lenguaje Modelica como operador de asignación. Esto no significa que sea necesario reescribir las ecuaciones en lenguaje Modelica. De hecho, el compilador reescribirá las ecuaciones en algún lenguaje de programación (probablemente C), como se explica en 4.4.

Tabla 2.3 Ecuaciones de asignación en el ejemplo 2.6

No.	Ecuación
1	$g.p.v := 0.0;$
2	$R.T\_heatPort := R.T;$
3	$V.v := V.V;$
4	$C.p.v := g.p.v;$
5	$R.R\_actual := R.R * (1.0 + R.alpha * (R.T\_heatPort - R.T\_ref));$
6	$C.n.v = C.p.v - C.v;$
7	$R.p.v := C.n.v;$
8	$V.n.v := C.p.v;$
9	$V.p.v := V.v + V.n.v;$
10	$R.n.v := V.p.v;$
11	$R.v := R.p.v - R.n.v;$
12	$R.i := R.v / R.R\_actual;$
13	$R.p.i := R.i;$
14	$C.n.i := - R.p.i;$
15	$C.p.i = - C.n.i;$
16	$R.n.i = - R.p.i;$
17	$C.i := C.p.i;$
18	$V.p.i := - R.n.i;$
19	$V.n.i := - V.p.i;$
20	$g.p.i := - C.p.i - V.n.i;$
21	$R.LossPower := R.v * R.i;$
22	$\text{der}(C.v) := C.i / C.C;$
23	$V.i := V.p.i;$



### 2.4.3 Algoritmos de rasgadura

Los algoritmos de rasgadura se aplican sobre los bloques de la BLT de dimensión mayor que 1. El algoritmo básico consta de los siguientes pasos:

- Rs.1. Seleccionar una o más variables del bloque como *variables de rasgadura*.
- Rs.2. Suponer que el valor de las variables de rasgadura es conocido.
- Rs.3. Aplicar el algoritmo de ordenamiento sobre el sistema de ecuaciones con las nuevas variables “conocidas”.
- Rs.4. Construir la ecuación:

$$\tilde{\mathcal{F}} = \tilde{\mathbf{x}}_n - \tilde{\mathbf{x}} = \mathbf{0} \quad (2.43)$$

en la que  $\tilde{\mathbf{x}}$  es el vector de variables de rasgadura, y el subíndice  $n$  se utiliza para distinguir un valor inicial supuesto (sin subíndice) y un valor nuevo.

- Rs.5. Utilizar un método numérico para resolver 2.43.

Este algoritmo reduce el costo computacional de la utilización del método numérico al aplicarlo sobre un número reducido de variables. El algoritmo requiere la selección del conjunto de variables de rasgadura, que resulta ser un problema NP. Por esta razón, se utilizan heurísticas para la selección de las variables. En [CK06] se propone la siguiente:

- Rv.1. Construir el dígrafo estructural de las ecuaciones.
- Rv.2. Identificar las ecuaciones con la mayor cantidad de conexiones asociadas a sus nodos (líneas negras).
- Rv.3. Para cada una de las ecuaciones identificadas, evaluar cuáles de las variables a las que están conectadas tienen el mayor número de conexiones asociadas (líneas negras).
- Rv.4. Para cada una de las variables identificadas en el paso anterior, determinar cuántas ecuaciones se convertirían en causales si dichas variables fueran conocidas, y marcar las que tienen el número mayor.
- Rv.5. Seleccionar una de las variables como la siguiente variable de rasgadura dentro las variables marcadas en el paso anterior.

**Ejemplo 2.7: Algoritmo de rasgadura.**

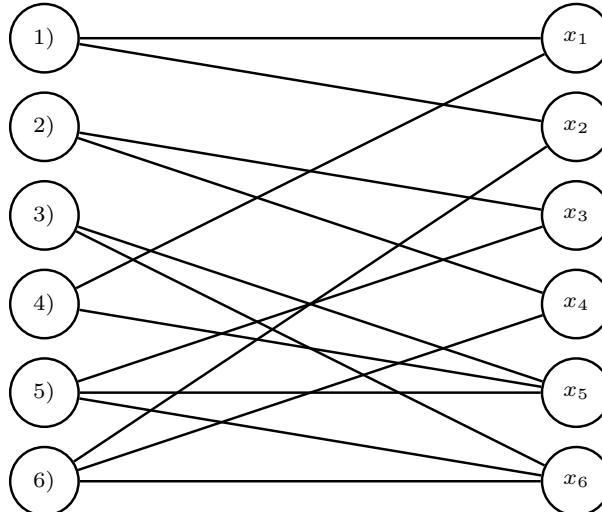
Considérese el siguiente sistema de ecuaciones:

$$\left\{ \begin{array}{l} 1) \quad x_1 - x_2 = 0 \\ 2) \quad x_3 - x_4 = 0 \\ 3) \quad x_5 - x_6 = 0 \\ 4) \quad x_1 + x_5 = b \\ 5) \quad x_3 - x_5 - x_6 = 0 \\ 6) \quad x_2 - x_4 - x_6 = 0 \end{array} \right. \quad (2.44)$$

cuya matriz de incidencia corresponde a:

Eq	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$
1)	1	1	0	0	0	0
2)	0	0	1	1	0	0
3)	0	0	0	0	1	1
4)	1	0	0	0	1	0
5)	0	0	1	0	1	1
6)	0	1	0	1	0	1

El dígrafo estructural del sistema resulta ser:



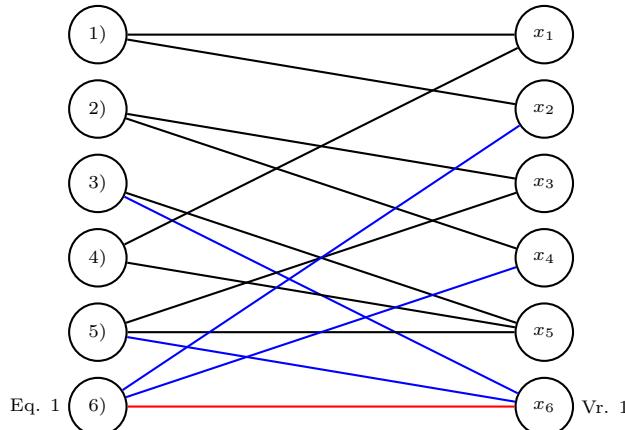
Al intentar aplicar el algoritmo de Tarjan, se encuentra que no es posible causalizar ninguna ecuación, porque para todas ellas sus nodos tienen asociados más de una conexión negra. Hay, por tanto, un lazo algebraico, e intentamos aplicar el algoritmo de rasgadura.

Para seleccionar las variables de rasgadura, identificamos la ecuación 6) como la que tiene un mayor número de conexiones asociadas (3 en total). Las variables correspondientes a esas ecuaciones son  $x_2$ ,  $x_4$  y  $x_6$ . De esas variables, la que tiene más conexiones asociadas es  $x_6$  (3 en total). Por esta razón, seleccionamos  $x_6$  como la variable de rasgadura en la ecuación 6).

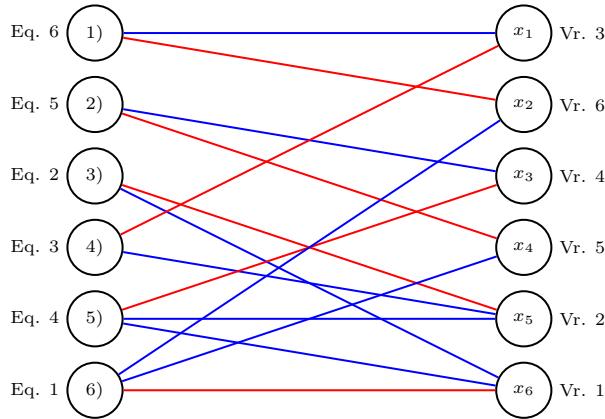
El nuevo sistema de ecuaciones es:

$$\left\{ \begin{array}{ll} 1) & x_1 - x_2 = 0 \\ 2) & x_3 - x_4 = 0 \\ 3) & x_5 - x_6 = 0 \\ 4) & x_1 + x_5 = b \\ 5) & x_3 - x_5 - x_6 = 0 \\ 6) & x_2 - x_4 - x_{6_n} = 0 \end{array} \right. \quad (2.45)$$

La causalización de  $x_6$  en la ecuación 6) permite colorear el dígrafo estructural así:



Despues de dos iteraciones, el algoritmo de Tarjan ha coloreado completamente el dígrafo:



El sistema de ecuaciones resultante es, entonces:

$$\left\{ \begin{array}{l} x_{6n} = x_2 - x_4 \\ x_5 = x_6 \\ x_1 = b - x_5 \\ x_3 = x_5 + x_6 \\ x_4 = x_3 \\ x_2 = x_1 \end{array} \right. \quad (2.46)$$

En el sistema de ecuaciones anterior,  $x_6$  es un valor inicial estimado, que se recalcula como  $x_{6n}$ . Por lo tanto, podemos escribir la ecuación residual:

$$x_{6n} - x_6 = 0 \quad (2.47)$$

que puede resolverse mediante métodos numéricos tipo Newton. Al comparar 2.44 con 2.47, se observa que hemos disminuido considerablemente la dimensión del problema genérico  $\mathbf{F}(\mathbf{x}) = \mathbf{0}$  (de dimensión 6 a dimensión 1).

■

#### 2.4.4 Algoritmo de relajación

Existen varios algoritmos denominados ‘de relajación’ en distintas aplicaciones computacionales. El algoritmo al que aquí nos referimos se emplea para manipular sistemas de ecuaciones lineales de la forma  $\mathbf{Ax} = \mathbf{b}$ , mediante una aplicación simbólica del método de eliminación de Gauss sin pivote. El algoritmo puede resumirse en el siguiente procedimiento:

- Rj.1. Causalizar las ecuaciones siguiendo el mismo procedimiento del algoritmo de rasgadura.
- Rj.2. Reescribir las ecuaciones dejando las variables desconocidas al lado derecho de la igualdad, y realizando las multiplicaciones necesarias para evitar que aparezcan fracciones.
- Rj.3. Formular la ecuación matricial usando el orden de ecuaciones y variables asignado en la causalización.
- Rj.4. Aplicar iterativamente la eliminación gaussiana sin pivote, eliminando en cada iteración el elemento de la primera fila y primera columna. En cada iteración, se definen nuevas variables (auxiliares) en lugar de hacer las sustituciones simbólicas.
- Rj.5. Al finalizar las iteraciones se resuelve simbólicamente la ecuación final y se realizan las sustituciones simbólicas para cada variable.
- Rj.6. El conjunto de ecuaciones se ordena tomando primero las ecuaciones que definen las variables auxiliares en orden de aparición, y posteriormente las ecuaciones que resuelven cada una de las incógnitas originales en orden contrario al original.
- Rj.7. Como resultado final se obtiene un conjunto de ecuaciones de tamaño mayor que el original, pero con la gran ventaja de tener una matriz de incidencia triangular inferior.

---

**Ejemplo 2.8: Algoritmo de relajación.**

Supóngase el sistema de ecuaciones:

$$\begin{cases} 1) & a_{11}x_1 + a_{12}x_2 = b_1 \\ 2) & a_{21}x_1 + a_{22}x_2 = b_2 \end{cases} \quad (2.48)$$

cuya matriz de incidencia es:

Eq	$x_1$	$x_2$
1)	1	1
2)	1	1

El sistema puede representarse usando la matriz extendida:

$$\left( \begin{array}{cc|c} a_{11} & a_{12} & b_1 \\ a_{21} & a_{22} & b_2 \end{array} \right) \quad (2.49)$$

En el primer paso de eliminación gaussiana se obtiene:

$$\left( \begin{array}{cc|c} a_{11} & a_{12} & b_1 \\ 0 & c_1 & c_2 \end{array} \right) \quad (2.50)$$

en donde ha sido necesario definir las variables auxiliares  $c_1$  y  $c_2$ :

$$\begin{aligned} c_1 &= a_{22} - a_{12} \frac{1}{a_{11}} a_{21} \\ c_2 &= b_2 - b_1 \frac{1}{a_{11}} a_{21} \end{aligned} \quad (2.51)$$

Las ecuaciones 2.50 y 2.51 se ordenan, entonces:

$$\begin{cases} 1) & a_{22} - a_{12} \frac{1}{a_{11}} a_{21} = c_1 \\ 2) & b_2 - b_1 \frac{1}{a_{11}} a_{21} = c_2 \\ 3) & c_1 x_2 = c_1 \\ 4) & a_{11} x_1 + a_{12} x_2 = b_1 \end{cases} \quad (2.52)$$

que resulta ser un bloque de tamaño 4, cuya matriz de incidencia es la matriz triangular inferior:

Eq	$c_1$	$c_2$	$x_2$	$x_1$
1)	1	0	0	0
2)	0	1	0	0
3)	1	0	1	0
4)	0	0	1	1

■

#### 2.4.5 El algoritmo de Pantelides para la reducción del índice

El algoritmo de Pantelides se utiliza para abordar las singularidades estructurales. Mediante un procedimiento simbólico, el algoritmo reduce en 1 el índice de perturbación. Para problemas de índice elevado, es necesario aplicar el algoritmo en varias ocasiones. Es usual que la aplicación del algoritmo conlleve la aparición de lazos algebraicos, que pueden abordarse con el algoritmo de rasgadura.

Las singularidades estructurales se manifiestan en la forma de ecuaciones de restricción algebraicas. En el proceso de ordenamiento de ecuaciones con el algoritmo de Tarjan, una de estas ecuaciones se detecta cuando su nodo solamente tiene asociadas conexiones azules. Se trata de ecuaciones en las que todas las variables ya tienen valores conocidos, y por tanto no hay ningún nuevo valor por calcular.

El algoritmo de Pantelides consta de los siguientes pasos:

- Pt.1. Derivar simbólicamente la ecuación de restricción. En este proceso, es posible que aparezcan nuevas variables (las derivadas simbólicas).
- Pt.2. Adicionar la nueva ecuación al sistema.

- Pt.3. Equilibrar el número de variables y ecuaciones. La estrategia consiste en considerar algunas de las derivadas de las variables como nuevas variables de estado; en otras palabras, se desconecta la relación de integración entre esa variable y su derivada. En ocasiones, basta con ignorar la ecuación de restricción.

### *Ejemplo 2.9: Algoritmo de Pantelides.*

---

Considérese el sistema de ecuaciones:

$$\left\{ \begin{array}{l} 1) \quad x_1 = f(t) \\ 2) \quad x_3 + x_4 = 0 \\ 3) \quad x_1 = \frac{dx_3}{dt} \\ 4) \quad x_2 = \frac{dx_4}{dt} \end{array} \right. \quad (2.53)$$

Como las variables  $x_3$  y  $x_4$  aparecen derivadas, se consideran como variables de estado y, por tanto, como variables conocidas. Eso significa que la segunda ecuación es de restricción. Al derivar esa ecuación, el sistema se convierte en:

$$\left\{ \begin{array}{l} 1) \quad x_1 = f(t) \\ 2) \quad x_3 + x_4 = 0 \\ 3) \quad x_1 = \frac{dx_3}{dt} \\ 4) \quad x_2 = \frac{dx_4}{dt} \\ 5) \quad \frac{dx_3}{dt} + \frac{dx_4}{dt} = 0 \end{array} \right. \quad (2.54)$$

Para equilibrar el número de variables y de ecuaciones, eliminamos la segunda ecuación:

$$\begin{cases} 1) & x_1 = f(t) \\ 2) & x_1 = \frac{dx_3}{dt} \\ 3) & x_2 = \frac{dx_4}{dt} \\ 4) & \frac{dx_3}{dt} + \frac{dx_4}{dt} = 0 \end{cases} \quad (2.55)$$

La matriz de incidencia de este conjunto de ecuaciones es:

Eq	$x_1$	$x_2$	$dx_3/dt$	$dx_4/dt$
1)	1	0	0	0
2)	1	0	1	0
3)	0	1	0	1
4)	0	0	1	1

Al reordenar las ecuaciones y variables se puede obtener una matriz triangular inferior:

Eq	$x_1$	$dx_3/dt$	$dx_4/dt$	$x_2$
1)	1	0	0	0
2)	1	1	0	0
4)	0	1	1	0
3)	0	0	1	1

■

## 2.5 SIMULACIÓN DE EVENTOS DISCRETOS

La simulación de una DAE se dificulta cuando las funciones involucradas presentan discontinuidades. En general, los algoritmos de simulación numérica han sido diseñados y analizados bajo premisas de continuidad de las funciones. Ante la presencia de discontinuidades, los algoritmos de integración suelen fallar. En el contexto de la simulación de sistemas dinámicos, las discontinuidades reciben el nombre de *eventos discretos* o, simplemente, *eventos*.

Los eventos pueden clasificarse en (ver [Naj05]):

- Eventos temporales: cuando su ocurrencia está prevista en un cierto instante de tiempo<sup>3</sup>. Las ecuaciones de diferencias finitas, por ejemplo, presentan eventos temporales, ya que las discontinuidades suceden en instantes de tiempo predefinidos.

---

<sup>3</sup>El término ‘temporal’ adquiere aquí la connotación de ‘asociado al tiempo’; no significa ‘no permanente’.

- Eventos de estado: cuando su ocurrencia depende del valor o de los valores que en algún momento tome o tomen una o varias variables del sistema. Un interruptor eléctrico controlado por un flotador, por ejemplo, es un evento de estado, pues la condición abierto/cerrado del interruptor dependerá del nivel del fluido que mide el flotador.

¿Cómo simular eventos que contengan simultáneamente fenómenos continuos y discretos? La estrategia consiste en separar el modelo en dos subsistemas, tal como se muestra en la figura 2.5. Los dos subsistemas pueden compartir variables comunes. La lógica subyacente a la separación puede explicarse así:

- El disparo de los eventos está determinado por los valores que tomen algunas variables del sistema continuo (el tiempo puede ser una de ellas).
- La ocurrencia de los eventos puede alterar el modelo del sistema continuo. Estos cambios pueden ser:
  - Un cambio (brusco) en el valor de una o más variables presentes en el modelo.
  - Un cambio (brusco) en las ecuaciones que definen el modelo.

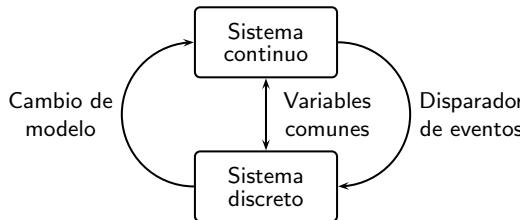


Figura 2.5 Interacción entre variables de sistemas híbridos. Adaptado de [Naj05]

Para que los dos subsistemas puedan convivir en un mismo ambiente de simulación, las variables discretas se transforman en continuas mediante retenedores de orden cero. La figura 2.6 reproduce el algoritmo propuesto en [And94] para simular conjuntamente los dos subsistemas. La lógica del algoritmo es la siguiente:

- Los métodos de simulación numérica se emplean sobre la DAE, *suponiendo que no ocurre ningún evento.*

- En cada paso de la simulación se verifica la ocurrencia o no de eventos, tanto temporales como de estado.
- Cuando sucede un evento, se actualiza el modelo de la DAE antes de continuar con el siguiente paso de simulación.

Existen algoritmos más sofisticados que el esbozado en la figura 2.6 (véanse por ejemplo [Naj05, CK06]). Además, las herramientas de modelado de eventos juegan un papel preponderante en el diseño de los algoritmos (véase, por ejemplo, [UD03]).

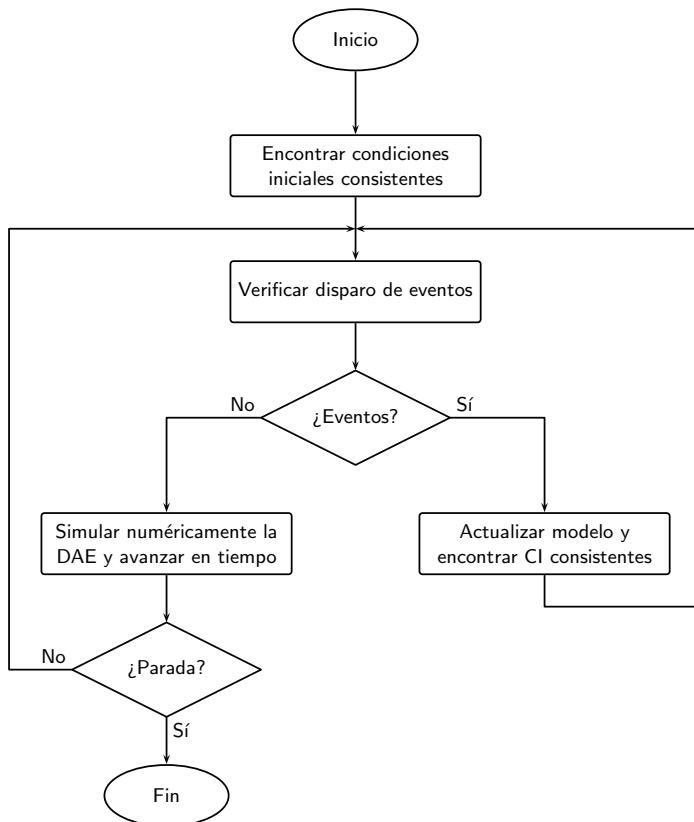


Figura 2.6 Diagrama de flujo básico de un simulador híbrido. Adaptado de [And94]

## 2.6 SELECCIÓN DEL MÉTODO NUMÉRICO

En el corazón del proceso de simulación está el método de simulación numérica utilizado. La literatura que analiza estos métodos es abundante

y profunda (véanse por ejemplo [HNG00, HW02, BT99]). La dificultad en la selección del método numérico radica en que su desempeño depende del problema que se va a simular. En otras palabras, no se puede conocer con anticipación cuál de los métodos disponibles es el más adecuado para un problema en particular.

Si se dispone de un análisis matemático del problema a simular, quizás sea posible elegir o descartar algún método, basándose en las propiedades generales de este. Por ejemplo, para simular una ODE lineal de bajo orden y valores propios comparables, el método explícito de Euler es un buen candidato (aunque no es infalible). No obstante, cuando se trata de simular modelos obtenidos bajo el paradigma de modelado orientado a objetos, es muy difícil contar con el análisis matemático del modelo, porque este está generalmente disperso (y quizás oculto) en los modelos de sus componentes y conexiones, y solo se construye de forma integrada en tiempo de compilación.

Las ventajas y desventajas de cada método disponible hacen que la selección sea una tarea de balance en el desempeño. Así, por ejemplo, la facilidad de implementación y rapidez de cómputo del método de Euler explícito debe contrastarse con su pobre robustez.

Se propone aquí un algoritmo simple para la elección del método de simulación:

- MI.1. Simular el modelo con todos los métodos disponibles y comparar los resultados.
- MI.2. Descartar los métodos que no convergen.
- MI.3. Si no existen diferencias notorias en los resultados, ir al paso MI.6.
- MI.4. Ajustar los parámetros de los métodos disponibles (tolerancia, paso de integración, orden, etc.) para intentar que los resultados de las simulaciones sean comparables. Usar como referencia de comparación los métodos más robustos.
- MI.5. Descartar los métodos con los que no haya sido posible obtener resultados comparables.
- MI.6. Analizar los tiempos de simulación de los métodos con simulaciones comparables.
- MI.7. Seleccionar el método con los menores tiempos de simulación.

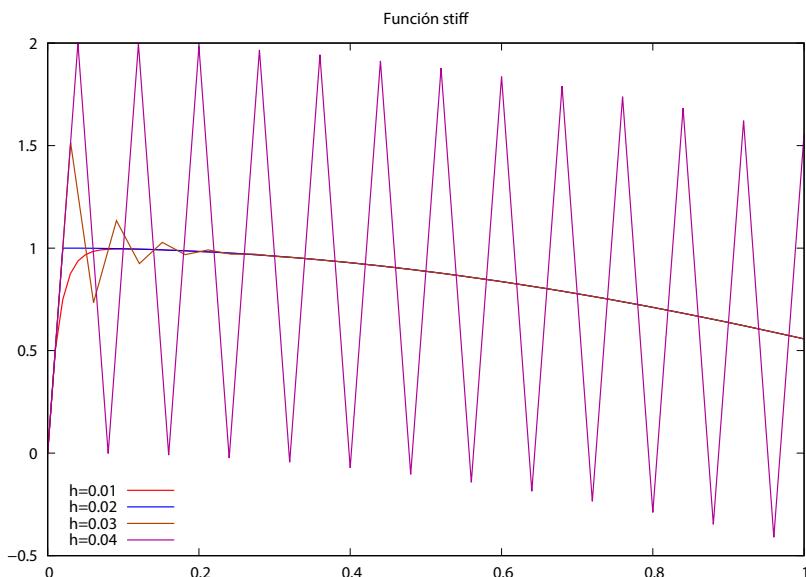
### Ejemplo 2.10: Ecuaciones stiff.

---

En [HW02] se analiza la siguiente ecuación como ejemplo de ecuación *stiff*

$$\frac{dy}{dt} = -50(y - \cos(t)) \quad (2.56)$$

La siguiente figura muestra los resultados de la simulación numérica con el método de Euler para distintos pasos de integración.



Si bien es cierto que, en términos generales, se espera que la precisión de la simulación dependa del paso de integración, la dependencia en este ejemplo es notable. Esa es la característica principal de los problemas *stiff*.

■

### Ejemplo 2.11: Desempeño de los métodos numéricicos de simulación.

---

Para ilustrar la dificultad en la selección del método, se ha desarrollado un experimento con algunos de los modelos que se presentan en la parte III. La tabla 2.4 muestra los modelos utilizados, así como las variables que se han escogido para comparar el desempeño de los métodos.

Tabla 2.4 Modelos y variables simulados en el ejemplo 2.11

item	Nombre	Capítulo	Archivo	Variable
a	Motor DC	8	DCmotor.mo	Corriente en la resistencia.
b	Cable aéreo	9	MyStandAloneLine.mo	Flecha de la catenaria.
c	Epidemia	10	dosserotipos.mo	Modelo SIR-SI con nacimientos. Infectados por el primer serotipo.
d	Estudiantes	11	ingresoConstante.mo	Ingreso constante. Número total de estudiantes.
e	Central hidroeléctrica	12	HidroPI.mo	Frecuencia de la tensión generada.
f	Modelo del mundo	13	MyWorld3.mo	Modelo de referencia. Población mundial.
g	Identificación algebraica	14	idmasa.mo	Masa estimada.

El experimento diseñado se presenta a continuación:

- Ex.1. Se han seleccionado cinco de los métodos de simulación disponibles en la herramienta OpenModelica (véase capítulo 4).
- Ex.2. Se han simulado los siete modelos de la tabla 2.4 con cada uno de los cinco métodos. Para cada pareja modelo-método, se han corrido diez simulaciones.
- Ex.3. En cada simulación se han obtenido varios indicadores de desempeño. Para cada pareja modelo-método se han calculado los promedios de las siete simulaciones.
- Ex.4. Para poder comparar los resultados de desempeño, se ha seleccionado el método Dassl como referencia. Los promedios de los indicadores de desempeño se han normalizado tomando como referencia los promedios de desempeño del método Dassl. Los resultados se muestran en las tablas 2.7 a 2.10.
- Ex.5. Para cada método de simulación, se han calculado los promedios normalizados de sus indicadores en todos los modelos simulados. Los resultados se muestran en la tabla 2.5.

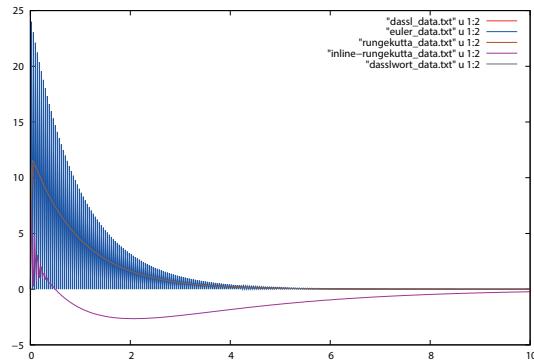
Tabla 2.5 Comparación experimental del desempeño de algunos métodos de simulación numérica.  
Véase el ejemplo 2.11

medida	Métodos				
	dassl	euler	rungekutta	inline-rungekutta	dasslwort
outputFilesize	1.000	0.992	1.002	1.008	1.000
overheadTime	1.000	0.929	0.926	0.969	0.995
preinitTime	1.000	0.950	0.945	0.976	1.003
initTime	1.000	1.006	1.009	1.015	1.001
eventTime	1.000	1.999	1.933	5.420	1.962
outputTime	1.000	0.989	1.103	1.012	0.997
totalTime	1.000	0.576	0.639	1.789	0.768
totalStepsTime	1.000	0.380	0.464	1.747	0.683
numStep	1.000	1.001	1.000	1.009	1.000
maxTime	1.000	0.943	0.869	0.861	1.577

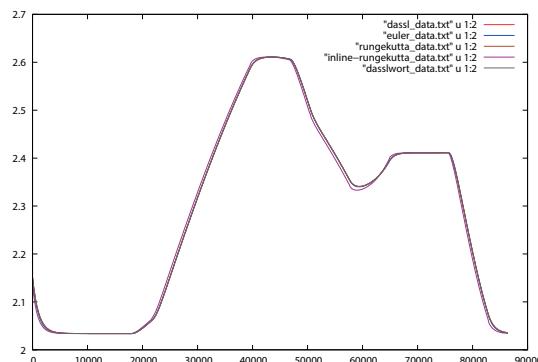
De acuerdo con los resultados de la tabla 2.5, el método más veloz (`totalTime`) es el de Euler, lo que lo hace muy atractivo. No obstante, la rapidez del método es un criterio secundario en función de su robustez: la figura 2.7 muestra los resultados de simulación de las variables que se listan en la tabla 2.4 con los cinco métodos.

Se puede observar que en tres de los siete casos analizados (a, e, g), los resultados de la simulación están fuertemente influenciados por la selección del método. Nótese que un caso aparentemente simple como el del motor DC (un modelo lineal de segundo orden) presenta dificultades para varios métodos. También es de anotar que para este experimento fue necesario descartar el método inline-Euler, debido a que no convergía con algunos de los modelos seleccionados.

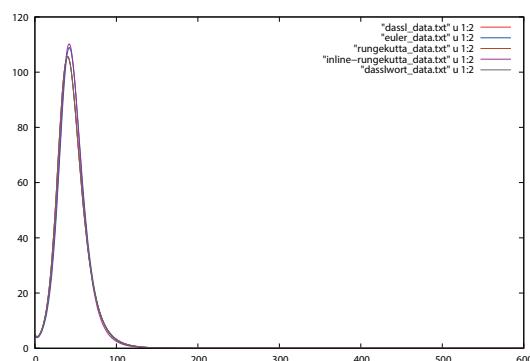
a) Motor DC



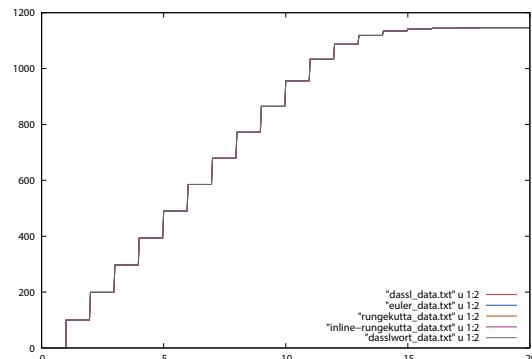
b) Cable Aéreo



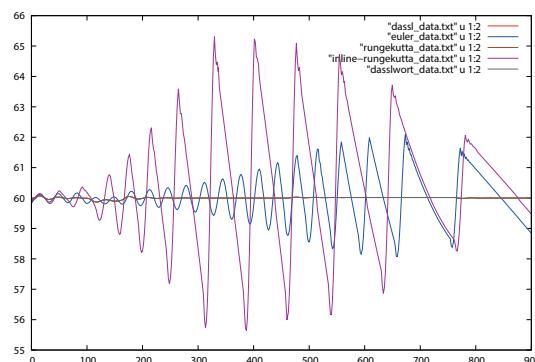
c) Epidemia



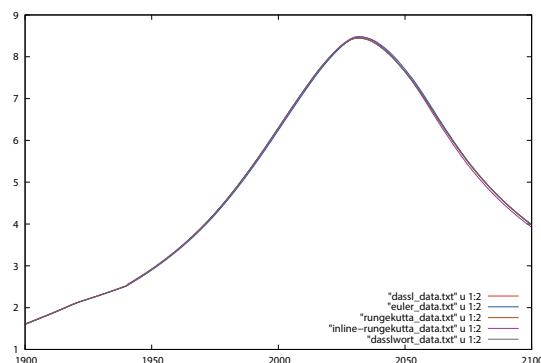
d) Estudiantes



e) Central hidroeléctrica



f) Modelo del mundo



## g) Identificación algebraica

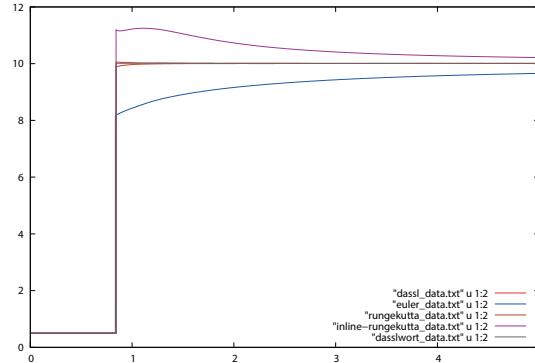


Figura 2.7 Simulación de algunos modelos de la parte III con varios métodos de simulación numérica.  
Ver el ejemplo 2.11

### 2.6.1 Algunas consideraciones sobre los métodos de simulación

La obtención de métodos de simulación numérica de ecuaciones diferenciales es un problema de vieja data. Leonard Euler expuso su método en el año 1768 ([Eul68]) e importantes avances se obtuvieron en el siglo XIX (el método de Adams aparece en 1855 y el de Runge en 1895).

El listado de métodos desarrollados desde ese entonces es extenso. Las principales herramientas de *software* de simulación actuales utilizan como algoritmo por defecto el método Dassl (*Differential/Algebraic System Solver*), que fue propuesto en 1982 por Linda Petzold en [Pet82]. El problema está lejos de considerarse resuelto, y sigue siendo tema de investigación (véase, por ejemplo, la página web del grupo de investigación de Petzold en [oCaSB14]).

El método Dassl tiene ciertas características que lo han convertido en la opción por defecto para la simulación numérica de DAEs<sup>4</sup>. Estas son algunas de ellas:

- Más que un método, se trata de la implementación de la familia de métodos BDF (*Backward Differentiation Formulas*) de órdenes 1 a 5.
- Como método, puede considerarse del tipo implícito, de paso variable y de orden variable.

---

<sup>4</sup>Para una discusión detallada, ver [Naj05], sección 2.4.2.

- Incorpora una versión modificada del método de Newton en la solución del sistema lineal en cada paso. El método puede utilizar el jacobiano dado de forma explícita, o calculado numéricamente.
- Su robustez radica en la habilidad para utilizar pasos estables grandes, manteniendo el nivel de precisión de otros métodos.
- Es un método con buen comportamiento para la simulación de problemas *stiff* con un gran número de variables.
- Tiene buena convergencia probada para sistemas explícitos de índice 1 y semiexplícitos de índice 2. Sin embargo, requiere ser inicializado con un conjunto de condiciones iniciales consistentes, de lo contrario la convergencia no está asegurada.
- Su implementación es abierta y robusta ([oCaSB14], sección *Software*).
- Su implementación incluye varios submétodos especializados.

Tabla 2.6 Comparación experimental del desempeño del método `dassl` en relación al método `dassl`.

Ver. Ver el ejemplo 2.11

medida	Casos							medida
	Motor DC	Cable	Central	SIR-SI	Estudiantes	Identificación	Mundo	
outputFilesize	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
overheadTime	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
preinitTime	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
initTime	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
eventTime	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
outputTime	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
totalTime	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
totalStepsTime	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
numStep	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
maxTime	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000

Tabla 2.7 Comparación experimental del desempeño del método **euler** en relación al método **dassl**.  
 Ver . Ver el ejemplo 2.11

medida	Casos							medida
	Motor DC	Cable	Central	SIR-SI	Estudiantes	Identificación	Mundo	
outputFilesize	0.972	1.000	1.017	0.959	1.000	0.994	1.000	0.992
overheadTime	0.979	0.942	0.974	0.911	0.960	0.890	0.849	0.929
preinitTime	0.942	1.000	0.896	0.926	0.974	0.946	0.967	0.950
initTime	1.008	1.018	0.988	1.038	1.000	1.001	0.993	1.006
eventTime	0.997	3.789	4.159	1.115	1.043	0.993	1.896	1.999
outputTime	1.022	1.002	1.000	0.980	1.000	0.976	0.942	0.989
totalTime	0.805	0.730	0.267	0.803	0.730	0.385	0.309	0.576
totalStepsTime	0.434	0.518	0.195	0.343	0.509	0.363	0.296	0.380
numStep	1.000	1.000	1.004	1.000	1.000	1.000	1.000	1.001
maxTime	0.032	1.109	0.891	0.279	0.719	0.654	2.914	0.943

Tabla 2.8 Comparación experimental del desempeño del método **rungekutta** en relación al método **dassl**. Ver . Ver el ejemplo 2.11

medida	Casos							medida
	Motor DC	Cable	Central	SIR-SI	Estudiantes	Identificación	Mundo	
outputFilesize	0.996	1.000	1.062	0.960	1.000	0.997	1.000	1.002
overheadTime	0.972	0.920	0.941	0.917	0.955	0.899	0.877	0.926
preinitTime	0.953	0.958	0.902	0.924	0.983	0.937	0.959	0.945
initTime	1.015	1.001	1.009	1.037	1.002	0.999	0.996	1.009
eventTime	1.005	3.744	3.721	1.122	1.041	0.999	1.895	1.933
outputTime	0.988	0.990	1.026	0.979	1.000	0.984	1.758	1.103
totalTime	0.793	0.764	0.586	0.822	0.730	0.392	0.389	0.639
totalStepsTime	0.458	0.588	0.542	0.415	0.508	0.369	0.368	0.464
numStep	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
maxTime	0.027	1.033	0.662	0.287	0.709	0.717	2.652	0.869

Tabla 2.9 Comparación experimental del desempeño del método `inline-rungekutta` en relación al método `dassl`. Ver . Ver el ejemplo 2.11

medida	Casos							medida
	Motor DC	Cable	Central	SIR-SI	Estudiantes	Identificación	Mundo	
outputFilesize	0.961	1.055	1.077	0.967	1.000	0.999	1.000	1.008
overheadTime	1.009	1.007	0.880	1.011	0.973	0.903	1.002	0.969
preinitTime	0.993	0.982	0.939	0.958	0.999	0.964	1.001	0.976
initTime	1.008	1.020	1.037	1.016	1.001	1.010	1.010	1.015
eventTime	1.010	9.425	22.425	1.127	1.044	1.012	1.900	5.420
outputTime	0.964	1.052	1.044	0.994	1.000	0.996	1.034	1.012
totalTime	0.785	0.961	8.398	0.842	0.731	0.399	0.406	1.789
totalStepsTime	0.474	0.913	9.116	0.447	0.509	0.376	0.394	1.747
numStep	1.000	1.024	1.040	1.000	1.000	1.000	1.000	1.009
maxTime	0.030	1.097	0.697	0.515	0.693	0.605	2.387	0.861

Tabla 2.10 Comparación experimental del desempeño del método `dasslwort` en relación al método `dassl`. Ver . Ver el ejemplo 2.11

medida	Casos							medida
	Motor DC	Cable	Central	SIR-SI	Estudiantes	Identificación	Mundo	
outputFilesize	1.000	1.001	1.000	1.000	1.000	1.001	1.000	1.000
overheadTime	1.034	0.983	1.018	0.952	1.033	0.991	0.957	0.995
preinitTime	1.035	0.996	0.986	0.990	1.006	1.017	0.989	1.003
initTime	1.008	1.000	1.011	0.998	0.989	0.996	1.004	1.001
eventTime	1.216	3.740	3.739	1.151	1.003	1.009	1.878	1.962
outputTime	1.002	0.988	1.003	1.001	1.008	0.999	0.980	0.997
totalTime	0.878	0.798	0.644	0.934	1.008	0.516	0.600	0.768
totalStepsTime	0.663	0.647	0.608	0.761	1.009	0.498	0.592	0.683
numStep	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
maxTime	0.876	1.040	1.990	0.887	1.088	0.716	4.444	1.577

# 3

## EL LENGUAJE MODELICA



Modelica es un lenguaje para el modelado de sistemas. No es un lenguaje de programación, aunque los modelos que se construyen con Modelica se simulan en computadores: el lenguaje está diseñado para *describir las propiedades* de sistemas. El énfasis en el diseño del lenguaje se ha hecho para adecuarlo al modelado de sistemas físicos dinámicos. No obstante, su versatilidad permite usarlo para otro tipo de sistemas, tales como los económicos y sociales.

El lenguaje es utilizado por varias herramientas de *software* cuyo propósito es la simulación del comportamiento de sistemas dinámicos. En el capítulo 4 se presenta una de ellas, y en [Ass18] se puede encontrar un listado de las más populares. Estas herramientas son las encargadas, entre otras cosas, de traducir la descripción del sistema en código ejecutable por un computador. En otras palabras, aunque Modelica no es un lenguaje de programación, en combinación con las herramientas adecuadas permite la generación automática de código fuente en un lenguaje de programación<sup>1</sup>. Se trata de un lenguaje de uso abierto, no propietario, diseñado y mantenido por la Modelica Association<sup>2</sup>.

En este capítulo se exploran algunas de las características más relevantes del lenguaje Modelica. No es, por tanto, un tutorial sino una presentación descriptiva del lenguaje. En la literatura se encuentran varios documentos que sí son tutoriales. Es posible distinguir, al menos, dos tipos de ellos:

- Tutoriales enfocados en las especificaciones del lenguaje: la estructura de estos documentos está alineada con la descripción formal del lenguaje (que se encuentra en [Ass12]). Estos documentos exploran el lenguaje a partir de las definiciones de sus elementos principales, tales como: tipo de operadores, clases, arreglos, ecuaciones, funciones, etc. Los ejemplos más relevantes de este tipo de tutoriales son [Fri04], [Fri14] y [Ass00].
- Tutoriales enfocados en las funcionalidades y ejemplos del lenguaje: la organización de estos documentos hace énfasis en cómo resolver problemas de modelado. El lenguaje se presenta a través de conceptos como ecuaciones diferenciales, eventos, modelos híbridos, sistemas multido- minio, etc. Los ejemplos más relevantes son [Til01] y [Til23].

---

<sup>1</sup>En la sección 4.4 se explica como realiza este proceso un compilador específico.

<sup>2</sup>La Modelica Association es una organización no gubernamental sin ánimo de lucro cuyo propósito es el de desarrollar y promocionar el lenguaje de modelado Modelica. Se encarga también del desarrollo y el mantenimiento tanto del lenguaje Modelica como de la Modelica Standard Library. Ver [Ass25a].

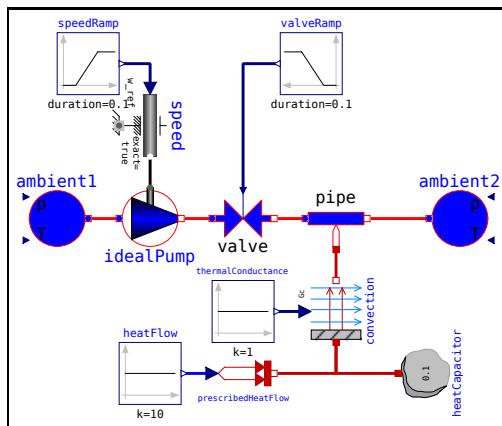
En las siguientes secciones se presentan las características más destacadas del lenguaje (sección 3.1) y algunos elementos mínimos para poder utilizarlo (sección 3.2). También se ha incluido una descripción de la librería abierta más utilizada de modelos Modelica (sección 3.3). A manera de introducción, los ejemplos 3.1 y 3.2 muestran la simulación de un modelo Modelica y un ejemplo mínimo, respectivamente.

### **Ejemplo 3.1: Simulación de un modelo Modelica.**

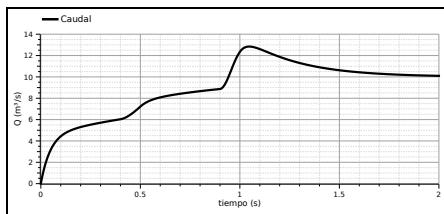
---

Uno de los ejemplos que se incluyen en la Modelica Standard Library (véase sección 3.3) se ilustra en la figura 3.1. El ejemplo consiste en el bombeo de un fluido a través de un ducto. El diagrama muestra que el fluido es impulsado por una bomba a través de una válvula para que circule por el ducto. En el ducto, además, el fluido es calentado por convección gracias a una fuente de calor.

a) Diagrama



b) Caudal



c) Temperatura

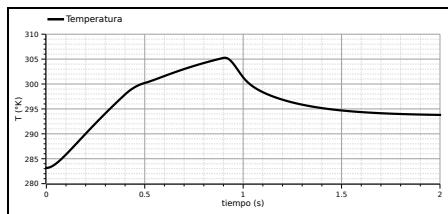


Figura 3.1 Diagrama y resultados de simulación para el modelo del ejemplo 3.1

En dicha figura se presentan también los resultados de la simulación para las variables de caudal y temperatura del fluido. La simulación se ha efectuado con la herramienta OMEdit (ver sección 4.7), incluida en la suite OpenModelica (ver capítulo 4).

El ejemplo, además, se simula bajo las siguientes condiciones:

- Temperatura ambiente de  $20^{\circ}\text{C}$ .
- La bomba funciona a la mitad de la velocidad durante  $0,4\text{s}$ ; se acelera uniformemente durante  $0,1\text{s}$  hasta llegar a la máxima velocidad. permanece a velocidad máxima durante el resto de la simulación.
- La válvula está abierta a la mitad durante  $0,9\text{s}$ ; se abre linealmente durante  $0,1\text{s}$  hasta lograr la apertura máxima, con la que permanece durante el resto de la simulación.



### *Ejemplo 3.2: Ejemplo mínimo.*

---

El código del archivo 3.1 corresponde al modelo de un sistema continuo de primer orden definido por la ecuación  $\frac{dx}{dt} = -x$ , sujeto a la condición inicial  $x(0) = 1$ :

#### Código

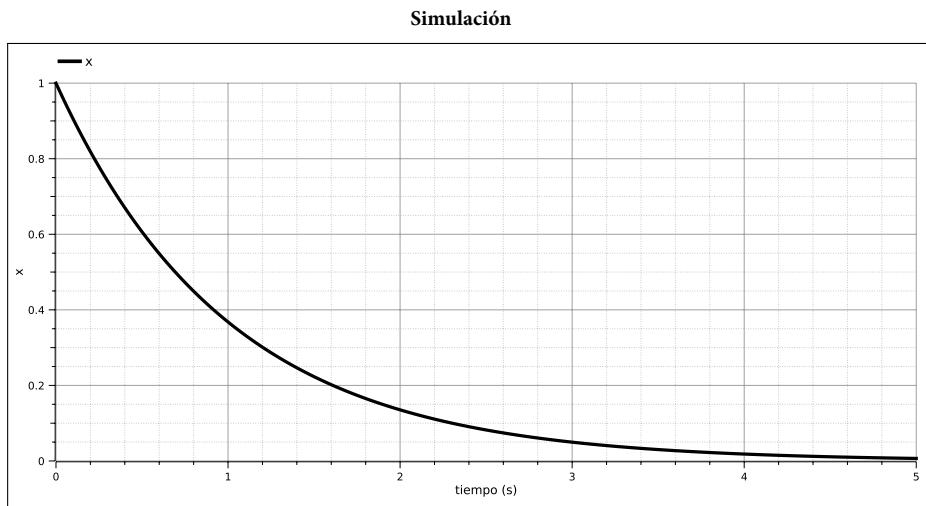
Archivo 3.1 primerOrden.mo

```
model test
  Real x(start=1);
equation
  der(x)=-x;
end test;
```

En el código anterior puede observarse:

- Un bloque delimitado por las expresiones `model test` y `end test`;
- La separación de ese bloque en dos partes delimitadas por la expresión `equation`: el encabezado y la ecuación.
- La definición en el encabezado de una variable `x` de tipo `Real`.
- Una definición de una relación matemática con la expresión `der(x)=-x`; que corresponde a la ecuación  $\frac{dx}{dt} = -x$ .

- La especificación de la condición inicial de la variable  $x$  con la expresión ( $start=1$ ).



■

### 3.1 CARACTERÍSTICAS

#### 3.1.1 Modelo de componentes y conectores

Un modelo complejo Modelica se construye *conectando* modelos más simples, tal como se ilustra en la figura 3.2. Cada *componente* del modelo está dotado de *conectores*. La *conexión* de dos o más componentes se realiza a través de esos conectores, ensamblando así el modelo mayor.

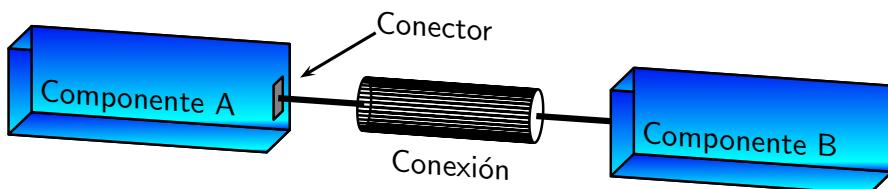


Figura 3.2 Conexión de dos componentes Modelica

El diseño de los conectores es un aspecto crítico del modelado orientado a objetos. Por una parte, para cada tipo de acople físico entre sistemas es necesario disponer de un tipo de conector. Por otra, el diseño de los conectores debe

ser lo suficientemente genérico para permitir acoplos de componentes diversos. La figura 3.3 muestra un diagrama de conexión de varios componentes, en el que se ilustra que más de dos componentes pueden estar conectados a un mismo conector.

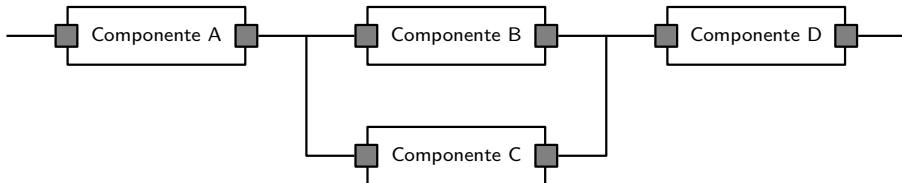


Figura 3.3 Esquema de conexión de varios componentes Modelica

En un conector se distinguen las variables *de flujo* de las que no lo son. La diferencia real entre estos tipos de variables se hace evidente cuando dos o más componentes se conectan a través de un conector:

- Con las variables de flujo se construye una ecuación del tipo *suma de variables igual a cero*.
- Con las variables que no son de flujo se construyen ecuaciones de *igualdad entre todas las variables*.

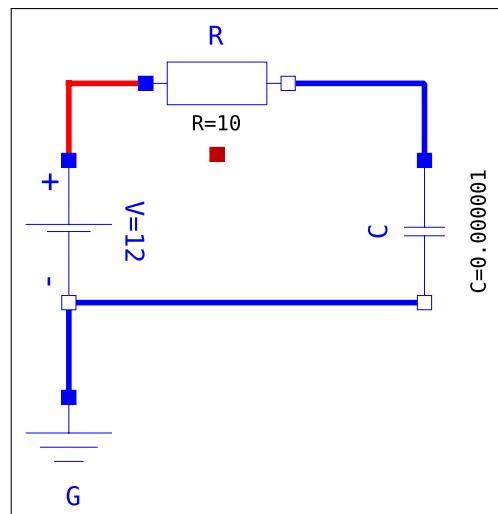
### **Ejemplo 3.3: Conexión de componentes.**

---

A continuación se muestra el diagrama de un circuito eléctrico RC alimentado por una fuente constante de tensión. En el diagrama se han destacado los conectores como pequeños cuadrados azules y blancos. Cada conector corresponde a un *terminal eléctrico*, los azules corresponden a los terminales positivos y los blancos a los negativos. Las líneas que unen los conectores representan las *conexiones*; se ha destacado en rojo una de esas conexiones, que conecta el terminal positivo de la fuente de tensión con el terminal negativo del resistor.

El archivo rc.mo muestra el código Modelica correspondiente al diagrama. Nótese que la sección equation consta de instrucciones connect(), cuya función es representar las conexiones entre componentes. La primera de esas instrucciones (connect(V.p,R.n);) corresponde a la conexión destacada en rojo en el diagrama.

Diagrama



Código

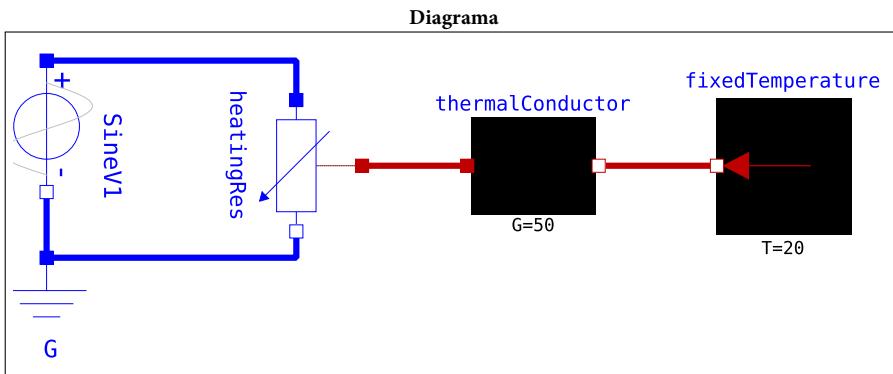
Archivo 3.2 rc.mo

```
within Modelica.Electrical.Analog;
model rc
  Basic.Ground g;
  Basic.Resistor R(R=10);
  Basic.Capacitor C(C=1e-6);
  Sources.ConstantVoltage V;
equation
  connect(V.p,R.n);
  connect(R.p,C.n);
  connect(C.p,V.n);
  connect(V.n,g.p);
end rc;
```



### Ejemplo 3.4: Conectores.

El diagrama que se presenta a continuación corresponde a uno de los ejemplos de la Modelica Standard Library, que representa el calentamiento de un cuerpo a través de una resistencia eléctrica. El modelo utiliza variables de tipo eléctrico y térmico. Las conexiones eléctricas se representan en azul, mientras que las conexiones térmicas se representan en rojo.



Las conexiones en un modelo Modelica solo pueden realizarse entre conectores del mismo tipo. No es correcto conectar, por ejemplo, un conector eléctrico con uno térmico. La razón fundamental para esta restricción es que la información asociada a cada tipo de conector es diferente. En los conectores de este ejemplo, el tipo de información asociada es la siguiente:

- En los conectores eléctricos:
  - Corriente eléctrica (variable de flujo).
  - Diferencia de potencial eléctrico (variable que no es de flujo).
- En los conectores térmicos:
  - Flujo de calor (variable de flujo).
  - Temperatura (variable que no es de flujo).

Un mismo modelo puede tener conectores de diferente tipo; tal es el caso del resistor de este ejemplo (heatingRes), que tiene dos conectores eléctricos y uno térmico. Con los primeros se conecta al resto del circuito eléctrico, mientras que con el último se conecta al sistema térmico.

### 3.1.2 Orientación

Modelica es un lenguaje *orientado a objetos* y, por tanto, comparte algunas características comunes con los lenguajes de programación orientados a objetos<sup>3</sup>, tales como encapsulamiento, herencia, polimorfismo, etc. (véase sección

<sup>3</sup>Vale la pena insistir en que Modelica es un lenguaje de modelado, no de programación.

1.2.6). En la práctica, esto significa que es posible definir *clases* y luego crear diferentes *instancias* de una misma clase, cada una de ellas con propiedades específicas. También significa que es posible crear una *jerarquía* de clases.

Como elemento particular del lenguaje, se definen *clases especializadas*, que tienen alguna especificación adicional para facilitar el modelado de algunos componentes. El listado de clases especializadas es el siguiente:

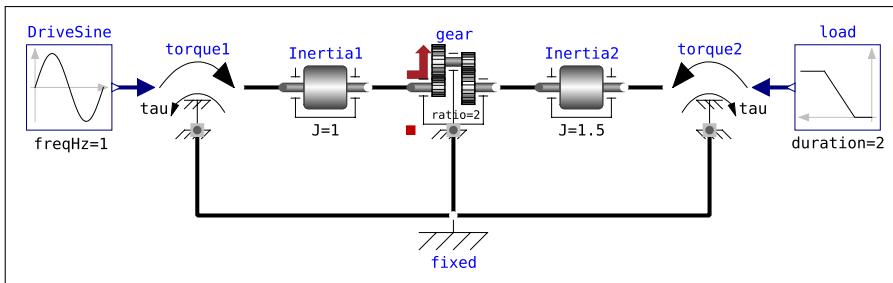
- **model:** es el tipo de clase más general. La única restricción consiste en que las instancias de estas clases no pueden usarse en conexiones.
- **record:** esta es una clase para encapsular propiedades sin comportamientos. Por tanto, no pueden definirse ecuaciones.
- **type:** se utiliza para definir nombres de clases como tipos de dato.
- **connector:** esta clase es usada específicamente para definir los conectores de los modelos. Se utilizan para almacenar información que se comparte entre dos componentes que se conectan. No pueden contener ecuaciones.
- **block:** se trata de una clase en la que la causalidad está predefinida (véase sección 3.1.4). Las variables declaradas deben acompañarse del prefijo **input** o **output**, para declarar la causalidad.
- **function:** son clases definidas para modelar el concepto de funciones matemáticas. Son semejantes a los bloques, pero no pueden definirse ecuaciones; en su lugar, se utilizan algoritmos explícitos o llamados a funciones externas.
- **package:** se utiliza para organizar espacios de nombres y librerías. Un paquete solo contiene declaraciones de clases.

### **Ejemplo 3.5: Clases e instancias.**

---

La Modelica Standard Library incluye un ejemplo que consiste en dos inercias rotacionales conectadas a través de un sistema de piñones sobre los que se aplican dos torques diferentes. El diagrama siguiente corresponde a dicho ejemplo:

Diagrama



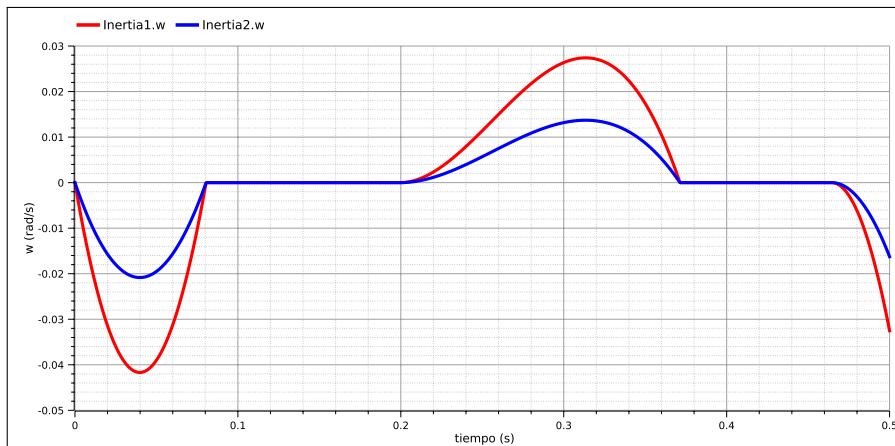
En ese diagrama pueden notarse dos componentes semejantes nombrados como `Inertia1` y `Inertia2`. Se trata de dos instancias de la misma clase. El archivo `LossyGearDemo1.mo` corresponde a la implementación del ejemplo. Allí puede notarse que los componentes `Inertia1` e `Inertia2` son ambos de la clase `Rotational.Components.Inertia`. Cada uno de los componentes tiene el atributo `J`, que adquiere valores diferentes en cada componente ( $J = 1$  en `Inertia1` y  $J = 1,5$  en `Inertia2`); en otras palabras, el parámetro  $J$  está encapsulado en cada una de las instancias.

Archivo 3.3 `LossyGearDemo1.mo`

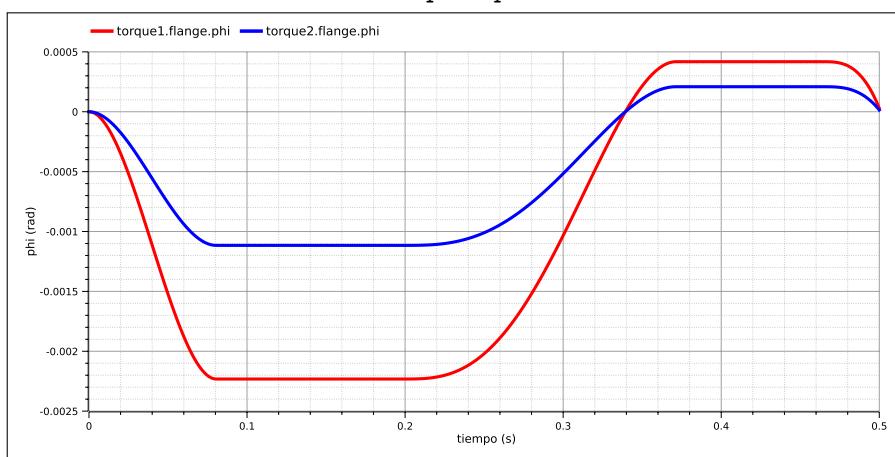
```
within Modelica.Mechanics.Rotational.Examples;
model LossyGearDemo1
  extends Modelica.Icons.Example;
  SI.Power PowerLoss = gear.flange_a.tau * der(gear.flange_a.phi)
    + gear.flange_b.tau * der(gear.flange_b.phi);
  Rotational.Components.LossyGear gear(ratio = 2, lossTable =
    [0,0.5,0.5,0,0]);
  Rotational.Components.Inertia Inertia1(J = 1);
  Rotational.Components.Inertia Inertia2(J = 1.5);
  Rotational.Sources.Torque torque1(useSupport = true);
  Rotational.Sources.Torque torque2(useSupport = true);
  Modelica.Blocks.Sources.Sine DriveSine(amplitude = 10, freqHz = 1);
  Modelica.Blocks.Sources.Ramp load(height = 5, duration = 2, offset =
    -10);
  Rotational.Components.Fixed fixed;
equation
  connect(Inertia1.flange_b, gear.flange_a);
  connect(gear.flange_b, Inertia2.flange_a);
  connect(torque1.flange, Inertia1.flange_a);
  connect(torque2.flange, Inertia2.flange_b);
  connect(DriveSine.y, torque1.tau);
  connect(load.y, torque2.tau);
  connect(fixed.flange, gear.support);
  connect(fixed.flange, torque1.support);
  connect(fixed.flange, torque2.support);
  annotation(experiment(StopTime = 0.5, Interval = 0.001));
end LossyGearDemo1;
```

El modelo también incluye dos instancias denominadas torque1 y torque2 de la misma clase Rotational.Sources.Torque. Las figuras que vienen a continuación muestran el resultado de la simulación para las variables  $w$  de los componentes Inertia1 e Inertia2 y para las variables  $\phi$  de los componentes torque1 y torque2. La diferencia en los comportamientos ilustra cómo se manifiesta el encapsulamiento de las variables simuladas.

Inercias.w



Torques.phi

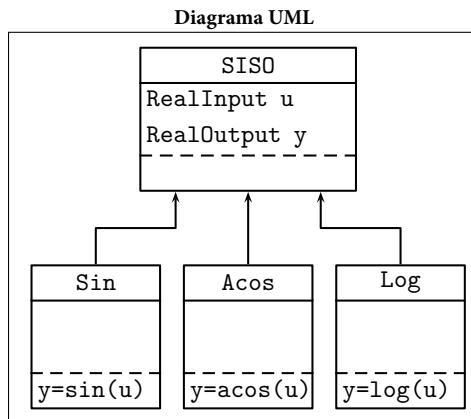


### Ejemplo 3.6: Jerarquía de clases.

En la Modelica Standard Library se definen varios bloques de funciones matemáticas. La diagrama UML que se muestra a continuación se refiere a unos pocos de esos bloques, que se podrían implementar usando el código del archivo bloquesMath.mo.

La clase SISO se define como un bloque que tiene dos parámetros: u (del tipo RealInput) e y, (del tipo RealOutput). Se trata de un bloque abstracto (debido al modificador partial) que engloba a todos los bloques que tienen una entrada Real y una salida Real.

Los bloques Acos, Sin y Log heredan del bloque SISO sus propiedades, y cada uno de ellos se especializa en una función, consistente en un cálculo aritmético específico. En el código Modelica la herencia se denota por la palabra clave extends; en el ejemplo, la especialización se encuentra en el trozo equation de cada modelo.



Código

Archivo 3.4 bloquesMath.mo

```

partial block SISO
  RealInput u;
  RealOutput y;
end SISO;

block Sin
  extends SISO;
  equation
    y = sin(u);
end Sin;

block Acos
  extends SISO;
  equation
    y = acos(u);
end Acos;

block Log
  extends SISO;
  equation
    y = log(u);
end Log;
  
```

```

y=sin(u);
end Sin;

block Acos
  extends SISO;
equation
  y=acos(u);
end Acos;

block Log
  extends SISO;
equation
  y=log(u);
end Log;

```



### 3.1.3 Multidominio

Modelica es un lenguaje de propósito general, en el sentido de que permite modelar sistemas de dominios físicos diversos. Por *dominio* entendemos aquí las distintas ramas de la ciencia y la tecnología, tales como biología, química, electricidad, mecánica, hidráulica, redes, etc.

Existen herramientas de software para simular fenómenos de cada uno de esos dominios<sup>4</sup> que se apoyan en lenguajes diseñados para describir dichos fenómenos. Estos lenguajes, no obstante, difícilmente pueden describir fenómenos de otros dominios, y por tanto las capacidades de simulación quedan necesariamente circunscritas a su dominio específico.

En su lugar, Modelica ha sido concebido para permitir el modelado y simulación de fenómenos multidominio. Un mismo modelo Modelica puede, por ejemplo, integrar fenómenos eléctricos, mecánicos y térmicos. La orientación a objetos permite construir jerarquías de clases a partir de definiciones muy generales hacia otras más específicas. Por esta razón, los modelos Modelica pueden a la vez:

- Ser tan fáciles de emplear como los modelos diseñados para herramientas de dominio específico.
- Tener la versatilidad necesaria para construir modelos multidominio.

---

<sup>4</sup>Tales como Orcad-PSpice para circuitos eléctricos (véase [Cad16]) o Aspen para procesos químicos (Véase [Asp16]).

### Ejemplo 3.7: Multidominio.

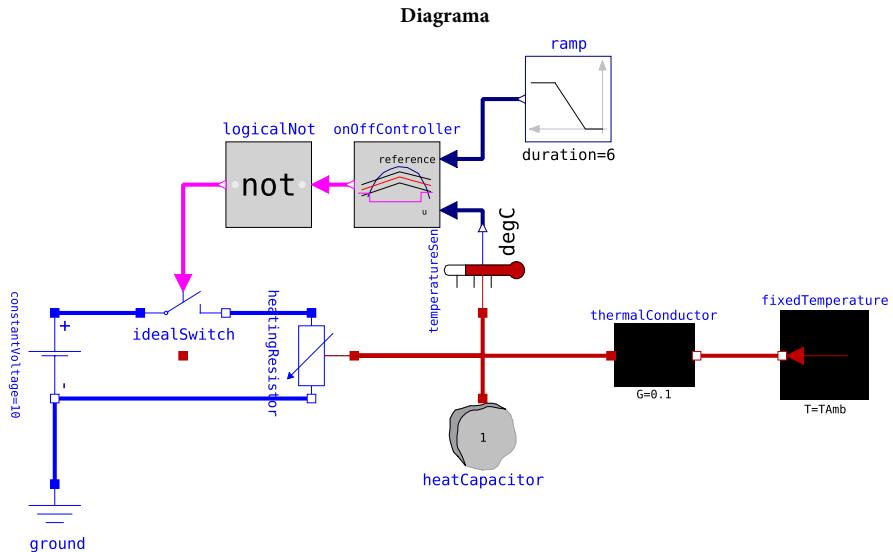
---

Considérese el problema abordado en [BS03]. Se desea construir y simular un modelo de un circuito de control de temperatura; el elemento cuya temperatura se desea controlar es una barra cilíndrica de aluminio. Los autores proponen el uso de Pspice como herramienta de diseño y simulación.

Debido a que Pspice es una herramienta de simulación de circuitos eléctricos, el modelo que se debe construir solo puede incluir elementos eléctricos. Sin embargo, el sistema contiene fenómenos térmicos y elementos de control. En consecuencia, es necesario construir modelos eléctricos equivalentes a los fenómenos no eléctricos:

- Para analizar el calentamiento en la barra de aluminio, esta se modela en seis secciones. En cada sección el modelo de transferencia de calor y calentamiento, se traduce a un circuito con resistores y capacitores.
- Los fenómenos de temperaturas constantes se traducen a fuentes de tensión constantes.
- El fenómeno de radiación de calor se traduce, en un primer modelo, a una fuente de corriente de potencia constante y, en un segundo modelo, a un bloque ABM (*Analog Behavioral Modeling*), es decir, como fuente controlada.
- La potencia entregada por el actuador (una resistencia eléctrica) se traduce a una fuente de voltaje que alimenta el circuito térmico.
- El sensor de temperatura (un puente de Wheatstone con un par de termistores) se traduce a una combinación de bloques ABM y una tabla de interpolación lineal.
- El controlador proporcional se traduce a un circuito basado en un amplificador operacional.

En contraste, considérese el controlador de temperatura que se incluye como ejemplo en la Modelica Standard Library, y cuyo diagrama se muestra a continuación:



El modelo corresponde a un sistema diferente al analizado en [BS03], pero es adecuado para ilustrar la naturaleza multidomínio de Modelica. El modelo incluye componentes eléctricos, térmicos, de procesamiento de señales y de control, cada uno de ellos modelado dentro de su dominio. Además, en un mismo componente se pueden modelar fenómenos de varios dominios. Tal es el caso de la resistencia eléctrica que calienta el sistema, en la que se simulan fenómenos eléctricos y térmicos acoplados.

■

### 3.1.4 Acausalidad

En el contexto de la simulación numérica, un modelo es *causal* si es posible clasificar de antemano las variables que definen su comportamiento como variables de entrada (conocidas) y de salida (por conocer).

Modelica permite definir modelos con relaciones causales y acausales. Para diferenciar unas de otras se utilizan dos operadores diferentes:

- El operador de igualdad = que establece relaciones *acausales*:

La expresión  $z=x+y$ ; establece una relación acausal entre las tres variables  $x$ ,  $y$ ,  $z$ . Cuando esta expresión se utilice en una simulación, dos de las variables se utilizarán para calcular la tercera. No obstante, no es posible saber de antemano qué papel cumplirá cada variable. De hecho, esto

deberá determinarse en el momento de la compilación, utilizando algún algoritmo de ordenamiento.

- El operador de asignación  $:=$  que establece relaciones *causales*:

La expresión  $z := x + y$ ; establece una relación causal entre las tres variables  $x$ ,  $y$ ,  $z$ . Esto significa que el valor de la variable  $z$  se calculará como la suma de los valores de las variables  $x$ ,  $y$ .

La acausalidad de Modelica se pone de manifiesto al notar que las siguientes expresiones son válidas y equivalentes:

$$z = x + y; \quad x + y = z; \quad z - x = y; \quad z - y = x;$$

La acausalidad es fundamental para implementar algoritmos de ordenamiento (véase sección 2.4.2). Estos algoritmos no solo ordenan las ecuaciones, sino que establecen qué variables pueden calcularse a partir de cada ecuación; en otras palabras, además de ordenar las ecuaciones, establecen la causalidad de estas. El compilador debe, entonces, estar en capacidad de despejar cualquiera de las variables en la ecuación.

En ocasiones es conveniente predefinir la causalidad de un modelo. Si esto es así, es posible emplear los modificadores *input* y *output* en la definición de las variables. El compilador debe reconocer esta condición en el algoritmo de ordenamiento.

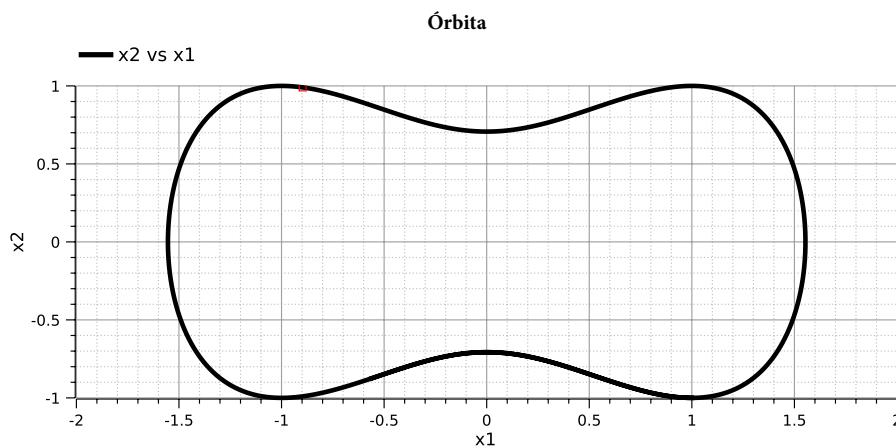
### *Ejemplo 3.8: Acausalidad.*

---

Considérese el oscilador de Duffing, que es un sistema no lineal definido por las ecuaciones 3.1:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -k * x_2 + x_1 - x_1^3 \end{aligned} \tag{3.1}$$

Este oscilador tiene un comportamiento interesante, entre otras cosas por la posibilidad de crear órbitas homoclínicas (véase, por ejemplo, [GH83]), como la que se muestra en la siguiente figura, que corresponde a las condiciones  $k = 0$ ,  $x_1(0) = 1$ ,  $x_2(0) = -1$ .



Los archivos duffing1.mo, duffing2.mo, duffing3.mo muestran tres posibles implementaciones en Modelica del oscilador.

Archivo 3.5 duffing1.mo

```
model duffing1
  Real x1(start=1);
  Real x2(start=-1);
  parameter Real k=0;
equation
  der(x1)= x2;
  der(x2)= -k*x2+x1-x1^3;
end duffing1;
```

Archivo 3.6 duffing2.mo

```
model duffing2
  Real x1(start=1);
  Real x2(start=-1);
  parameter Real k=0;
equation
  der(x1)=x2;
  -k*x2+x1-x1^3=der(x2);
end duffing2;
```

## Archivo 3.7 duffing3.mo

```

model duffing3
Real x1(start=1);
Real x2(start=-1);
parameter Real k=0;
equation
der(x1)=x2;
x1=k*x2+x1^3+der(x2);
end duffing3;

```

Pese a que las tres implementaciones contienen tres versiones distintas de la misma ecuación, el compilador genera para los tres modelos el mismo código en lenguaje C. El trozo de este código que corresponde a las ecuaciones del modelo es el siguiente:

```

void eqFunction_1(DATA *data)
{
    modelica_real tmp0;
    tmp0 = $Px1;
    $P$DER$Px2 = (((-$Pk) * $Px2) + ($Px1 - (tmp0 * tmp0 * tmp0)));
}

void eqFunction_2(DATA *data)
{
    $P$DER$Px1 = $Px2;
}

```

**Ejemplo 3.9: Causalidad.**

Los archivos gTest1.mo y gTest2.mo muestran dos implementaciones semejantes para un sistema descrito por las ecuaciones simples:

$$\begin{aligned}y &= 2u \\y &= \sin(t)\end{aligned}$$

En ambas implementaciones se ha definido un bloque que relaciona las variables  $u$ ,  $y$  a través de la ecuación  $y = 2u$ ; también se ha definido un bloque para el comportamiento de la variable  $y = \sin(t)$ , y un tercer bloque que conecta los dos anteriores.

La diferencia entre las dos implementaciones, consiste en la adición de los modificadores `input` y `output` en los dos primeros bloques. Como consecuencia de ello, el tercer bloque conecta dos variables de salida. El compilador detecta esta situación y arroja un mensaje de advertencia:

Warning: Connecting two signal sources  
while connecting G.y to S.y

Archivo 3.8 gTest1.mo

```
model ganancial
  Real u;
  Real y;
  parameter Real k=2;
equation
  y=k*u;
end ganancial;

model seno1
  Real y;
equation
  y=sin(time);
end seno1;

model gTest1
  ganancial G;
  seno1 S;
equation
  connect(G.y,S.y);
end gTest1;
```

Archivo 3.9 gTest2.mo

```
model ganancia2
  input Real u;
  output Real y;
  parameter Real k=2;
equation
  y=k*u;
end ganacia2;

model seno2
  output Real y;
equation
  y=sin(time);
end seno2;

model gTest2
  ganacia2 G;
  seno2 S;
equation
  connect(G.y,S.y);
end gTest2;
```

### 3.1.5 Modelo del tiempo

En un modelo Modelica, el tiempo se representa como una variable continua. El nombre de la variable tiempo es `time`, y la función `der()` representa la derivada respecto al tiempo. Al realizar una simulación numérica `time` se discretiza, de acuerdo con el método de integración utilizado.

No obstante, también es posible construir modelos discretos. Para ello, Modelica utiliza el concepto de *evento*. Un evento es algo que ocurre (véase sección 2.5); en la definición del lenguaje Modelica, un evento tiene las siguientes propiedades:

- Es instantáneo, su duración en el tiempo es cero.
- Tiene una *condición del evento* asociada que debe pasar de falsa a verdadera para que el evento ocurra.
- Tiene un conjunto de variables asociadas al evento, cuyo valor cambia cuando el evento ocurre.
- Tiene algún comportamiento asociado a través de ecuaciones condicionales.

La función `sample(to, dt)` está diseñada para facilitar el modelado de sistemas discretos de periodo constante. El resultado de la función cambia de falso a verdadero periódicamente y, por tanto, puede utilizarse para activar periódicamente un evento. El periodo es `dt`, y el primer cambio sucede en `to`.

Gracias al concepto de evento, también es posible construir modelos híbridos, es decir, que combinen comportamientos continuos y discretos (véase sección 2.5). En tiempo de simulación, el método numérico debe integrar las variables continuas, y actualizar las variables discretas en los instantes en que se activen los eventos.

#### *Ejemplo 3.10: Modelo discreto.*

---

La ecuación discreta de primer orden:

$$x(k+1) = ax(k) \quad a = 0,5 \quad x(0) = 1 \quad (3.2)$$

se puede modelar tal como aparece en el archivo `discreto.mo`.

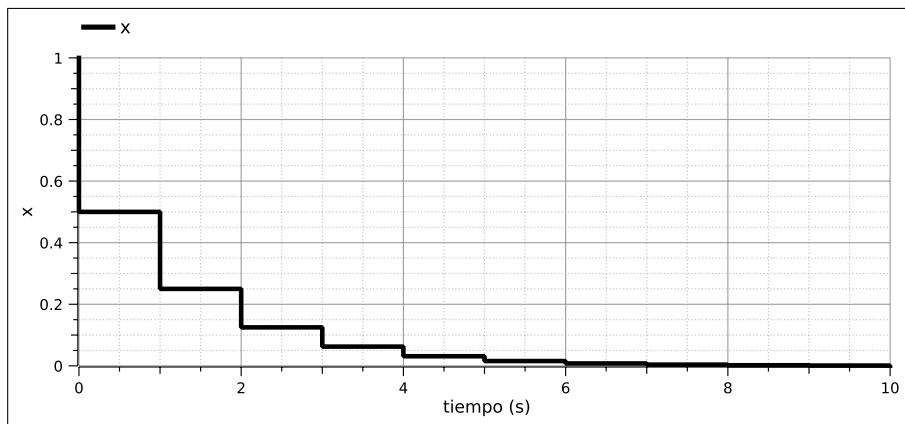
**Código**

Archivo 3.10 discreto.mo

```

model discreto
  Real x( start=1);
  parameter Real a=0.5;
equation
  when sample(0,1) then
    x=pre(x)*a;
  end when;
end discreto;

```

**Simulación**

La gráfica de arriba muestra el resultado de la simulación. Puede verse que la variable  $x$  sólo cambia en valores discretos del tiempo. La instrucción `when` detecta en qué momento la instrucción `sample(0,1)` pasa de falso a verdadero, es decir, detecta los valores de  $t = 1, 2, 3, \dots$ . En esos instantes activa la instrucción `x=pre(x)*a;` que implementa la ecuación 3.2.

**Ejemplo 3.11: Modelo híbrido.**

El código del archivo hibrido.mo implementa (en un mismo modelo) un sistema con dos variables, una continua y otra discreta:

$$\begin{aligned}
 x_1(k+1) &= ax_1(k) & a &= -1 \\
 \dot{x}_2(t) &= bx_1(k)x_2(t) & b &= 3 \\
 x_1(0) &= x_2(0) = 1;
 \end{aligned} \tag{3.3}$$

en la que  $k$  representa el tiempo en forma discreta y  $t$  en forma continua.

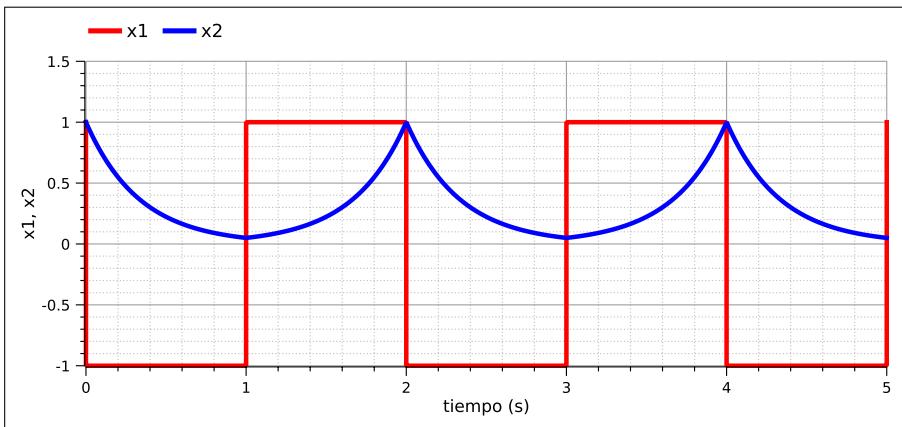
**Código**

Archivo 3.11 hibrido.mo

```

model hibrido
  Real x1(start=1);
  Real x2(start=1);
  parameter Real a=-1;
  parameter Real b=3;
equation
  when sample(0,1) then
    x1=pre(x1)*a;
  end when;
  der(x2)=b*x1*x2;
end hibrido;

```

**Simulación**

La simulación que se presenta arriba permite visualizar el comportamiento de las dos variables  $x_1$  y  $x_2$ . La primera de ellas varía de forma discreta y la segunda de forma continua.

**3.1.6 Interoperabilidad con lenguajes de programación**

Es posible incorporar en un modelo Modelica relaciones causales escritas en lenguajes de programación. Específicamente, es posible utilizar funciones externas escritas en lenguaje C o Fortran.

El compilador debe conocer la ubicación del código fuente de la función externa para poder compilarlo y enlazarlo con el programa de simulación principal. Debido a que la mayoría de herramientas de simulación basadas en Modelica generan código fuente en lenguaje C como una etapa intermedia

en el proceso de simulación (véase sección 4.3), no es necesario instalar un compilador adicional para lograr incorporar funciones externas escritas en lenguaje C.

El proceso contrario, es decir, incorporar trozos de código Modelica en programas escritos en lenguajes de programación, no está soportado. No obstante, un programador experimentado quizás pueda realizar ingeniería inversa sobre el código C que genera el simulador, para incorporarlo en su código.

### *Ejemplo 3.12: Funciones externas.*

---

Supóngase la necesidad de construir un selector de funciones, es decir, una función  $y = f(i, x) = f_i(x)$  para una colección de funciones  $f_1(x), f_2(x), \dots, f_n(x)$ . El archivo selector.mo muestra una implementación de este selector.

#### Código Modelica

Archivo 3.12 selector.mo

```
function selector
  input Integer caso;
  input Real x;
  output Real y;
  external "C" y=ext_selector(caso,x) annotation(Include="#include \
    extSelector.c");
end selector;

model selTest
  Real x,y1,y2,y3;
equation
  x=sin(time);
  y1=selector(1,x);
  y2=selector(2,x);
  y3=selector(0,x);
end selTest;
```

La función selector recibe dos parámetros: caso y x. La función devuelve la variable y, que se calcula en la función externa extSelector() escrita en lenguaje C. Esta función se muestra en el archivo extSelector.c y ha sido implementada utilizando una estructura switch-case.

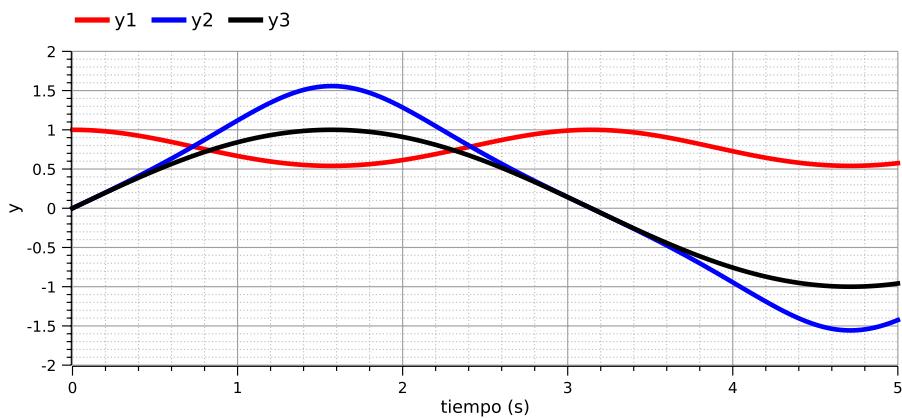
Para probar la función se ha diseñado el modelo selTest, que calcula y1, y2, y3 con tres llamados diferentes a la función selector, para el mismo valor de  $x=\sin(t)$ . El resultado de la simulación se muestra a continuación.

**Código C**

Archivo 3.13 extSelector.c

```
#include <math.h>

double ext_selector(int caso, double x)
{
    double y=0;
    switch(caso)
    {
        case 1: y = cos(x); break;
        case 2: y = tan(x); break;
        default: y = x; break;
    }
    return y;
}
```

**Simulación****3.1.7 Documentación de los modelos**

Cada modelo Modelica puede contener su propia documentación interna a través de la palabra clave annotation. Los dos usos principales de esta capacidad son:

- Incorporar una descripción textual del modelo, sus características y forma de uso. Esta descripción se guarda en formato HTML.
- Incorporar una descripción gráfica del modelo, en forma de ícono. Esta descripción se hace a través de primitivas gráficas propias del lenguaje.

Las herramientas diseñadas para la construcción y edición de modelos Modelica pueden aprovechar estas dos capacidades de documentación para facilitar la labor del modelador.

### **Ejemplo 3.13: Documentación textual y gráfica.**

---

El diagrama del circuito RC del ejemplo 3.3 se ha obtenido con la herramienta gráfica OMEdit utilizando el código del archivo rcOMedit.mo. Las instrucciones annotation se han utilizado para definir los elementos gráficos del modelo.

#### Código Modelica

Archivo 3.14 rcOMedit.mo

```
within Modelica.Electrical.Analog;
model rc
  Basic.Ground G annotation(Placement(visible = true, transformation(
    origin = {-60,-5}, extent = {{-12,-12},{12,12}}, rotation = 0)));
  annotation(Diagram());
  Sources.ConstantVoltage V(V = 12) annotation(Placement(visible = true,
    transformation(origin = {-60,35}, extent =
    {{-12,12},{12,-12}}, rotation = -90));
  Basic.Capacitor C(C = 0.000001) annotation(Placement(visible = true,
    transformation(origin = {0,35}, extent = {{12,-12},{-12,12}},
    rotation = -270));
  Basic.Resistor R(R = 10) annotation(Placement(visible = true,
    transformation(origin = {-35,60}, extent = {{12,12},{-12,-12}},
    rotation = -180));
equation
  connect(V.p,R.n) annotation(Line(points =
    {{-60,47},{-60,60},{-47,60}}));
  connect(R.p,C.n) annotation(Line(points = {{-23,60},{0,60},{0,47}}));
  connect(V.n,C.p) annotation(Line(points = {{-60,23},{0,23}}));
  connect(V.n,G.p) annotation(Line(points = {{-60,23},{-60,7}}));
end rc;
```



### **3.1.8 Reusabilidad**

Modelica ha sido diseñado para potenciar la reutilización de componentes. La orientación a objetos, la acausalidad y el modelo de componentes-conectores son la columna vertebral del lenguaje que lo permite.

En la práctica, esta característica se manifiesta en la existencia de librerías de componentes. En [Ass07] se puede consultar una amplia colección de

librerías, tanto comerciales como libres. Entre de estas librerías se destaca la Modelica Standard Library, que se presenta en la sección 3.3.

No todas las librerías, sin embargo, son 100 % funcionales. No es extraño encontrar en la red librerías que han perdido vigencia, debido a los cambios en las especificaciones del lenguaje al pasar de una versión a otra. También es usual encontrar librerías que solo funcionan adecuadamente con algunas herramientas de simulación<sup>5</sup>.

### *Ejemplo 3.14: Librerías.*

---

El archivo osciladores/package.mo muestra la implementación de una librería de osciladores. La instrucción package empaqueta las definiciones. La librería tiene una modelo abstracto oscilador, y tres modelos herederos especializados para implementar el oscilador lineal, el oscilador de Duffing y el oscilador de Van der Pol (véase [GH83]).

El archivo oscTest.mo es un modelo para probar la librería. Contiene tres osciladores diferentes (uno de cada modelo de la librería), cuyo comportamiento se muestra en los retratos de fase de abajo, obtenidos a partir de la simulación. Obsérvese el uso de la sintaxis osciladores . X para referirse al modelo X incluido en la librería osciladores.

#### Paquete

Archivo 3.15 osciladores/package.mo

```
package osciladores

partial model oscilador
  Real x1(start=1);
  Real x2(start=1);
equation
  der(x1)=x2;
end oscilador;

model lineal
  extends oscilador;
equation
  der(x2)=-x1;
end lineal;

model duffing
  extends oscilador;
```

---

<sup>5</sup>Para los usuarios de OpenModelica (véase capítulo 4), es desafortunado que no todas las librerías disponibles se puedan compilar adecuadamente con el compilador omc (véase sección 4.4). En [Con07] puede verse el análisis de compatibilidad de una amplia colección de librerías respecto a OpenModelica.

```

parameter Real k=0;
equation
  der(x2) = -k*x2 + x1 -x1^3;
end duffing;

model vanderpol
  extends oscilador;
  parameter Real mu=1;
equation
  der(x2)=mu*(1-x1^2)*x2-x1;
end vanderpol;

end osciladores;

```

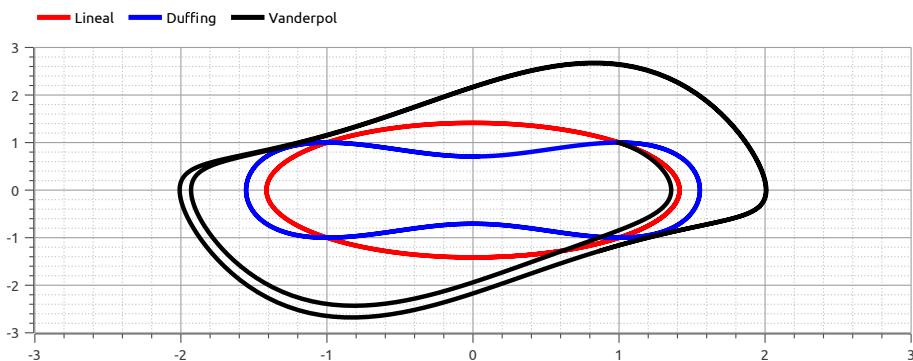
**Modelo**

Archivo 3.16 oscTest.mo

```

model oscTest
  osciladores.lineal O1;
  osciladores.duffing O2;
  osciladores.vanderpol O3;
end oscTest;

```

**Retratos de fase****3.2 ALGUNOS ELEMENTOS DEL LENGUAJE**

En esta sección se presentan algunos elementos del lenguaje que pueden ayudar a ilustrar su potencial de descripción, y que no han sido presentados en los ejemplos de la sección 3.1. Tales elementos son:

- Tipos de datos: las definiciones de los tipos de datos son más ricas que las usuales en lenguajes de programación. El tipo de dato `Real`, por ejemplo, es en realidad una clase que tiene los siguientes atributos:

- **value:** es el valor en punto flotante de doble precisión de la variable.
- **quantity:** nombre de la cantidad física que representa; por ejemplo “Longitud”.
- **unit:** unidades en que se mide la cantidad; por ejemplo “m”.
- **displayUnit:** unidades utilizadas para desplegar información; por ejemplo “km”.
- **min:** valor mínimo que puede tomar la variable.
- **max:** valor máximo que puede tomar la variable.
- **start:** valor inicial fijo o candidato<sup>6</sup>.
- **fixed:** valor booleano que especifica si el parámetro start es un valor inicial fijo o candidato.
- **nominal:** valor nominal de la variable.
- **stateSelect:** variable auxiliar para diferenciar estados.

Adicionalmente, la definición de `Real` incluye dos ecuaciones del tipo `assert` que sirven como alertas que se activan cuando el valor de la variable está por fuera de los límites establecidos por los atributos `min` y `max`.

De forma semejante, los tipos de datos `Integer`, `Boolean` y `String` son clases con atributos. La clase especializada `type` permite, además, definir nuevos tipos de datos que enriquecen el lenguaje.

b. Arreglos de datos: en la terminología de Modelica se utilizan cuatro designaciones diferentes para referirse al concepto de *arreglo de datos*:

- Escalares: datos simples.
- Vectores: matrices unidimensionales.
- Matrices: matrices bidimensionales.
- Arreglos: matrices con  $k$  dimensiones ( $k > 0$ ).

El lenguaje es rico en instrucciones para manipulación de arreglos<sup>7</sup>. Cuenta con funciones que permiten:

- Declarar, dimensionar y redimensionar arreglos.
- Gestionar los datos en los arreglos.

---

<sup>6</sup>Candidato para obtener un conjunto de condiciones iniciales consistente.

<sup>7</sup>Aunque no es un lenguaje orientado a matrices, como pueden serlo los lenguajes Matlab o Scilab.

- Aplicar operaciones sobre los datos de los arreglos.
  - Operar arreglos entre sí.
  - Aplicar operaciones algebraicas sobre los arreglos.
- c. Ecuaciones condicionales e iterativas: en Modelica las ecuaciones definen las relaciones entre las variables, por ejemplo, a través de la instrucción `connect()`. En ese sentido, están estrechamente relacionadas con la estructura del modelo.

En Modelica es posible modelar ecuaciones, es decir conexiones de componentes, apoyados en estructuras del tipo `for`, `if-then-else`, `while`, `when`. Esta propiedad permite construir modelos cuya estructura se decide en línea mediante iteraciones, condicionales o eventos, lo que incrementa la flexibilidad descriptiva del lenguaje.

### 3.3 LA MODELICA STANDARD LIBRARY

La Modelica Standard Library es un paquete desarrollado y mantenido por la *Modelica Association*. Contiene la definición de un conjunto amplio de constantes, unidades, funciones, tipos, conectores, modelos parciales, modelos y ejemplos de componentes.

La librería se distribuye de forma libre con un licenciamiento establecido por la *Modelica license*. El propósito de este licenciamiento es:

[...] que las librerías de modelos, software, imágenes, documentos, archivos de datos, etc. puedan ser usados libremente en su forma original o en una forma modificada, en ambientes de código abierto y en ambientes comerciales. (véase [Ass25b]).

La tabla 3.1 muestra el conjunto de paquetes y subpaquetes incluidos en la versión 3.2.1. de la Modelica Standard Library. Para elaborar la tabla se han tomado solamente los tres primeros niveles de complejidad del árbol de la librería. El número de clases, registros, modelos, etc., varía de un paquete a otro. Para tener una idea del tamaño de la librería, considérese que se trata de una colección de archivos de texto plano, cuyo tamaño es de cerca de 30Mb.

Tabla 3.1 Paquetes y subpaquetes de la Modelica Standard Library

Blocks	Continuous	Internal
	Discrete	
	Examples	BusUsage_Utilitys
	Icons	
	Interaction	Show
	Interfaces	Adaptors
	Logical	
	Math	UnitConversions
	MathBoolean	
	MathInteger	
	Nonlinear	
	Routing	
	Sources	
	Tables	
	Types	
ComplexBlocks	ComplexMath	
	Examples	
	Interfaces	
	Sources	
ComplexMath	Vectors	
Constants		
Electrical	Analog	Basic
		Examples
		Ideal
		Interfaces
		Lines
		Semiconductors
		Sensors
		Sources
	Digital	Basic
		Converters
		Delay
		Examples
		Gates
		Interfaces
		Memories
		Multiplexers
		Registers
		Sources
		Tables
		Tristates
		UsersGuide
		BasicMachines
		Examples
		Icons
		Interfaces

Electrical	Machines	Losses
		Sensors
		SpacePhasors
		Thermal
		Utilities
	MultiPhase	Basic
		Examples
		Ideal
		Interfaces
		Sensors
		Sources
	QuasiStationary	Machines
		MultiPhase
		SinglePhase
		Types
		UsersGuide
	Spice3	Additionals
		Basic
		Examples
		Interfaces
		Internal
		Semiconductors
		Sources
		Types
		UsersGuide
		HeatTransfer
Fluid	Dissipation	PressureLoss
		UsersGuide
		Utilities
		AST_BatchPlant
	Examples	ControlledTankSystem
		DrumBoiler
		Explanatory
		HeatExchanger
		Tanks
		TraceSubstances
		BaseClasses
	Fittings	Bends
		GenericResistances
		Orifices
		Icons
		Interfaces
	Machines	BaseClasses
	Pipes	BaseClasses
	Sensors	BaseClasses
	Sources	BaseClasses
	Types	

Fluid	UsersGuide	BuildingSystemModels
		ComponentDefinition
	Utilities	
	Valves	BaseClasses
Icons	Vessels	BaseClasses
	Library	
	Library2	
Magnetic	Package	
	FluxTubes	Basic
		Examples
		Interfaces
		Material
		Sensors
		Shapes
		Sources
		UsersGuide
	FundamentalWave	BasicMachines
		Components
		Examples
		Interfaces
		Sensors
Math	Matrices	Sources
		Types
		UsersGuide
		BooleanVectors
		Icons
	Nonlinear	Examples
		LAPACK
		Utilities
	Vectors	Examples
		Interfaces
Mechanics	MultiBody	Utilities
		Examples
		Forces
		Frames
		Icons
		Interfaces
		Joints
		Parts
		Sensors
		Types
	Rotational	UsersGuide
		Visualizers
		Components
		Examples
		Icons
		Interfaces

Mechanics	Rotational	Sensors
		Sources
		UsersGuide
	Translational	Components
		Examples
		Interfaces
		Sensors
		Sources
	Media	DryAirNasa
		MoistAir
		SimpleAir
		OneNonLinearEquation
		ThermoFluidSpecial
		Common
		LinearColdWater
		LinearWater_pT_Ambient
		SolveOneNonlinearEquation
		TestOnly
		Tests
		TwoPhaseWater
		Common
		MixtureGases
	Incompressible	SingleGases
		Common
		Examples
		TableBased
		Choices
		PartialCondensingGases
		PartialLinearFluid
		PartialMedium
		PartialMixtureMedium
		PartialPureSubstance
	Interfaces	PartialSimpleIdealGasMedium
		PartialSimpleMedium
		PartialTwoPhaseMedium
		TemplateMedium
		MediumDefinition
		MediumUsage
		Water
		Constantinopolitano
		IF97_Utilsities
		IdealSteam
	Water	StandardWater
		StandardWaterOnePhase
		WaterIF97OnePhase_ph
		WaterIF97_R1pT
		WaterIF97_R1ph
		WaterIF97_R2pT

Media	Water	WaterIF97_R2ph
		WaterIF97_R3ph
		WaterIF97_R4ph
		WaterIF97_R5ph
		WaterIF97_base
		WaterIF97_fixedregion
		WaterIF97_pT
		WaterIF97_ph
SIunits	Conversions	NonSIunits
	Icons	
	UsersGuide	
StateGraph	Examples	Utilities
	Interfaces	
	Temporary	
	UsersGuide	
Thermal	FluidHeatFlow	Components
		Examples
		Interfaces
		Media
		Sensors
		Sources
	HeatTransfer	Celsius
		Components
		Examples
		Fahrenheit
		Interfaces
		Rankine
		Sensors
		Sources
UsersGuide	Conventions	Documentation
		ModelicaCode
		UsersGuide
	ReleaseNotes	
Utilities	Examples	
	Files	
	Internal	PartialModelicaServices
	Streams	
	Strings	Advanced
	System	
	Types	
	UsersGuide	



# 4

## LA SUITE OPENMODELICA



La simulación de modelos preparados en lenguaje Modelica requiere el uso intensivo de herramientas de *software*. En [Ass18] se listan varias de estas herramientas de compilación, simulación y visualización, algunas de las cuales son propietarias y otras son libres. Como parte de la oferta de herramientas libres, destaca OpenModelica por su grado de maduración, soporte y desarrollo permanente.

OpenModelica nació en el laboratorio de ambientes de programación de la Universidad de Linköpings (Suecia), gracias al trabajo pionero de Peter Fritzson (véase [Uni18, FE98]). En la actualidad, el *Open Source Modelica Consortium* (OSMC) lidera el proceso de desarrollo y soporte de un conjunto de herramientas que se distribuyen juntas y que conforman la *suite* (ver [Con08a]).

En las siguientes secciones se hace una presentación de las principales herramientas de la *suite* que sirven para el desarrollo y simulación de modelos<sup>1</sup>. El propósito del capítulo es ilustrar el potencial de uso de la *suite* OpenModelica. Además, en la sección 4.3 se explica cómo es el proceso de compilación de modelos Modelica, ya que su comprensión puede facilitar la interpretación de algunos de los mensajes de error del compilador y, en general, la tarea de modelado.

#### 4.1 ARQUITECTURA

La figura 4.1 muestra el esquema de comunicación entre los diferentes módulos de OpenModelica que operan bajo el esquema cliente/servidor. La función de cada uno de ellos es la siguiente<sup>2</sup>:

- Manejador de sesión interactiva: interpreta y traslada comandos del sistema e instrucciones Modelica que permiten la evaluación y simulación de modelos, así como la visualización de resultados. Su utilización se explica en la sección 4.2.
- Compilador omc: genera el código binario ejecutable a partir del código fuente en lenguaje Modelica. Este sofisticado proceso se expli-

---

<sup>1</sup>Este capítulo no pretende ser un extenso manual de usuario de la *suite*. Para tener información detallada sobre el uso de cada una de las herramientas de la suite pueden consultarse [FPA<sup>+</sup>13a, Con08b, FPA<sup>+</sup>13b].

<sup>2</sup>Una descripción extensa de estos módulos se encuentra en [FPA<sup>+</sup>13b].

ca en la sección 4.3, mientras que la estructura del módulo se describe en la sección 4.4.

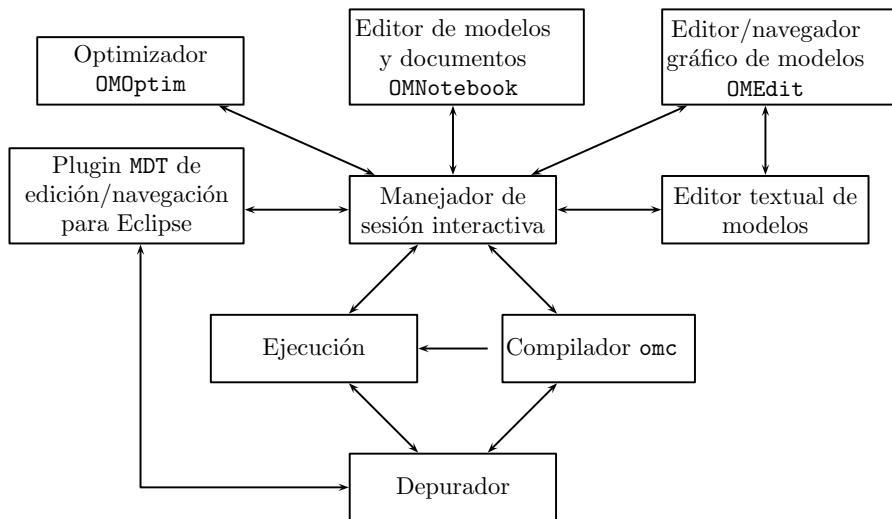


Figura 4.1 Interacción de los módulos de la suite OpenModelica. Adaptado de [FPA<sup>+</sup>13b]

- Ejecución: la ejecución del archivo compilado realmente es responsabilidad del sistema operativo. En la sección 4.5 se explica este proceso.
- Editor textual de modelos: la construcción del código fuente de forma textual puede realizarse a través de cualquier editor de textos convencional. La *suite* no provee ningún editor de textos especial.
- Editor de modelos y documentos OMNotebook: permite la elaboración de textos simples en los que se pueden insertar bloques de simulación y graficación de resultados. La sección 4.6 muestra la forma de utilizarlo.
- Editor/navegador gráfico de modelos OMEedit: se trata de una herramienta gráfica que permite la edición de modelos Modelica, la navegación a través de librerías y modelos, y la visualización de resultados de simulación. Su utilización se presenta en la sección 4.7.
- *Plugin* MDT de edición/navegación para Eclipse: Eclipse es un ambiente de desarrollo de *software* ampliamente utilizado. Este *plugin* permite el desarrollo de modelos Modelica en dicho ambiente, la compilación de los modelos y la visualización de los resultados. En la sección 4.8 se ilustra su utilización.

- Depurador: se utiliza también a través de Eclipse. Para ello se cuenta con un plugin adicional.
- Optimizador OMOptim: herramienta basada en heurísticas para la búsqueda de los parámetros de un modelo Modelica que minimicen una función objetivo incluida en dicho modelo.
- Otros módulos: existen otros módulos y desarrollos (aún en consolidación) que no se han incorporado en la figura 4.1.

## 4.2 LA CONSOLA OMShell

La figura 4.2 muestra una captura de pantalla de la consola OMShell. A través de esta consola es posible interactuar con el sistema operativo y con el compilador omc. La tabla 4.1 presenta el listado de los comandos disponibles más frecuentemente usados, con una breve descripción de su utilidad<sup>3</sup>. Dada la importancia del comando simulate, se ha preparado la tabla 4.2 que muestra los argumentos con los que se puede controlar las condiciones de simulación. Para ilustrar el potencial de la consola, a continuación se presentan los ejemplos de uso 4.1 a 4.3:

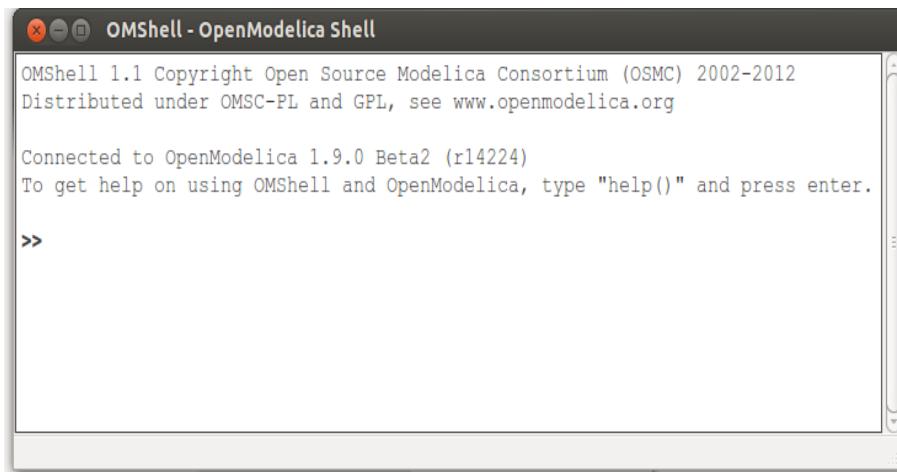


Figura 4.2 Captura de pantalla de la consola OMShell

<sup>3</sup>Para una descripción completa de la sintaxis, los argumentos y el valor returnedo de cada comando en la tabla, puede consultarse [FPA<sup>+</sup>13a] (sección 1.3). Una descripción de otros comandos se encuentra en las secciones de 2.3. y 2.4.3. de [FPA<sup>+</sup>13b].

Tabla 4.1 Comandos disponibles más usuales en una sesión interactiva

Nombre	Funcionalidad
cd	Cambia el directorio de trabajo
clear	Borra todas las definiciones de memoria
clearVariables	Borra las definiciones de variables de la memoria
dumpXMLDAE	Exporta información de un modelo a formato XML
exportDAEtoMatlab	Exporta la matriz de incidencia de un modelo en formato legible por Matlab
help	Despliega ayuda sobre el uso del compilador omc
instantiateModel	Crea una instancia del código y retorna una cadena con la definición plana
list	Retorna el listado de una o todas las clases definidas
listVariables	Retorna el listado de las variables definidas
loadFile	Carga un archivo Modelica
loadModel	Carga un modelo o paquete almacenado en el directorio OPENMODELICALIBRARY
plot	Grafica una o más variables simuladas contra el tiempo
plotParametric	Grafica una variable simulada en función de otra
quit	Salida y cierre de la consola
readFile	Lee un archivo
runScript	Ejecuta un script
saveModel	Guarda un modelo
simulate	Traduce un modelo y lo simula (ver tabla 4.2)
system	Traslada una orden al sistema operativo
timing	Evaluá una expresión y retorna el tiempo empleado
typeOf	Determina de qué tipo es una variable
val	Calcula el valor interpolado de una variable simulada en un instante de tiempo específico

***Ejemplo 4.1: Interpretación de comandos Modelica.***

En la consola pueden escribirse directamente comandos en lenguaje Modelica; por ejemplo, para definir un arreglo x con los enteros del 1 al 5 puede digitarse:

```
>> x:=1:5
```

y en la consola se despliega el resultado:

```
>> x:=1:5
{1,2,3,4,5}
```

pueden realizarse operaciones sobre la variable definida:

```
>> x*2
{2,4,6,8,10}
```

■  
Tabla 4.2 Principales argumentos del comando `simulate`

Argumento	Tipo	Opcional	Valor por defecto	Descripción
<code>className</code>	<code>TypeName</code>	SI		Nombre del modelo a simular
<code>startTime</code>	<code>Real</code>	NO	0.0	Tiempo de inicio de la simulación
<code>stopTime</code>	<code>Real</code>	NO	1.0	Tiempo de finalización de la simulación
<code>numberOfIntervals</code>	<code>Integer</code>	NO	500	Número de intervalos de tiempo en el archivo de resultados
<code>method</code>	<code>String</code>	NO	'dassl'	Método de integración a emplear. Ver tabla 4.3
<code>tolerance</code>	<code>Real</code>	NO	0.000001	Tolerancia para el método de integración
<code>outputFormat</code>	<code>String</code>	NO	'mat'	Formato de salida del archivo de resultados. Ver sección 4.5.2
<code>fileNamePrefix</code>	<code>String</code>	NO	"	Prefijo para nombrar los archivos. Si se omite se asume igual a <code>className</code>
<code>variableFilter</code>			'*'	Filtro para decidir qué variables se almacenan en el archivo de resultados. Cada variable se separa con ' '. El valor por defecto significa almacenar todas las variables.
<code>cflags</code>	<code>String</code>	NO	"	Banderas adicionales para el compilador de C.
<code>storeInTemp</code>	<code>Boolean</code>	NO	false	Uso del directorio temporal.
<code>measureTime</code>	<code>Boolean</code>	NO	false	Activa el análisis de desempeño. Genera archivos <code>html</code> , <code>xml</code> y <code>svg</code> con información sobre el tiempo de compilación y ejecución con un reloj en tiempo real.

### Ejemplo 4.2: Definición en línea de modelos Modelica.

---

Pueden definirse modelos cortos escribiendo en línea su especificación en lenguaje Modelica. Por ejemplo, para modelar un sistema de primer orden simple  $\dot{x} = -x$  con la condición inicial  $x(0) = 1$  se puede escribir el siguiente modelo:

```
>> model test Real x(start=1);equation der(x)=-x;end test;  
{test}
```

para luego simular su comportamiento:

```
>> simulate(test);
```

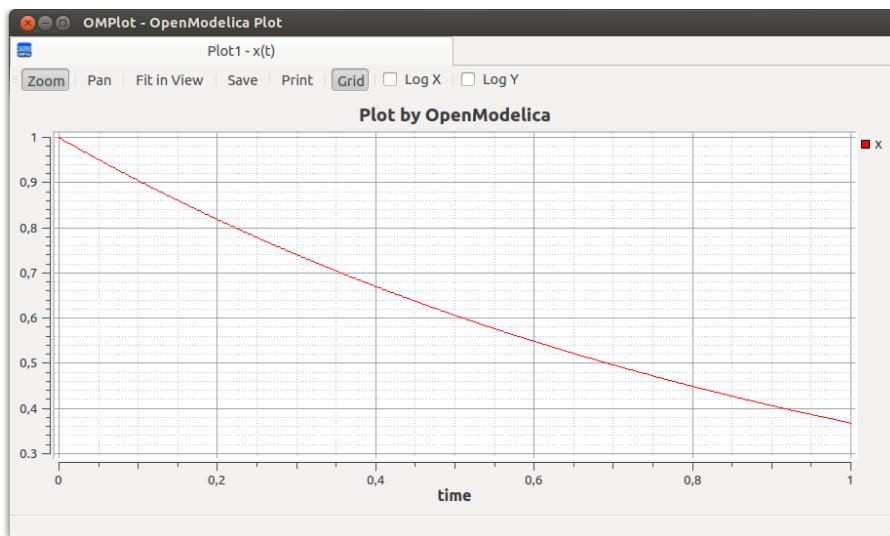
Para conocer el valor de la variable x para el instante de tiempo 0,5, se emplea la función val:

```
>> val(x,0.5)  
0.6065310076919237
```

Mientras que para visualizar el comportamiento de la variable x a lo largo de toda la simulación se puede emplear la instrucción plot:

```
>> plot(x);
```

que arroja como resultado:



### **Ejemplo 4.3: Uso de librerías y modelos externos.**

---

Para cargar la librería Modelica Standard Library se escribe:

```
>> loadModel(Modelica)
```

Y para cargar un archivo de nombre `miarchivo.mo`:

```
>> loadFile("miarchivo")
```

Si la sintaxis es correcta, se desplegará el mensaje `true`, de lo contrario se informará la línea y columna del archivo en donde se detectó el error.

Puede ser conveniente cambiar el directorio de trabajo para facilitar la ubicación de los archivos que se desean cargar:

```
>> cd("nuevodirectorio")
```



### **4.3 EL PROCESO DE COMPILACIÓN**

La figura 4.3 ilustra el proceso típico llevado a cabo en la simulación de un archivo Modelica. Puede verse que la misión del compilador `omc` es la de convertir el modelo preparado en el *lenguaje de descripción* Modelica en un nuevo código que es escrito en el *lenguaje de programación* C. Este nuevo código se convierte en un programa ejecutable, utilizando un compilador externo de C<sup>4</sup>.

Mediante un llamado al sistema operativo, se lanza el programa ejecutable. Este programa lee algunas condiciones de simulación desde un archivo de entrada, simula el proceso, y guarda los resultados de la simulación en un archivo de salida (en la sección 4.5 se presentan más detalles de este proceso). De nuevo, mediante un llamado al sistema operativo, se invoca el programa de visualización que permite desplegar en forma gráfica los resultados de la simulación.

El código generado en lenguaje C utiliza un conjunto de librerías que se instalan en el sistema operativo en el proceso de instalación de OpenModelica. Estas librerías implementan, entre otras cosas, los métodos numéricos de solución de ODEs y DAEs; la tabla 4.3 presenta los métodos de integración disponibles en la versión 1.9.0. del compilador `omc`. Las librerías externas también se encargan de gestionar el proceso general de simulación, así como de la lectura y escritura de los archivos de entrada y salida. Algunas de estas librerías emplean, a su vez, la librería LAPACK++ (véase [OST14]).

---

<sup>4</sup>OpenModelica utiliza `gcc`.

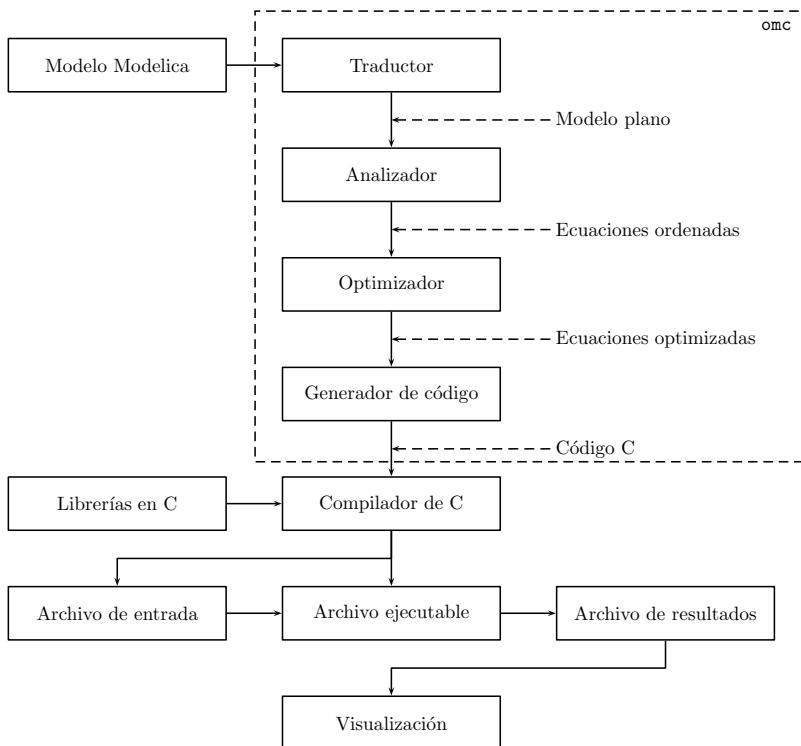


Figura 4.3 Proceso de compilación, ejecución y visualización con OpenModelica

La misma figura 4.3 muestra que la generación de código C requiere una serie de etapas intermedias, que se explican a continuación:

- Traducción: en esta etapa se extrae la representación de los modelos, clases, bloques, conectores, funciones, etc., escritos en lenguaje Modelica para generar una colección de ecuaciones (diferenciales, algebraicas y simbólicas) en forma *plana*. Al realizar esta traducción, también se construye una representación abstracta simbólica de las ecuaciones.
- Análisis: la representación abstracta de la colección de ecuaciones permite la construcción de la matriz de incidencia, y, a partir de ella, el procesamiento mediante los algoritmos de Tarjan, Rasgadura, Pantelides, etc., para establecer una nueva colección de ecuaciones ordenadas.

Tabla 4.3 Métodos de integración disponibles en la versión 1.9.0. del compilador omc

Nombre	Orden	Control de paso	Control de orden	Anotaciones
dassl	1-5	SI	SI	Es el método de integración por defecto. Corresponde al método Adams Moulton con Jacobiano numérico y búsqueda de intervalo raíces
dasslsteps	1-5	SI	SI	Es el mismo método dassl, pero ignora las definiciones de <code>numberOfIntervals</code> y <code>stepSize</code> .
dasslwort	1-5	SI	SI	dassl; sin búsqueda de intervalo raíces
euler	1	NO	NO	Método de euler explícitos.
rungekutta	4	NO	NO	Método clásico de Runge-Kutta.
radau1	1	NO	NO	Método Radau IIA con un puntos.
radau3	3	NO	NO	Método Radau IIA con dos puntos.
radau5	5	NO	NO	Método Radau IIA con tres puntos
lobatto2	2	NO	NO	Método de lobatto IIIA con dos puntos
lobatto4	4	NO	NO	Método de lobatto IIIA con tres puntos
lobatto6	6	NO	NO	Método de lobatto IIIA con cuatro puntos

- c. Optimización: las ecuaciones resultantes se analizan para establecer si es posible realizar simplificaciones en ellas, reemplazando expresiones triviales por formas más compactas.
- d. Generación de código: las ecuaciones optimizadas se implementan en lenguaje C; también se generan los archivos de compilación (makefile) que serán leídos por el sistema operativo con las instrucciones adecuadas de compilación.

El ejemplo 4.4 ilustra la transformación que sufre el código Modelica hasta convertirse en código en lenguaje C.

#### **Ejemplo 4.4: Traducción de código Modelica.**

---

El código Modelica del archivo rc.mo implementa un circuito simple RC:

## Archivo 4.1 rc.mo

```

within Modelica.Electrical.Analog;
model rc
  Basic.Ground g;
  Basic.Resistor R(R=10);
  Basic.Capacitor C(C=1e-6);
  Sources.ConstantVoltage V;
equation
  connect(V.p,R.n);
  connect(R.p,C.n);
  connect(C.p,V.n);
  connect(V.n,g.p);
end rc;

```

Con el comando `LoadFile(``rc.mo'')` se carga el archivo en el compilador. El comando `instantiateModel(rc)` muestra el modelo plano del circuito. En ese modelo las ecuaciones son las siguientes:

```

g.p.v = 0.0;
assert(1.0 + R.alpha * (R.T_heatPort - R.T_ref)
>= 0.00000000000001, "Temperature outside scope of model!");
R.R_actual = R.R * (1.0 + R.alpha * (R.T_heatPort - R.T_ref));
R.v = R.R_actual * R.i;
R.LossPower = R.v * R.i;
R.v = R.p.v - R.n.v;
O.O = R.p.i + R.n.i;
R.i = R.p.i;
R.T_heatPort = R.T;
C.i = C.C * der(C.v);
C.v = C.p.v - C.n.v;
O.O = C.p.i + C.n.i;
C.i = C.p.i;
V.v = V.V;
V.v = V.p.v - V.n.v;
O.O = V.p.i + V.n.i;
V.i = V.p.i;
g.p.i + C.p.i + V.n.i = O.O;
R.p.i + C.n.i = O.O;
R.n.i + V.p.i = O.O;
R.n.v = V.p.v;
C.n.v = R.p.v;
C.p.v = V.n.v;
C.p.v = g.p.v;

```

Después del proceso de optimización y ordenamiento, el sistema DAE queda reducido a:

- 1)  $R.v = C.v - V.V$
- 2)  $C.i = R.v / R.R\_actual$
- 3)  $R.LossPower = R.v * C.i$
- 4)  $\text{der}(C.v) = C.i / C.C$

El código en lenguaje C que implementa estas ecuaciones se encuentra dentro del archivo rc.c.

Archivo 4.2 rc.c

```
void eqFunction_1 (DATA * data)
{
    $PR$Pv = ((-$PC$Pv) - $PV$PV);
}

void eqFunction_2 (DATA * data)
{
    modelica_real tmp14;
    tmp14 = DIVISION($PR$Pv, $PR$PR_actual, _OMC_LIT2);
    $PC$Pi = tmp14;
}

void eqFunction_3 (DATA * data)
{
    $PR$PLossPower = ($PR$Pv * $PC$Pi);
}

void eqFunction_4 (DATA * data)
{
    modelica_real tmp15;
    tmp15 = DIVISION($PC$Pi, $PC$PC, _OMC_LIT3);
    $P$DER$PC$Pv = tmp15;
}
```



## 4.4 EL COMPILADOR OMC

El proceso de compilación de OpenModelica se ejecuta a través del programa `omc`, que puede considerarse como el núcleo de la suite completa.

### 4.4.1 Estructura interna del compilador

De acuerdo con la información disponible en el repositorio del código fuente (véase [Con08a]), la estructura interna del compilador tiene dos módulos principales que agrupan a más de cincuenta submódulos especializados:

- a. FrontEnd: encargado de traducir el modelo Modelica en un modelo plano.
- b. BackEnd: encargado de la generación del código C a partir del modelo plano.

El módulo BackEnd se subdivide en varios módulos, con las tareas específicas que se desarrollan de forma secuencial, y que se relacionan a continuación:

■ **Análisis y optimización del modelo acausal:**

- Evaluar y remplazar los parámetros de tipo `final` y los parámetros con la anotación `Evaluate=true`.
- Simplificar las ecuaciones tipo `if`.
- Remplazar los llamados a funciones `equal`.
- Particionar los bloques independientes.
- Expandir el operador de derivación `dér`.
- Encontrar el orden del sistema (número de estados).
- Remplazar los llamados a las funciones `edge` y `change`.
- Aplanar las ecuaciones de arreglos.
- Remover la mayoría de las ecuaciones simples.

■ **Causalización del modelo:**

- Revisar si el sistema es singular.
- Aplicar el algoritmo de Pantelides para reducción de orden.
- Introducir las funciones `inline`.
- Seleccionar los estados.
- Aplicar el algoritmo de Tarjan para ordenamiento de ecuaciones.
- Analizar los componentes fuertemente conectados.

■ **Análisis y optimización del modelo causal:**

- Simplificar el sistema lineal constante.
- Simplificar los llamados a la función `semiLinear`.
- Remover la totalidad de las ecuaciones simples.
- Encapsular las condiciones `when`.
- Aplicar el algoritmo de rasgado de subsistemas.

■ **Generación de código C**

#### 4.4.2 Uso del compilador

El compilador puede usarse directamente a través de comandos de consola, o utilizando el manejador de sesiones interactivas, bien sea con la consola OMShell o con alguna otra interfaz. El uso directo mediante consola consiste fundamentalmente en la generación del código C de un modelo Modelica, como se ilustra en los ejemplos 4.5 a 4.7.

##### *Ejemplo 4.5: Generación de código C a partir de un archivo simple.*

---

Supóngase que el archivo test.mo contiene el siguiente código Modelica:

Archivo 4.3 test.mo

```
model test
  Real x(start=1);
equation
  der(x) = -x;
end test;
```

La ejecución del comando:

omc +s test.mo

lanzará el compilador, que generará los siguientes archivos, que estarán listos para ser compilados por el compilador de C:

```
test.c
test_functions.c
test_functions.h
_test.h
test_init.xml
test.makefile
test.mo
test_records.c
```

Para compilar estos archivos se utiliza el comando:

make - f test.makefile

que genera el archivo ejecutable test.



##### *Ejemplo 4.6: Generación de código C a partir de un archivo que usa la librería Modelica.*

---

Supóngase que el archivo rc.mo contiene el siguiente código Modelica:

## Archivo 4.4 rc.mo

```

within Modelica.Electrical.Analog;
model rc
  Basic.Ground g;
  Basic.Resistor R(R=10);
  Basic.Capacitor C(C=1e-6);
  Sources.ConstantVoltage V;
equation
  connect(V.p,R.n);
  connect(R.p,C.n);
  connect(C.p,V.n);
  connect(V.n,g.p);
end rc;

```

El modelo anterior utiliza definiciones de la Modelica Standard Library. Para generar el código C de ese modelo, es necesario que el compilador cargue la librería. Para ello, se utiliza la instrucción:

```
omc +s rc.mo Modelica
```

Y para compilar dicho código:

```
make -f rc.makefile
```

*Ejemplo 4.7: Instrucciones Modelica.*

El compilador omc también puede usarse para ejecutar una secuencia de instrucciones Modelica. Supóngase que el archivo rc.mos (véase ejemplo 4.6) tiene por contenido las siguientes instrucciones Modelica:

## Archivo 4.5 rc.mos

```

loadModel(Modelica);
loadFile("rc.mo");
simulate(rc);

```

Para ejecutar esas instrucciones se utiliza el comando:

```
omc rc.mos
```

Estas instrucciones permiten la simulación del modelo rc que se define en el archivo rc.mo. El compilador cargará la Modelica Standard Library, el archivo rc.mo, creará el código C, lo compilará y lanzará el archivo ejecutable.



## 4.5 EJECUCIÓN DEL MODELO COMPILADO

La figura 4.4 es una versión reducida de la figura 4.3. En esta se destaca que el resultado de la compilación es un archivo ejecutable que lee un archivo de entrada y genera un archivo de resultados. El nombre del archivo ejecutable corresponde al prefijo utilizado en la compilación (argumento `fileNamePrefix`, tabla 4.2)<sup>5</sup>. En las siguientes secciones se explica la estructura de los archivos de entrada y salida.

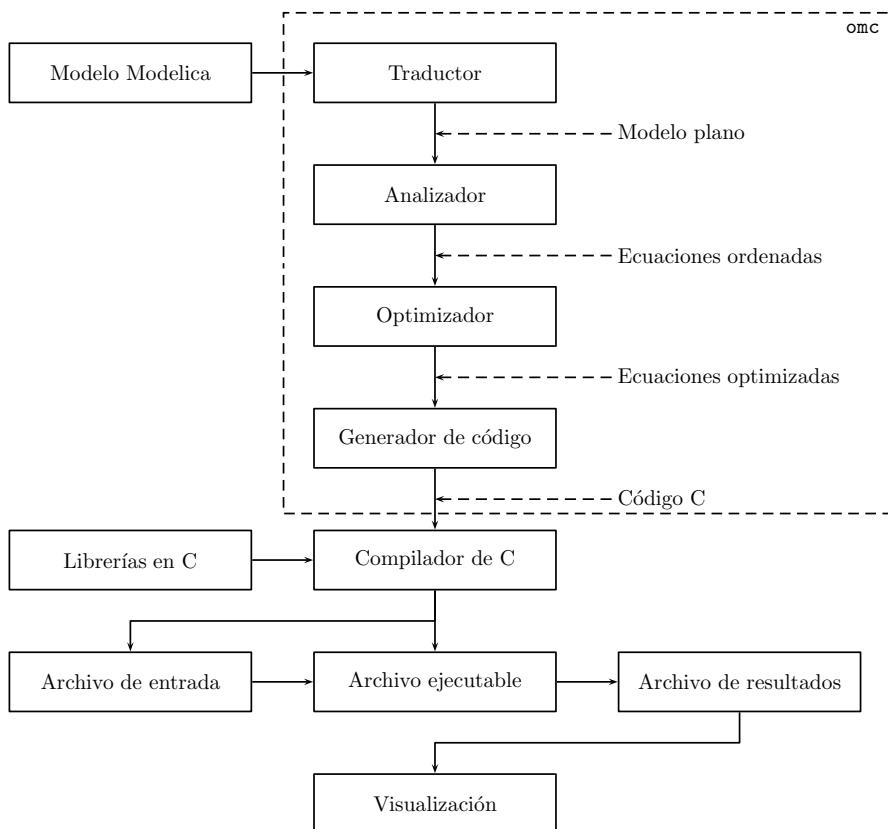


Figura 4.4 Interacción de algunos archivos en una simulación de OpenModelica

<sup>5</sup>En Windows el nombre toma la extensión .EXE.

#### 4.5.1 Archivo de entrada

El archivo de entrada está escrito en lenguaje XML y tiene dos características importantes:

- El nombre del archivo termina en `_init.xml`. El inicio del nombre corresponde al prefijo utilizado en la compilación (argumento `fileNamePrefix`, tabla 4.2).
- La estructura del archivo sigue las definiciones del *Functional Mockup Interface* (FMI) (véase [Gro17]).

El FMI es un estándar para intercomunicar herramientas de simulación (véase [Con10, BOA<sup>+11</sup>]). No es un estándar propietario de una herramienta específica. El propósito del estándar es facilitar dos tareas: el intercambio de modelos y la cosimulación.

Una de las definiciones del estándar FMI es el esquema de descripción de modelos. OpenModelica lo emplea para construir el archivo de entrada al que se refiere la figura 4.4. La información que contiene este esquema es la siguiente:

- Atributos generales del modelo, tales como nombre, número de estados continuos, número de eventos, autor, etc.
- Definición de las variables visibles del modelo. Se trata de una descripción detallada de cada variable, que incluye nombre, tipo, valor inicial, valores mínimos y máximos, etc.
- Definición del experimento por defecto, con datos tales como tolerancia y tiempo de inicio y parada.

#### *Ejemplo 4.8: Archivo de entrada.*

---

El trozo del archivo `test_init.xml` que corresponde a la ejecución del modelo del ejemplo 4.2, en el que se describe el experimento por defecto, es el siguiente:

```
<DefaultExperiment
  startTime      = "0.0"
  stopTime       = "1.0"
  stepSize       = "0.002"
  tolerance      = "0.000001"
  solver         = "dassl"
  outputFormat   = "mat"
  variableFilter = ".*" />
```

En ese mismo archivo, el trozo en el que se describe la variable x es:

```
<ScalarVariable
  name = "x"
  valueReference = "1000"
  variability = "continuous" isDiscrete = "false"
  causality = "internal"
  alias = "noAlias"
  classIndex = "0" classType = "rSta"
  fileName = "&lt;interactive&gt;" startLine = "1" startColumn = "12"
  endLine = "1" endColumn = "26" fileWritable = "true">
<Real useStart="true" start="1.0" fixed="false" useNominal="false"
  nominal="1.0" min="-1.7976931348623157E+308"
  max="1.7976931348623157E+308"  />
</ScalarVariable>
```



#### 4.5.2 Archivo de resultados

OpenModelica puede generar archivos de resultados en tres formatos distintos (argumento `outputFormat` en la tabla 4.2):

- Formato mat: formato binario utilizado por el programa Matlab (véase [Mat14]).
- Formato csv: formato de texto plano que almacena una matriz. Las características del formato son las siguientes:
  - El separador es la coma.
  - Cada línea corresponde a un mismo tiempo de simulación.
  - Cada columna corresponde a una variable simulada.
  - La primera línea contiene el nombre de las variables. Cada nombre se escribe entre comillas dobles.
- Formato plt: formato de texto específico para la herramienta de graficación plot, incluida en OpenModelica<sup>6</sup>. Algunas de sus características son:
  - Las primeras líneas son un encabezado descriptivo.
  - Cada variable almacenada genera un *DataSet*.
  - Cada *DataSet* es un conjunto de líneas de texto.

---

<sup>6</sup>Utiliza el proyecto Ptolemy de la Universidad de Berkeley (véase [Dep17]).

- La primera línea de un *DataSet* describe el nombre de las variables.
- Las líneas siguientes de cada *DataSet* contienen parejas de datos separados por coma. El primer dato corresponde a la variable tiempo, y el segundo a la variable almacenada.

#### **Ejemplo 4.9: Archivo de resultados en formato csv.**

---

Al simular el modelo del ejemplo 4.2 con formato de salida csv, se genera el archivo `test_res.csv`. Las primeras filas de ese archivo, correspondientes a la primera centésima de segundo de simulación, son:

```
"time","x","der(x)",
0,1,-1,
0.002,0.9980019984073524,-0.9980019984073524,
0.004,0.9960079871985149,-0.9960079871985149,
0.006,0.9940179539149203,-0.9940179539149203,
0.008,0.9920318976611787,-0.9920318976611787,
0.01,0.9900497885986662,-0.9900497885986662,
```



#### **Ejemplo 4.10: Archivo de resultados en formato plt.**

---

Al simular el modelo del ejemplo 4.2 con formato de salida plt, se genera el archivo `test_res=plt`. Las primeras filas de ese archivo, correspondientes al encabezado, son:

```
#Ptolemy Plot file, generated by OpenModelica
#NumberofVariables=3
#IntervalSize=502
TitleText: OpenModelica simulation plot
XLabel: t
```

Las primeras líneas del *DataSet* de la variable x, correspondientes a la primera centésima de segundo de simulación, son:

```
DataSet: x
0, 1
0.002, 0.9980019984073524
0.004, 0.9960079871985149
0.006, 0.9940179539149203
0.008, 0.9920318976611787
0.01, 0.9900497885986662
```



## 4.6 EL EDITOR DE MODELOS Y DOCUMENTOS OMNOTEBOOK

OMNotebook es una herramienta de modelamiento y documentación. En ese sentido, es una aplicación del concepto de *Literate Programming* que consiste en la producción simultánea de código y documentación (véase [Knu84]). Utilizando OMNotebook es posible construir un documento (un texto) en el que se inserta y ejecuta código Modelica.

Las capacidades de edición de texto son limitadas en comparación con las de los procesadores de texto convencionales. No obstante, la posibilidad de incorporar código ejecutable dentro del documento hace que esta herramienta sea muy atractiva para el desarrollo de textos interactivos.

Un documento de OMNotebook es un conjunto de celdas. Una celda es un espacio rectangular que ocupa el ancho del documento. Cada celda puede ser de dos tipos:

- Celdas de texto: en estas celdas se puede editar y dar formato (alineación, tipo y tamaño de *font*, etc.) a texto. También se pueden insertar imágenes. OMNotebook cuenta con algunas facilidades de dibujo de diagramas.
- Celdas de entrada de código: en estas celdas se inserta código Modelica e instrucciones para el compilador omc. Para que el código sea procesado por el compilador, el usuario debe *evaluar* la celda, bien mediante una opción en el menú, o mediante la combinación de teclas *Shift + Enter*.

Varias celdas pueden formar un grupo, y los grupos pueden anidarse entre sí. Además, cada grupo puede expandirse o colapsarse. Esta funcionalidad permite la construcción de documentos estructurados (en capítulos, secciones y subsecciones, por ejemplo), que se navegan de forma interactiva. La suite OpenModelica provee dos textos elaborados sobre OMNotebook:

- a. DrModelica: tutorial sobre el lenguaje Modelica.
- b. DrControl: texto guía para un curso de teoría de control.

### *Ejemplo 4.11: OMNotebook.*

---

La figura 4.5 muestra la apariencia de OMNotebook con una implementación del ejemplo 4.2. Se han usado tres celdas de texto y tres celdas de entrada de código, intercaladas entre sí. El propósito da cada una de las celdas de entrada de código es, respectivamente:

- Introducir el código Modelica del modelo.

- Simular el modelo.
- Visualizar los resultados.

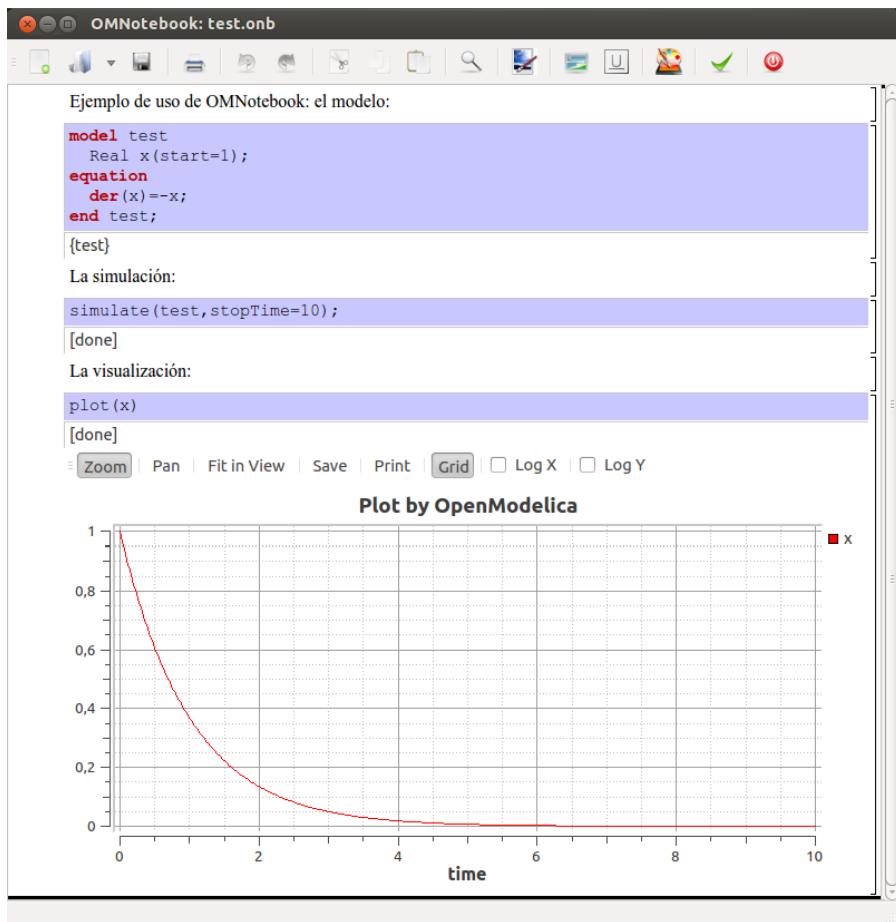


Figura 4.5 Apariencia de OMNotebook con una implementación del ejemplo 4.2

## 4.7 EL EDITOR GRÁFICO OMEDIT

OMEdit es una herramienta de edición y simulación de modelos Modelica. La figura 4.6 muestra las ventanas de la aplicación, que se describen a continuación:

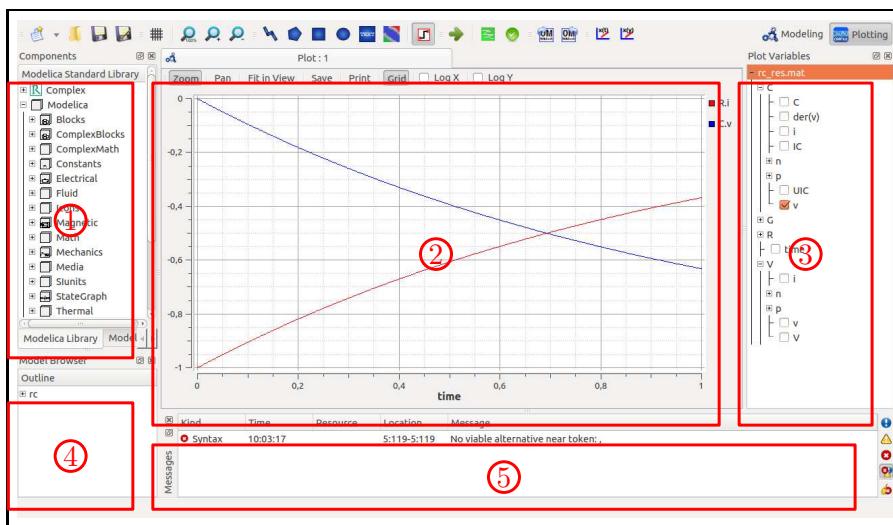


Figura 4.6 Ventanas de OMEdit

- 1. Ventana de librerías:** permite la navegación a través del árbol de las librerías. OMEdit carga por defecto la Modelica Standard Library. El usuario puede adicionar otras librerías.
- 2. Ventana del diseñador:** permite la edición de modelos y la visualización de resultados. La figura 4.7 muestra la apariencia de la aplicación en tres modos diferentes:
  - Edición gráfica: en este modo, el usuario puede arrastrar modelos desde la ventana de librerías y conectarlos entre sí usando una herramienta gráfica. También, puede complementar el modelo con textos y objetos gráficos, tales como líneas, rectángulos, elipses, etc. Este modo permite también la edición de íconos asociados al modelo.
  - Edición textual: en este modo, el usuario puede escribir directamente el código Modelica que define el modelo.
  - Visualización: en este modo, se despliegan las curvas del comportamiento de las variables simuladas que seleccione el usuario.
- 3. Ventana de variables graficadas:** muestra el árbol de todas las variables simuladas, para permitir que el usuario seleccione las que desea graficar.

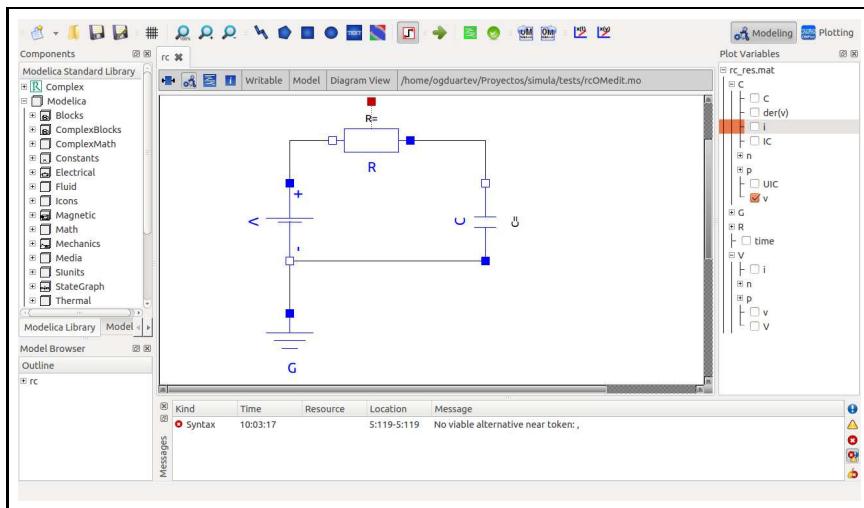
4. **Ventana de navegación del modelo:** muestra el árbol de los componentes del modelo, con información específica de cada uno de los parámetros.
5. **Ventana de mensajes:** despliega los mensajes provenientes de la aplicación o del compilador.

**Ejemplo 4.12: OMEdit.**

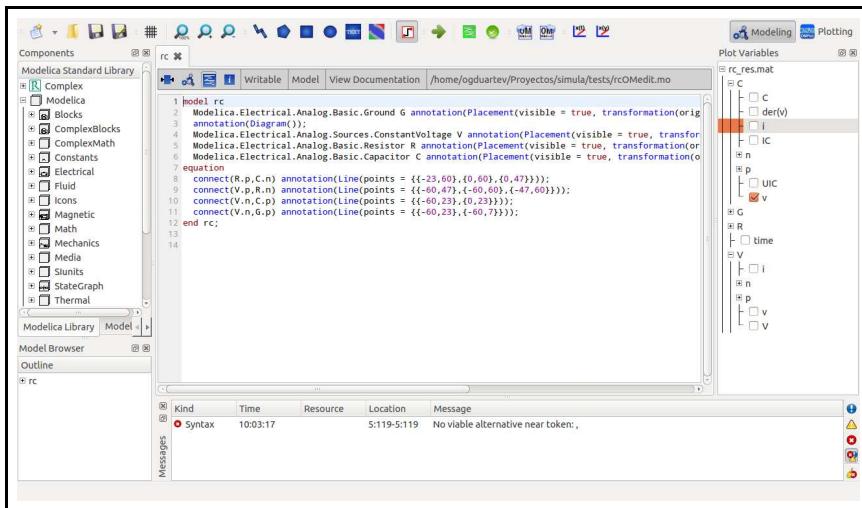
---

La figura 4.7 muestra la apariencia de OMEdit para una implementación del modelo del ejemplo 4.4. El modelo se ha construido usando el modo gráfico; las coordenadas se han ajustado en modo de texto; se han visualizado dos variables simultáneamente: la corriente en la resistencia y la diferencia de potencial en el condensador.

Modo de edición gráfico



## Modo de edición de texto



## Visualización de resultados

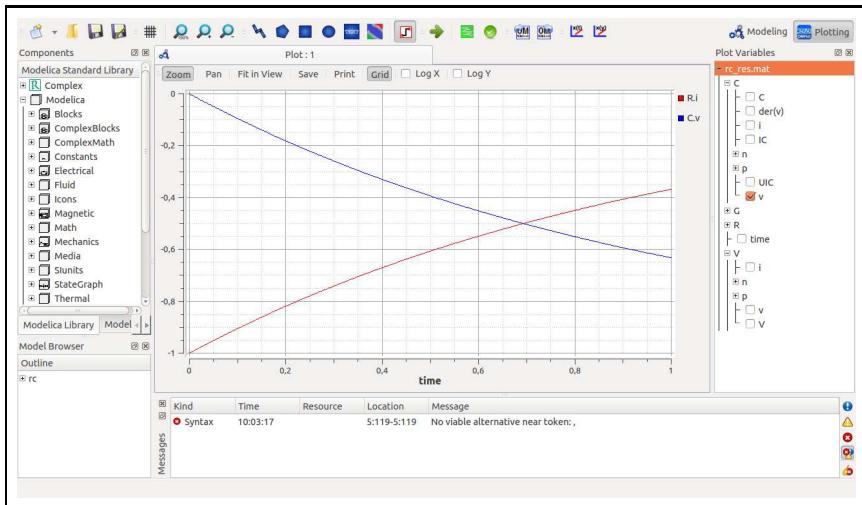


Figura 4.7 Apariencia de OMEedit con una implementación del ejemplo 4.4

## 4.8 EL PLUGIN PARA MDT ECLIPSE

Eclipse es un plataforma modular de desarrollo de software (véase [Fou18]). Inicialmente fue concebida como un ambiente de desarrollo de programas en lenguaje Java; no obstante, su modularidad le permite utilizar *plugins* para adecuar la plataforma a diferentes lenguajes de programación<sup>7</sup>.

La suite OpenModelica incluye el *plugin* Modelica Development Toolkit (MDT) que facilita el desarrollo de modelos Modelica. Específicamente provee las siguientes funcionalidades (véase [FPA<sup>+</sup>13a]):

- Navegación a través de árboles de proyectos, paquetes y clases.
- Asistentes para la creación de proyectos, paquetes y clases.
- Resaltado de sintaxis en colores.
- Revisión de sintaxis.
- Navegación a través del árbol de la Modelica Standard Library y otras librerías.
- Asistente para escritura de nombres de clases, tipos y funciones.
- Enlaces a la definición de clases, tipos y funciones.
- Despliegue de información sobre el tipo de dato de variables y parámetros.
- Despliegue de la consola Modelica.

---

**Ejemplo 4.13: El plugin MDT para Eclipse.**

---

La figura 4.8 muestra la apariencia de Eclipse para una implementación del modelo del ejemplo 4.2 usando el *plugin* MDT. El modelo se ha escrito en el editor de texto; para compilar el modelo se ha abierto una consola Modelica que se comunica con omc; la visualización se ha efectuado mediante un llamado a la instrucción plot desde esa misma consola.

---

<sup>7</sup>El *plugin* para Modelica tiene algunas dificultades de operación en sistema operativo Linux. La definición de las variables de ambiente OPENMODELICAHOME y OPENMODELICALIBRARY soluciona algunas de ellas. Una buena combinación de trabajo consiste en utilizar eclipse-mdt para editar los modelos y OMShell para simularlos.

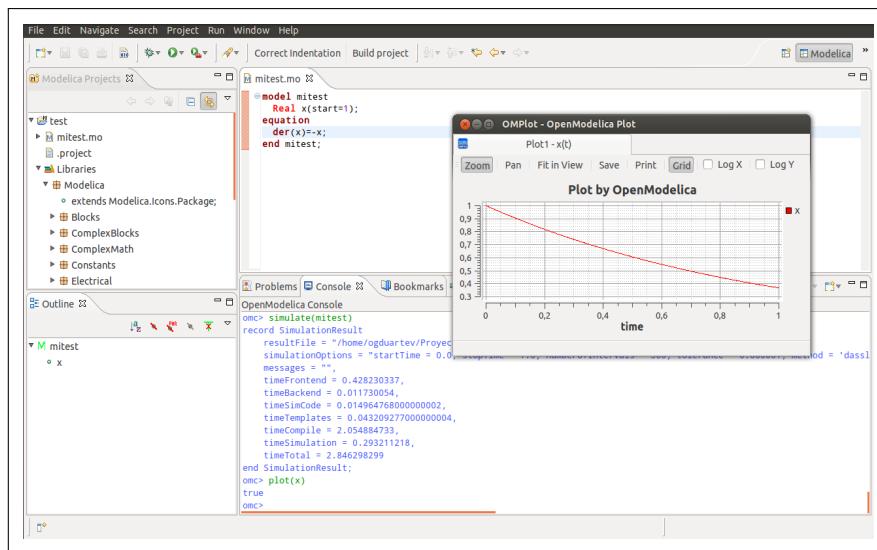


Figura 4.8 Apariencia de eclipse con una implementación del modelo del ejemplo 4.2

## 4.9 OTROS MÓDULOS Y DESARROLLOS

El desarrollo de la *suite* OpenModelica es permanente. Dentro de las herramientas disponibles hay varias que no son de utilidad directa para el presente documento, o que no han alcanzado aún un grado de maduración suficiente. Algunas de ellas son:

- La herramienta de optimización OmOptim: herramienta independiente que utiliza heurísticas para optimizar modelos. El enfoque consiste en buscar cuál es el valor de algunos parámetros seleccionados por el usuario que minimizan alguna función objetivo incorporada en el modelo. En la actualidad sólo se ha incorporado la heurística de algoritmos genéticos.
- La interfaz para python OmPython: manejador de sesiones interactivas escrito en lenguaje python. Permite llamar desde un programa python al compilador omc.
- Depuración con ModelicaML: utilizando las opciones de depuración del compilador de C, se puede hacer un rastreo de la evolución de las variables del modelo, incluida la pila del sistema. El método, no obstante, es

dispendioso. El *plugin* MDT GDB para eclipse, facilita en algo la depuración. Este plugin funciona sobre un conjunto extendido de instrucciones denominado ModelicaML.

- Simulaciones interactivas: las simulaciones interactivas permiten modificar parámetros del modelo en tiempo de ejecución. El compilador omc puede generar ejecutables con esta funcionalidad.
- Cosimulación con la interfaz FMI: la cosimulación consiste en la simulación de un sistema complejo a través de la simulación de sus componentes por diferentes herramientas de *software* que trabajan de manera coordinada. El estándar *Functional Mockup Interfaz* (FMI) se ha desarrollado con ese propósito. OpenModelica tiene un desarrollo incipiente de cosimulación, que consiste en la adecuación de sus archivos de entrada al estándar (ver sección 4.5.1).

# **Parte II**

# **UNVirtualLab**



# 5

## LABORATORIOS VIRTUALES. ESTADO DEL ARTE



En este capítulo se hace un compendio de algunos aspectos importantes asociados a los laboratorios virtuales a partir de una revisión de la literatura técnica disponible. Para ello, la sección 5.1 enumera las principales características asociadas al término genérico laboratorio virtual; estas características sirven para proponer una clasificación en la sección 5.2; en la sección 5.3, por su parte, se discute la importancia de la experimentación en el proceso de aprendizaje de la ingeniería, mientras que en la sección 5.4 se destacan algunos aspectos a tener en cuenta en el diseño de laboratorios virtuales.

## 5.1 CARACTERÍSTICAS DE LOS LABORATORIOS VIRTUALES

En general, y en un sentido amplio, el término laboratorio virtual se usa para referirse a una amplia variedad de soluciones de *hardware* y *software* que permiten realizar experimentos sobre modelos matemáticos de la realidad ([SOHA12]). No obstante, en ocasiones se emplea el término también para referirse a la manipulación remota de elementos físicos presentes en un laboratorio *real* (p. ej., en [GR09] y [AKB<sup>+</sup>01]).

En medio de la heterogeneidad de los laboratorios virtuales reportados en la literatura, es posible identificar algunas características que distinguen a unos de otros, las cuales se exponen a continuación.

### 5.1.1 Orientación a la web

Para algunos autores, la disponibilidad en la web es una característica intrínseca de los laboratorios virtuales (p. ej., [SDGL10]), mientras que otros autores utilizan el término para referirse también a soluciones *autónomas* o *de escritorio* (p. ej., [Mar07]).

Un laboratorio orientado a la web está disponible a través de internet. Generalmente el usuario accede a él a través de navegadores convencionales. Por el contrario, el laboratorio autónomo (*stand-alone*) se instala directamente en el equipo del usuario. La figura 5.1 ilustra los dos enfoques. Las diferencias más notorias entre los dos casos son:

- La infraestructura física necesaria para implementar una solución orientada a la web es claramente mayor. No obstante, los servicios web hoy disponibles en forma directa o a través de terceros suelen facilitar el acceso físico a internet.

- La complejidad de las soluciones orientadas a la web suele ser mayor, ya que requiere de servicios adicionales de comunicación, balance de carga, gestión de seguridad, entre otros.
- Como consecuencia de lo anterior, los tiempos de desarrollo y los recursos necesarios para instalación, soporte, mantenimiento y evolución de las soluciones orientadas a la web suelen ser considerablemente mayores.
- Las soluciones autónomas en ocasiones son desarrolladas para un único sistema operativo, o pueden estar sujetas al uso de otras herramientas de *software* que deben instalarse en el equipo. Esto impone unas exigencias al equipo del usuario. Por el contrario, las soluciones orientadas a la web por lo general emplean del lado del usuario navegadores web convencionales ampliamente disponibles.
- La disponibilidad del laboratorio tiene dos escenarios diferentes: el laboratorio autónomo está tan disponible como el equipo donde está instalado, mientras que la disponibilidad del laboratorio orientado a la web está determinada por el acceso a un equipo conectado a internet.
- Los tiempos de respuesta de los laboratorios autónomo son menores, ya que en las soluciones orientadas a la web hay que sumar los tiempos de comunicación y de respuesta del servidor web a los tiempos propios de la simulación.

Figura 5.1 Orientación de laboratorios virtuales<sup>1</sup>

<sup>1</sup>Construida a partir de tres íconos distribuidos con licencia de Creative Commons disponibles en [link:2], [link:3] y [link:4].

Existe también un tipo de laboratorio que combina elementos de los dos enfoques. Se trata de laboratorios conectados a redes locales, que se operan desde equipos fijos. Una de las aplicaciones de este enfoque consiste en aislar al usuario de situaciones peligrosas. El esquema usual consta de un laboratorio real, altamente instrumentado para ser operado parcial o totalmente desde una sala de control local. Comparte con el enfoque orientado a la web muchas de las necesidades de infraestructura y gestión, pero no es accesible a través de internet.

### 5.1.2 Naturaleza de los modelos

En algunos laboratorios virtuales los experimentos se realizan sobre plantas físicas, mientras que en otros se realizan sobre modelos de *software*. Para distinguirlos, en este documento se empleará la convención de algunos autores, que denominan a los primeros laboratorios como *remotos* y *virtuales* a los segundos ([AN11, SOHA12, GR09]). Las siguientes son algunas de las diferencias más evidentes:

- Los laboratorios remotos requieren, además de la planta en sí, un conjunto de instrumentos y actuadores. Usualmente requieren también transmisión de video desde el laboratorio hacia el usuario final. Esta sofisticación los hace más costosos, tanto para su implementación como para su mantenimiento y gestión.
- Los modelos de *software* pueden ser accedidos simultáneamente por varios usuarios y, por tanto, se pueden efectuar varios experimentos diferentes a la vez; por el contrario, una planta física solo puede usarse a la vez en un único experimento.
- El usuario de los laboratorios remotos realiza experimentos que suelen estar prediseñados y no tiene oportunidad de modificarlos. Esto es así porque la planta física está preparada para un uso, y es difícil darle flexibilidad para reconfigurarla a través de la instrumentación que se opera remotamente. Por el contrario, los modelos de *software* pueden desarrollarse con una alta flexibilidad. Además, el usuario puede eventualmente acceder al código fuente para modificar el experimento completamente.
- Los modelos de *software* reflejan una fracción de la realidad: aquella que está incluida en el modelo matemático subyacente. Esto significa que muchos aspectos de la experimentación sobre plantas físicas no se verán reflejados en el laboratorio virtual.

- Las exigencias en el canal de comunicación son mayores para los laboratorios remotos. Un tiempo de retardo importante o muy variable puede hacer que la usabilidad del laboratorio se vea desmejorada, que se afecte la seguridad de la planta o, simplemente, que se arruine el experimento. Por el contrario, el impacto de estos retardos en los laboratorios virtuales se limita a la usabilidad.

### 5.1.3 Dominio de los experimentos

Las herramientas de *software* que sirven de soporte de simulación a algunos laboratorios virtuales están diseñadas para trabajar con modelos de un único dominio físico, ya sea eléctrico, mecánico, químico, biológico, etc. (p. ej., [RFT<sup>+</sup>09, SYLL02]), mientras que otras permiten la simulación de problemas multidominio (ver p. ej., [JWJH12, MUD08]). Aun cuando se utilicen herramientas multidominio, la interfaz desarrollada puede limitar su uso a un único dominio ([SOHA12, UMD05, SZaZ11]).

Como diferencias entre estos enfoques, pueden señalarse las siguientes:

- Las herramientas de simulación de dominio específico suelen tener APIs e interfaces gráficas que utilizan las convenciones y los usos propios de la disciplina en la que se aplican. Por esta razón, el desarrollo de aplicaciones tales como laboratorios virtuales puede ser más rápido y eficiente que si se desarrollaran estos con herramientas multidominio.
- Por la misma razón anterior, el desarrollo de los modelos que seán simulados suele ser más rápido al usar herramientas de dominio específico.
- El desarrollo de herramientas multidominio requiere un grado de generalización mayor que el de las herramientas de dominio específico. Por esta razón, los algoritmos de compilación y simulación pueden ser más lentos y menos eficaces.
- El tipo de experimentos que se pueden llevar a cabo con herramientas multidominio es más rico y por tanto más cercano a la realidad. La combinación de elementos de varios dominios brinda una perspectiva de aprendizaje que facilita la integración de conocimientos.

### 5.1.4 Interactividad

De acuerdo con las opciones de interacción que tiene el usuario con los parámetros de la simulación, pueden identificarse los siguientes casos ([Mar07]):

- Sin interactividad: los valores de los parámetros no pueden ser modificados por el usuario.
- De interactividad discontinua: los valores de algunos parámetros pueden modificarse antes de correr la simulación.
- De interactividad continua: los valores de algunos parámetros pueden modificarse antes de correr la simulación o mientras esta se ejecuta.

Las diferencias más evidentes según el grado de interactividad son las siguientes:

- La complejidad de la implementación se incrementa conforme se incrementa la interactividad. Es decir, los laboratorios sin interactividad son menos complejos que los de interactividad discontinua, y estos a su vez presentan una complejidad menor que los laboratorios de interactividad continua.
- La interactividad continua es especialmente útil en el entrenamiento del uso de equipos, pues allí se puede simular el efecto de cambiar los puntos de operación de sistemas en funcionamiento.

### 5.1.5 Función del laboratorio

Los laboratorios no virtuales suelen tener tres posibles funciones:

- Docencia: el laboratorio se concibe como un espacio de aprendizaje y se enfoca en estudiantes.
- Investigación: el laboratorio se concibe como un espacio para descubrir nuevo conocimiento y se enfoca en grupos de investigación.
- Servicios: el laboratorio se concibe como un espacio para brindar servicios tales como evaluar muestras o productos, enfocándose en el cliente.

No es extraño que un mismo laboratorio no virtual pueda atender más de una de esas funciones. Sin embargo, los experimentos de docencia, investigación y servicios tienen características y exigencias diferentes. En los laboratorios virtuales esto se traduce en modelos con requerimientos muy diferentes:

- Los requerimientos de seguridad para el acceso a los datos son bajos para docencia, medios para investigación y altos para servicios. En este último caso, la confidencialidad de la información puede ser un aspecto crítico.

- Mientras que los experimentos para docencia y servicios pueden ser relativamente repetitivos, la flexibilidad en la configuración de los experimentos puede ser un requisito funcional para investigación.
- Algunos laboratorios virtuales orientados a investigación permiten la interacción de grupos de instituciones de diversos países. Estos laboratorios deben brindar la posibilidad de conducir experimentos de forma colaborativa, donde los usuarios tengan acceso a los resultados de los experimentos realizados por otros usuarios ([SS11]).

### 5.1.6 Disponibilidad de los modelos

La posibilidad de acceder a los modelos matemáticos y al código fuente de su implementación en *software* no es igual para todos los laboratorios virtuales. El acceso a una buena documentación que explique las relaciones matemáticas de los modelos es necesario para poder hacer análisis profundos de los fenómenos simulados. Por otra parte, disponer del código fuente y de las herramientas necesarias para su interpretación o compilación es condición indispensable para modificar los modelos y adecuarlos a nuevas situaciones.

### 5.1.7 Integración con LMS y compatibilidad SCORM

Los Sistemas de Administración de Aprendizaje (LMS, por sus siglas en inglés) son aplicaciones web que permiten la publicación de cursos en internet, así como el acceso de los estudiantes a tales cursos. En este contexto, la iniciativa SCORM (Sharable Content Object Reference Model) ha generado una serie de especificaciones que permiten reutilizar los objetos de aprendizaje en entornos LMS ([Dua09]).

Los experimentos virtuales pueden asimilarse a objetos de aprendizaje. Por esta razón, es interesante analizar el nivel de integración con sistemas LMS que tienen los laboratorios virtuales, particularmente aquellos orientados a docencia. Estos niveles de integración pueden ser:

- Sin integración: el laboratorio no contempla la integración con LMS.
- Enlazables dentro de contenidos SCORM: los experimentos específicos pueden enlazarse de forma íntegra y ordenada dentro de objetos de aprendizaje que cumplen la especificación SCORM.
- Generadores de contenidos SCORM: el laboratorio genera contenidos exportables que cumplen la especificación SCORM (p. ej. [JPR08], [RGLC12]).

- Integrados en LMS: el laboratorio se integra con uno o más LMS para facilitar la administración de tareas, notas, foros, etc. (p. ej. [dPRRFH12]).

### 5.1.8 Licenciamiento y acceso

Los aspectos de licenciamiento y acceso deben analizarse desde varias perspectivas:

- El licenciamiento del laboratorio en sí mismo.
- El licenciamiento de las herramientas de *software* que emplea el laboratorio.
- El licenciamiento de los modelos disponibles en el laboratorio.
- Las políticas de acceso a las facilidades del laboratorio.

Las tres primeras perspectivas son importantes a efectos de replicar el laboratorio. Curiosamente, son muy pocos los desarrollos de laboratorios virtuales que se comparten con licenciamiento libre. La cuarta perspectiva incide en quiénes son los usuarios potenciales del laboratorio, así como en las posibilidades que hay de enlazarlo desde otras páginas (desde un LMS, por ejemplo).

## 5.2 CLASIFICACIÓN Y EJEMPLOS

En la literatura técnica se han reportado numerosos laboratorios virtuales con combinaciones diversas con respecto a las características enumeradas en la sección 5.1 (véanse, por ejemplo, [AN11, SYLL02, RFT<sup>+09</sup>, MUD08, GR09, CLL12, DS10, MUD06, PMM12, SOHA12, SZaZ11]). Con el propósito de presentar un panorama general, a continuación se propone una clasificación de una amplia variedad de herramientas de *software* y *hardware* que permiten realizar experimentos.

- a. Programas de simulación de dominio específico: herramientas de *software* autónomo que han sido desarrolladas para simulación de sistemas de una única disciplina. Ejemplos muy conocidos son OrCad-PSpice ([Cad16] en el dominio de circuitos eléctricos), SolidWorks ([Sol16] en el dominio de modelado mecánico) y AspenHysys ([Asp16] en el dominio de procesos químicos).

En estos programas de simulación generalmente el usuario debe construir el modelo mediante la conexión de componentes disponibles en una librería. El usuario también debe ajustar los parámetros de dichos modelos.

El *software* se encarga de generar las ecuaciones correspondientes (a partir de las leyes físicas que gobiernan el dominio específico), simularlas y presentar los resultados.

El conjunto de leyes físicas que cada uno de estos simuladores maneja se limita al dominio específico, y el usuario no tiene la posibilidad de ampliar dicho conjunto. Por esta razón, los fenómenos que se pueden simular están circunscritos al dominio. A manera de ejemplo, usando un simulador de circuitos eléctricos, el usuario podrá incorporar en su modelo una bombilla eléctrica, y analizar sus variables eléctricas en un determinado circuito, pero probablemente no pueda simular la radiación de luz generada por la bombilla.

- b. Programas de cálculo de propósito general: programas de cálculo autónomos que incluyen rutinas de solución numérica de ecuaciones diferenciales y de diferencia. Algunos de los más conocidos ejemplos son Matlab ([Mat16]), Scilab ([Ent16]), Gnu Octave ([pG16]) y Mathematica ([Wol16]).

Estos programas brindan ambientes de programación que permiten desarrollar aplicaciones de cálculo sofisticadas. Aprovechando esta posibilidad, se han desarrollado librerías con modelos de dominios específicos. En comparación con los simuladores de dominio específico, estas librerías suelen ser menos completas. Sin embargo, el potencial de cálculo y la libertad de programación hacen que esta alternativa sea muy versátil.

Para desarrollar simulaciones multidominio, el usuario puede utilizar las rutinas de cálculo de las librerías de dominios específicos, pero es él quien debe formular las relaciones matemáticas que vinculan los distintos dominios. Esta tarea puede ser complicada porque las librerías no necesariamente son compatibles entre sí, en términos de precisión de cálculo, tiempos de respuesta de los modelos y estructura de datos (véase [FDS<sup>+</sup>12]).

- c. Programas de simulación multidominio: programas autónomos que permiten la simulación de modelos multidominio a partir de la interconexión de elementos simples disponibles en librerías. El programa construye las relaciones matemáticas al interior de cada dominio y entre los diferentes dominios. En [FDS<sup>+</sup>12] se listan dieciséis de estos programas y se destacan sus principales características.

- d. Demostraciones en línea: aplicaciones orientadas a la web que muestran experimentos no interactivos; es decir, el usuario no puede modificar las condiciones del experimento. Pueden ser animaciones, videos o pequeñas aplicaciones de propósito específico. Por ejemplo, en [oCaLA10] se encuentra una colección de videos sobre experimentos del Department of Life Sciences Core Education Laboratories de la Universidad de California en Los Ángeles. Estos videos tiene por propósito familiarizar a los estudiantes con los experimentos que luego desarrollarán en el laboratorio físico, haciendo énfasis en las exigencias de seguridad.
- e. Laboratorios interactivos en la web: laboratorios virtuales por autonomía. Se trata de aplicaciones web en las que se simulan fenómenos a través de modelos matemáticos implementados en *software*. Usualmente los experimentos están predefinidos: el usuario no tiene la posibilidad de crear sus propios experimentos ni de modificar la estructura de los ya existentes. La interacción del usuario se limita a la modificación de parámetros del experimento.
- f. Laboratorios remotos: aplicaciones en la web que permiten manipular dispositivos físicos presentes en un laboratorio físico. Estos laboratorios se implementan generalmente con uno de dos propósitos: ampliar el horario de servicio de algún laboratorio físico o brindar acceso al laboratorio a estudiantes en modelos de educación a distancia.

A la anterior clasificación es necesario adicionarle numerosas combinaciones de herramientas tales, como:

- El uso de programas de simulación autónomos como motor de cálculo de laboratorios interactivos en línea (véanse [MUD08, SZaZ11]).
- La interfaz entre programas de simulación de dominio específico o multidominio con programas de cálculo de propósito general (véanse [NfN05, Mod16, Mod16]).

### 5.3 LABORATORIOS Y APRENDIZAJE DE LA INGENIERÍA

Para explorar el papel que cumplen los laboratorios en los procesos de aprendizaje de la ingeniería, en esta sección se abordan tres perspectivas diferentes y complementarias:

- La importancia de la experimentación por sí misma, como uno de los conocimientos propios de la ingeniería.

- La importancia del modelado y de la simulación por sí mismos, como herramientas básicas del ejercicio de la ingeniería.
- La importancia de la experimentación como estrategia de aprendizaje de algunos temas fundamentales en la ingeniería.

### 5.3.1 Importancia de la experimentación

La importancia de la experimentación en el aprendizaje de la ingeniería difícilmente puede ser sobreestimada. La iniciativa CDIO (véase [BBC<sup>+</sup>03]) ha identificado un conjunto amplio de habilidades generales que un ingeniero recién graduado debería tener. Este listado se ha construido bajo un concepto según el cual los ingenieros deben estar en capacidad de *concebir, diseñar, implementar y operar sistemas complejos* (véase [Cra01]). Si bien este listado no es taxativo (debe adecuarse y priorizarse según el contexto), sí es una buena referencia para señalar los principales requerimientos del aprendizaje de la ingeniería.

El listado de habilidades CDIO se organiza en cuatro tópicos:

1. Conocimiento técnico y razonamiento.
2. Aptitudes personales y profesionales y sus atributos.
3. Habilidades interpersonales: trabajo en equipo y comunicación.
4. Concebir, diseñar, implementar y operar sistemas en un contexto empresarial y social.

El tópico número 1 es específico para cada programa de ingeniería, mientras que los demás se consideran comunes a todos los programas de ingeniería. El número de habilidades listadas en los tópicos 2, 3 y 4 es cercano a 350. La tabla 5.1 muestra los subtópicos relacionados directamente con la experimentación, todos ellos incluidos en el tópico 2.

Este listado de habilidades destaca la importancia de la experimentación por sí misma. En otras palabras, *el ingeniero debe saber experimentar*. Para desarrollar estas habilidades, es preciso que las actividades de experimentación que realice un estudiante incluyan las siguientes tareas:

- Documentación y análisis de la información previa.
- Formulación de hipótesis.
- Diseño de experimentos.

- Ejecución de experimentos.
- Análisis de resultados.

Tabla 5.1 Subtópicos CDIO relacionados directamente con habilidades de experimentación

<p>2.2. Experimentación y descubrimiento de conocimientos.</p> <p>2.2.1. Formulación de hipótesis.</p> <ul style="list-style-type: none"> <li>2.2.1.1. Preguntas clave que deben examinarse.</li> <li>2.2.1.2. Hipótesis que serán probadas.</li> <li>2.2.1.3. Controles y grupos de control.</li> </ul> <p>2.2.2. Búsqueda de literatura impresa y electrónica.</p> <ul style="list-style-type: none"> <li>2.2.2.1. Estrategia de búsqueda de literatura.</li> <li>2.2.2.2. Búsqueda de información e identificación usando herramientas como (catálogos en línea, bases de datos y motores de búsqueda).</li> <li>2.2.2.3. Ordenamiento y clasificación de la información primaria.</li> <li>2.2.2.4. Calidad y confiabilidad de la información.</li> <li>2.2.2.5. Lo esencial y la innovación contenida en la información.</li> <li>2.2.2.6. Preguntas de investigación que están sin respuesta.</li> <li>2.2.2.7. Citas a referencia.</li> </ul> <p>2.2.3. Investigación experimental.</p> <ul style="list-style-type: none"> <li>2.2.3.1. El concepto y la estrategia del experimento.</li> <li>2.2.3.2. Las precauciones cuando los seres humanos son usados en experimentos.</li> <li>2.2.3.3. Construcción de experimentos.</li> <li>2.2.3.4. Protocolos de prueba y procedimientos experimentales.</li> <li>2.2.3.5. Medidas experimentales.</li> <li>2.2.3.6. Datos experimentales.</li> <li>2.2.3.7. Datos experimentales <i>vs.</i> modelos disponibles.</li> </ul> <p>2.2.4. Pruebas de hipótesis y defensa.</p> <ul style="list-style-type: none"> <li>2.2.4.1. Validez estadística de datos.</li> <li>2.2.4.2. Las limitaciones de los datos empleados.</li> <li>2.2.4.3. Conclusiones respaldadas en los datos, necesidades y valores.</li> <li>2.2.4.4. Posibles mejoras en el proceso de descubrimiento del conocimiento.</li> </ul>
--

### 5.3.2 Importancia del modelado y de la simulación

La tabla 5.2 muestra el listado de habilidades CDIO relacionadas con el modelado y la simulación. Estas dos actividades se circunscriben en dos tópicos diferentes: 1) el razonamiento y la resolución de problemas, y 2) el diseño. Se colige, entonces, que *un ingeniero debe saber construir modelos y simularlos* para poder resolver problemas y diseñar esas soluciones.

Tabla 5.2 Listado de habilidades CDIO relacionadas con el modelado y la simulación

- |  |
|--|
| <p>2.1. Razonamiento y resolución de problemas de Ingeniería.</p> <p>2.1.1. Modelado.</p> <ul style="list-style-type: none"> <li>2.1.1.1. Suposiciones para simplificar sistemas complejos y las circunstancias que los rodean.</li> <li>2.1.1.2. Modelos conceptuales y cualitativos.</li> <li>2.1.1.3. Modelos cuantitativos y simulaciones.</li> </ul> <p>2.1.2. Estimación y análisis cualitativo.</p> <ul style="list-style-type: none"> <li>2.1.2.1. Órdenes de magnitud, fronteras y tendencias.</li> <li>2.1.2.2. Pruebas de coherencia y errores (límites, unidades, etc.).</li> <li>2.1.2.3. Generalización de soluciones analíticas.</li> </ul> <p>2.1.3. Análisis de incertidumbre.</p> <ul style="list-style-type: none"> <li>2.1.3.1. Información incompleta y ambigua.</li> <li>2.1.3.2. Modelos probabilísticos y estadísticos de eventos y secuencias.</li> <li>2.1.3.3. Análisis de riesgos y costo-beneficio.</li> <li>2.1.3.4. Análisis de decisiones.</li> <li>2.1.3.5. Márgenes y reservas.</li> </ul> <p>2.1.4. Soluciones y recomendaciones.</p> <ul style="list-style-type: none"> <li>2.1.1.1. Solución de problemas.</li> <li>2.1.1.2. Resultados esenciales de las soluciones y los datos de los ensayos.</li> <li>2.1.1.3. Discrepancias en los resultados.</li> <li>2.1.1.4. Resumen de las recomendaciones.</li> <li>2.1.1.5. Posibles mejoras en el proceso de solución de problemas.</li> </ul> <p>4.4. Diseñar.</p> <p>4.4.4. Diseño en la disciplina.</p> <ul style="list-style-type: none"> <li>4.4.4.1. Técnicas apropiadas, herramientas y procesos.</li> <li>4.4.4.2. Herramienta de diseño, calibración y validación.</li> <li>4.4.4.3. Análisis cuantitativo de alternativas.</li> <li>4.4.4.4. Modelado, simulación y pruebas.</li> <li>4.4.4.5. Perfeccionamiento analítico del diseño.</li> </ul> <p>4.4.5. Diseño multidisciplinario.</p> <ul style="list-style-type: none"> <li>4.4.5.1. Interacciones entre las disciplinas.</li> <li>4.4.5.2. Distintos convenios e hipótesis.</li> <li>4.4.5.3. Diferencias en la madurez de modelos disciplinarios.</li> <li>4.4.5.4. Ambientes multidisciplinario para el diseño.</li> <li>4.4.5.5. Diseño multidisciplinario.</li> </ul> |
|--|

El modelo, como abstracción de la realidad, y la simulación, como predicción del comportamiento de esa realidad, son dos herramientas fundamentales de la ingeniería. En ocasiones, basta con saber emplear los modelos y herramientas de simulación disponibles. No obstante, en muchas aplicaciones de ingeniería es necesario construir nuevos modelos o adaptar las herramientas de simulación. Por esta razón, es importante que el ingeniero se aproxime a los simuladores no como un mero usuario, sino con un conocimiento profundo de los modelos matemáticos subyacentes y la forma en que esos modelos son tratados por los simuladores.

En este sentido, es interesante destacar uno de los hallazgos reportados en [GKC<sup>+</sup>09]. Se trata del reporte final producido por un panel de expertos del World Technology Evaluation Center (véase [Cen08]) en el que se evalúan de forma prospectiva las principales tendencias en investigación y desarrollo de simulaciones para las ciencias y la ingeniería. El hallazgo número 3 identifica la necesidad de una “nueva y moderna aproximación a la enseñanza y el entrenamiento de la próxima generación de investigadores en computación de alto desempeño específicamente, y en modelado y simulación en general, para el descubrimiento científico y la innovación en ingeniería”.

Las herramientas de simulación están teniendo una evolución importante. La integración de herramientas computacionales es una nueva tendencia cuyo propósito es lograr un uso coherente de diversas herramientas de *software* especializadas en diseño, optimización, manufactura, etc. Implícita está la integración de herramientas de simulación entre sí, y con herramientas de diseño, optimización, documentación, etc. La formación de ingenieros y científicos que puedan desenvolverse en este escenario de integración, es identificada en [CoICME08] como un asunto de seguridad y competitividad nacionales.

### 5.3.3 La experimentación como estrategia de aprendizaje

Son varios los estudios que reportan cómo la experimentación (real, virtual o remota) puede ayudar significativamente a la comprensión y el aprendizaje de ciertos temas. Algunos ejemplos de estos estudios son los siguientes:

- En [FE10] se realiza un estudio comparativo para evaluar el impacto del uso de laboratorios (físicos y virtuales) en el aprendizaje de circuitos eléctricos DC. El estudio evidencia un impacto considerable tanto en el aprendizaje del tema como en el desarrollo de habilidades de trabajo virtual.

- En [SOHA12] se realiza un estudio comparativo entre tres tecnologías de laboratorio: real, virtual y de realidad aumentada. El laboratorio virtual se implementa en Matlab y Simulink. Por su parte, el enfoque de realidad aumentada consiste en la incorporación de videos e imágenes. Ahora, el estudio se basa en una encuesta a estudiantes que desarrollan prácticas en los tres tipos de laboratorio. El tema sobre el que se desarrollan las prácticas son conceptos básicos de circuitos eléctricos. Las encuestas revelan que el enfoque de realidad aumentada tiene como fortaleza la facilidad de uso y una mejor comprensión de los conceptos.
- En [SDGL10] se reporta una mejoría en varios aspectos del proceso de aprendizaje sobre perforación de pozos mediante el uso de una laboratorio virtual:
  - Mayor motivación hacia los estudiantes.
  - Mejora en las habilidades de aprendizaje autónomo.
  - Mayor eficiencia en la enseñanza.
  - Mejor soporte al aprendizaje continuo a distancia.
- En [KERS<sup>+</sup>12] se presentan las experiencias de docentes y estudiantes en el uso de herramientas de simulación en la enseñanza de ingeniería química. Se emplean herramientas comerciales como DSpice y KSpice. Los temas abordados son:
  - Ingeniería química básica, un curso de pregrado en el Oslo University College.
  - Análisis de operabilidad y seguridad, un curso de posgrado en la Technical University of Denmark.
  - Control de procesos, un curso de pregrado y posgrado en la Aston University.

El artículo hace énfasis en la manera como se han integrado las actividades de simulación en cada uno de los cursos. El diseño de cada experimento está relacionado con los objetivos de cada curso y el nivel de conocimiento de los estudiantes.

- En [RFT<sup>+</sup>09] se presenta un extenso portal para la enseñanza de ingeniería química. La validación del portal se realiza mediante una encuesta a los estudiantes, quienes en su mayoría lo consideran útil o muy útil, y bien estructurado. La arquitectura empleada utiliza CGI para comunicar la interfaz web con el motor de cálculo, que emplea Matlab.

- En [MoL06] se enumeran algunos argumentos a favor del uso de juegos y simulaciones en la educación de adultos:
  - Ambientes libres de riesgo.
  - Experimentación.
  - Habilidades para la solución de problemas.
  - Evaluación.
  - Interacción social.
  - Cultura de jugadores.
- En [MoL06] se identifican cuatro aspectos que se deben abordar para aprovechar el potencial pedagógico de las simulaciones y los juegos:
  - El diseño instruccional.
  - La facilitación.
  - La evaluación.
  - Los estándares de *e-learning*.

En el mismo documento, se clasifica la literatura revisada sobre unos aspectos pedagógicos relacionados con las simulaciones:

- Sistemas de creencias de los docentes: ocurrencia en 17 %.
- Sistemas de creencias de los estudiantes: ocurrencia en 49 %.
- Obtención de competencias profesionales: ocurrencia en 15 %.
- Requerimientos de habilidades en el uso de nuevas tecnologías: ocurrencia en 7 %.
- En [AN11] se muestra una estrategia para incorporar tres tipos de experimentación en educación: directa, virtual y remota. Los tres tipos de acceso se utilizan en un mismo curso, con el propósito de explotar las principales ventajas de cada uno de ellos. Como resultado, se evidencian los beneficios de utilizar un laboratorio virtual antes de acceder al laboratorio directo.
- En [MN11] se hace una revisión de cincuenta y tres experiencias reportadas en diez años. En dicho trabajo se destaca la importancia de la presencia y la inmersión, como variables que requieren mayor investigación. Estos conceptos, no obstante, son bastante amplios; la presencia,

por ejemplo, se considera como una experiencia subjetiva, de tal forma que es coherente hablar de presencia a distancia, a través de medios de comunicación.

No es de extrañar, entonces, que los planes de estudio de las distintas ingenierías incluyan actividades de experimentación en los temas considerados como críticos o fundamentales para cada una de ellas. Se suelen seleccionar de forma estratégica los temas sobre los cuales se debe realizar una experimentación, para optimizar el uso de los recursos físicos disponibles en los laboratorios.

#### **5.4 CONSIDERACIONES EN EL DISEÑO DE LABORATORIOS VIRTUALES**

Varias de las experiencias reportadas en la literatura sobre el uso de ambientes virtuales de aprendizaje se acompañan de evaluaciones sobre su efectividad. Son menos las que exploran la incidencia de las características de diseño del sistema que impactan en su éxito. En [MS11] se hace una revisión detallada de treinta de esas evaluaciones, la mayoría de las cuales están basadas en encuestas realizadas a los usuarios. Para ello, clasifica las características de diseño en las que están relacionadas con el sistema y en aquellas relacionadas con la información (véase tabla 5.3).

Tabla 5.3 Características de diseño de un ambiente virtual de aprendizaje reportadas en [MS11]

Relacionadas con el sistema	Relacionadas con la información
Accesibilidad, comunicatividad, interactividad, diseño de la interfaz, comunidad de aprendizaje, navegación, personalización, disponibilidad, adaptabilidad, flexibilidad, funcionalidad, tiempo de respuesta, usabilidad, adaptación al usuario, herramientas para el usuario.	Características del curso, calidad del curso, formato, tipo de material, relevancia, terminología.

Por su parte, en [CLL12] se propone el uso del proceso analítico jerárquico difuso para identificar los requerimientos no funcionales de un laboratorio virtual. El asunto se aborda como un problema de decisión multicriterio para establecer la priorización de los criterios cualitativos que se involucran en el diseño del laboratorio como un servicio web. Estos criterios son tomados del estándar ISO/IEC 9126. Tras efectuar un análisis con 20 actores, se obtiene un listado ordenado de criterios (el primero es el prioritario):

- Funcionalidad
- Confiabilidad

- Usabilidad
- Eficiencia
- Mantenibilidad
- Portabilidad
- Seguridad

Por otra parte, no hay que olvidar que un laboratorio virtual no es más que una herramienta. En este sentido, su éxito depende del uso que de ella se haga. Esto es particularmente importante en su aplicación en la docencia, en donde el uso depende de la estrategia pedagógica seleccionada.

Los experimentos disponibles a través de laboratorios virtuales enfocados a la docencia pueden considerarse como Objetos Virtuales de Aprendizaje (OVA) ([Dua09]). Como tales, es deseable que sean reutilizables en diferentes contextos de aprendizaje. Este aspecto no suele ser explorado en los estudios de éxito de ambientes virtuales de aprendizaje, ya que estos se realizan en contextos limitados, tales como una cohorte de un único curso.

En [Dav08] se hace un compendio de un conjunto muy útil de recomendaciones y guías para el diseño de laboratorios y actividades de laboratorio para planes de estudio de ingeniería. El documento se estructura alrededor de cinco ejes de diseño de un curso: 1) propósitos y objetivos de aprendizaje, 2) diseño de aprendizaje, 3) entregas, 4) valoración, y 5) evaluación. A continuación se resumen algunas de las recomendaciones para los dos primeros aspectos:

a. Propósitos y objetivos de aprendizaje: el documento recoge las recomendaciones de [GGM97] según las cuales es preferible concentrarse en los objetivos de alto nivel:

- Desarrollo de habilidades experimentales, de diseño, de análisis y solución de problemas.
- Desarrollo de habilidades de análisis y registro de datos.
- Familiarizar a los estudiantes con equipo, técnicas y materiales.
- Desarrollo de habilidades prácticas.
- Desarrollo de habilidades interpersonales y de comunicación.
- Desarrollo de juicios técnicos y práctica profesional.
- Integración de teoría y práctica.

- Motivación a los estudiantes.
- b. Diseño de aprendizaje: el diseño de las actividades específicas a desarrollar está influenciado por el modelo de aprendizaje subyacente. El modelo de Aprendizaje por Experiencia (*Experiential Learning*) propuesto en [Kol84] es particularmente útil para entender el papel que los laboratorios deben cumplir en el proceso de aprendizaje.

De este modelo, destacamos los siguientes aspectos:

- De acuerdo con el modelo, el aprendizaje es un proceso iterativo (o una espiral) como el representado en la figura 5.2.

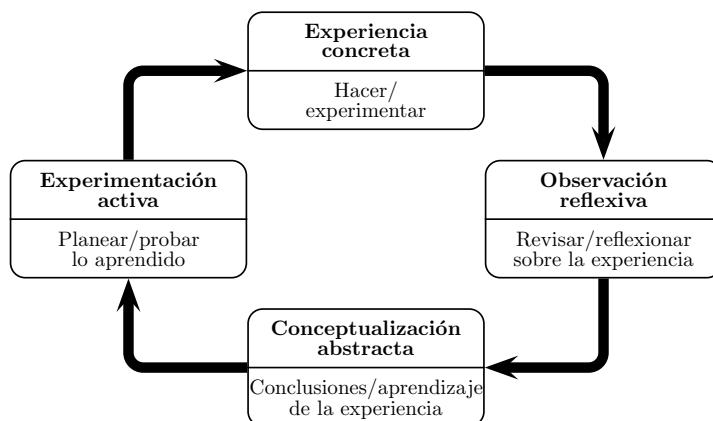


Figura 5.2 Modelo del aprendizaje de Kolb (Tomado de [Dav08])

- Cada estudiante tiene preferencias propias sobre cómo utilizar diferentes fases del ciclo, lo que establece diferentes *estilos de aprendizaje*. Los estilos de aprendizaje están influenciados por varios factores internos y externos, y evolucionan a lo largo de la vida de cada individuo (véanse [Kol81], [KK05], [JK09], [BK91]).

De acuerdo con este enfoque, es importante proveer a los estudiantes diferentes entradas factibles al ciclo de aprendizaje. Igualmente importante es distinguir el nivel de autonomía requerido para cada tipo de actividad de laboratorio, tal como se muestra en la tabla 5.4. Un diseño curricular coherente debería contemplar un nivel progresivo de autonomía a lo largo del plan de estudios<sup>2</sup>.

<sup>2</sup>En [AN09] se puede ver una aplicación de los principios del aprendizaje por experiencia a un curso control de procesos en Ingeniería Química que combina laboratorios remotos, virtuales y físicos.

Tabla 5.4 Niveles de autonomía y tipos de actividad de laboratorio (tomado de [Dav08])

Nivel de autonomía	Tipo de actividad	Condiciones			
		Objetivos	Material	Método	Respuesta
0	Demostración	Dado	Dado	Dado	Dado
1	Ejercicio	Dado	Dado	Dado	Abierto
2	Investigación estructurada	Dado	Dado o parcialmente dado	Abierto o parcialmente abierto	Abierto
3	Investigación abierta	Dado	Abierto	Abierto	Abierto
4	Proyecto	Abierto	Abierto	Abierto	Abierto



# 6

## UNVIRTUALLAB. ARQUITECTURA



En este capítulo se presenta en detalle la estructura interna de UNVirtualLab. El capítulo inicia con la presentación de las especificaciones de la herramienta (sección 6.1); a continuación se presenta el modelo Entidad-Relación de la base de datos subyacente (sección 6.2), así como los componentes de la implementación (sección 6.3), la interfaz gráfica (sección 6.4), la estrategia de implementación (sección 6.5) y las opciones de configuración (sección 6.6). En los apéndices B, C y D se ha incluido información complementaria.

## 6.1 ESPECIFICACIONES

### 6.1.1 Características

UNVirtualLab ha sido concebido como un ambiente de simulación de sistemas dinámicos enfocado al aprendizaje. Dicho ambiente es una herramienta de propósito general, en el sentido de que puede alojar plantas de experimentación de diferentes disciplinas y niveles de formación. Las plantas alojadas en UNVirtualLab se desarrollan bajo modelos que facilitan la representación de fenómenos en múltiples dominios.

A continuación se presentan las características de UNVirtualLab siguiendo la enumeración dada en la sección 5.1:

- a. Orientación a la web: UNVirtualLab es una herramienta web cuyos modelos pueden usarse en herramientas *stand-alone*. De esta forma, mientras la orientación hacia la web permite el acceso remoto a las plantas de experimentación disponibles, el uso de las herramientas stand-alone facilita el desarrollo de experimentos más complejos por parte del usuario.
- b. Naturaleza de los modelos: UNVirtualLab es una herramienta de simulación. Los experimentos son ejecuciones de programas construidos sobre modelos matemáticos. Se ha decidido que sea así y no un laboratorio remoto, para que sea muy versátil y de esa forma pueda alojar experimentos de múltiples disciplinas y niveles de formación.
- c. Dominio de los experimentos: UNVirtualLab puede alojar plantas de experimentación multidominio. Los modelos que subyacen a los experimentos se desarrollan en un lenguaje que facilita la representación multidomínio. Este aspecto permite desarrollar experimentos cuya complejidad lleva

al usuario a enfrentarse con la necesidad de abordar de forma interdisciplinaria los fenómenos modelados.

- d. Interactividad: UNVirtualLab permite que el usuario tenga una interactividad discontinua con el experimento. La interactividad es indispensable para que el usuario modifique los parámetros de los experimentos y así amplie su experiencia de aprendizaje. Cabe anotarse que la interactividad continua no se ha incorporado por no considerarse un aspecto crítico en la experiencia de aprendizaje.
- e. Apoyo al aprendizaje: UNVirtualLab tiene un enfoque hacia el aprendizaje. Puede considerarse como una herramienta de apoyo a la docencia, aunque también puede utilizarse en modelos de autoaprendizaje. El aprendizaje por experiencia es el modelo subyacente ([Kol84]). UNVirtualLab provee diferentes elementos para facilitar que el estudiante acceda al ciclo de aprendizaje en diferentes puntos, de acuerdo con su estilo de aprendizaje ([KK05]). La figura 6.1 muestra los diferentes puntos de entrada al ciclo, así como los elementos que lo facilitan. Desde otra perspectiva, es posible emplear UNVirtualLab para desarrollar actividades de aprendizaje de todos los tipos propuestos en [Dav08] y recogidos en la tabla 5.4.
- f. Disponibilidad de los modelos: UNVirtualLab permite que el usuario tenga acceso a una amplia documentación de los modelos. También permite que el usuario descargue el código fuente de los modelos y de los experimentos, para que los adapte y corra en herramientas *stand-alone*.
- g. Integración con LMS y compatibilidad SCORM: los experimentos disponibles en UNVirtualLab pueden enlazarse directamente desde ambientes LMS, o ser empaquetados como enlaces en archivos SCORM.
- h. Licenciamiento y acceso: UNVirtualLab se distribuye bajo licenciamiento libre (véase apéndice A). Los experimentos y modelos se distribuyen también bajo esa misma licencia, y son compilables en herramientas de *software* libre. El acceso a los experimentos es abierto al público en general.

### 6.1.2 Concepto de diseño

UNVirtualLab combina 1) el potencial de modelamiento del lenguaje Modelica, 2) la capacidad de simulación de OpenModelica, y 3) la facilidad de acceso que brinda la web.

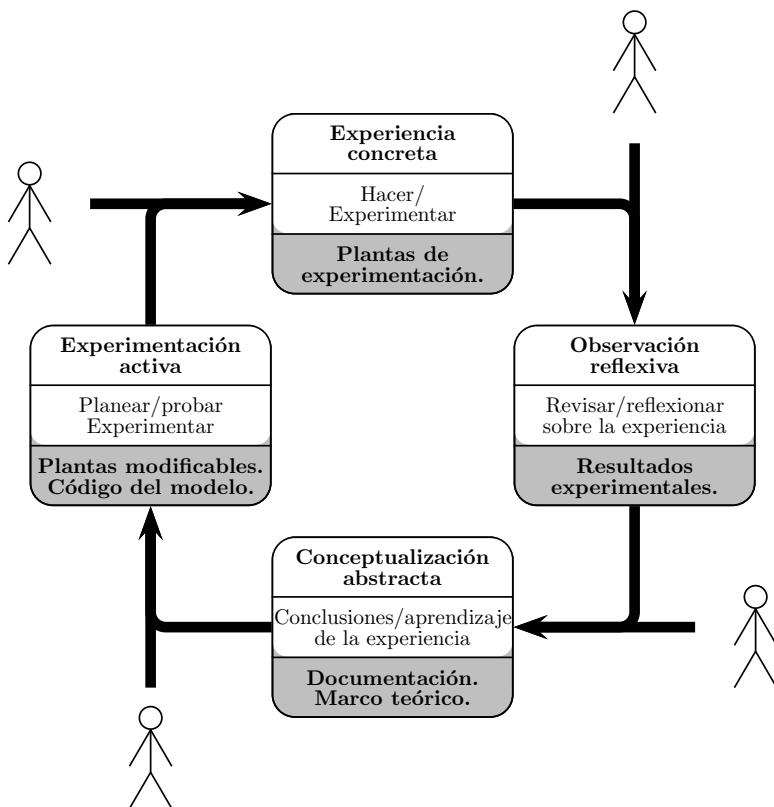


Figura 6.1 UNVirtualLab como herramienta de aprendizaje bajo el modelo de Kolb [Dav08]. En sombreado los elementos presentes en UNVirtualLab que brindan acceso al ciclo de aprendizaje

Para explicar cómo se logra esta combinación, conviene recordar cómo interactúa OpenModelica con los archivos para producir una simulación (véase la figura 4.4): OpenModelica toma un archivo de texto escrito en lenguaje Modelica y produce varios archivos de los cuales destacamos tres:

- Un archivo ejecutable.
- Un archivo de texto con parámetros de entrada para el ejecutable.
- Un archivo con los resultados de la simulación, que es generado por el ejecutable cuando este se corre.

Este esquema permite aprovechar el mismo ejecutable para correr simulaciones en condiciones diferentes. Para ello es necesario modificar el archivo

con los parámetros de entrada. No obstante, ninguna de las herramientas disponibles en la *suite* OpenModelica están diseñadas para editar el archivo de entrada; todas ellas han sido concebidas para interactuar con el modelo en sí mismo.

UNVirtualLab brinda una interfaz web para modificar los archivos de entrada de simulaciones compiladas previamente con OpenModelica. La figura 6.2 ilustra el proceso: el usuario interactúa con la interfaz web para ajustar los parámetros del experimento y para ver los resultados de este; cuando se lanza una simulación a través de la interfaz, UNVirtualLab crea un archivo de entrada temporal con los ajustes del usuario, corre la simulación con ese archivo y lee los resultados para desplegarlos en diferentes formatos<sup>1</sup>.

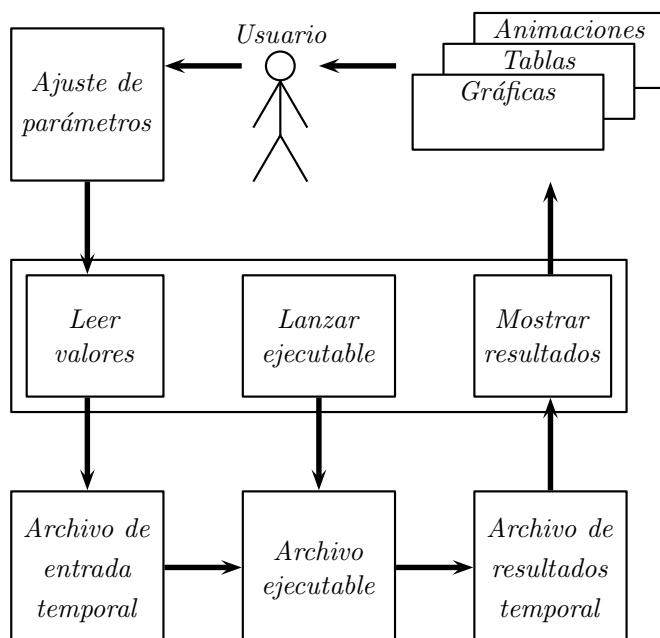


Figura 6.2 Archivos en una simulación de UNVirtualLab

<sup>1</sup>En [TTHFR11] se muestra otro enfoque diferente para dotar a OpenModelica de la facilidad de acceso que brinda la web: la interfaz permite modificar los archivos Modelica, de tal manera que al lanzar una simulación el sistema debe cada vez compilar el modelo para generar un nuevo ejecutable antes de correr la simulación. Para los autores de [TTHFR11], el propósito del ambiente de simulación es principalmente el aprendizaje del lenguaje Modelica, por lo que la edición en línea de los archivos del modelo es fundamental. No obstante, al introducir esta funcionalidad, los tiempos de simulación se incrementan considerablemente.

La información asociada a UNVirtualLab se aloja en una base de datos relacional. El diagrama UML de actividades de la figura 6.3 muestra con más detalle el proceso de simulación en UNVirtualLab enumerado a continuación:

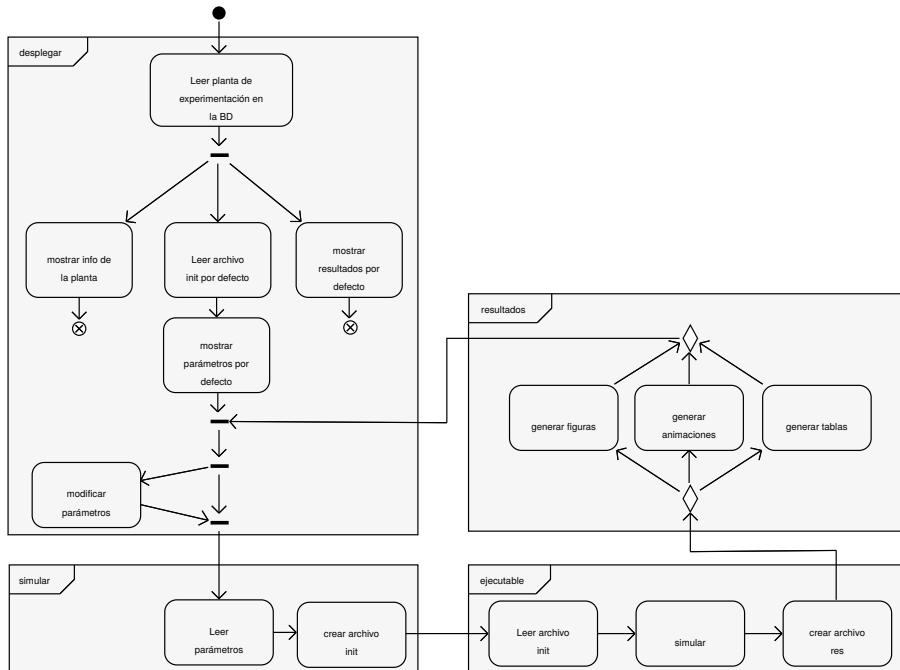


Figura 6.3 Diagrama UML de actividades de una simulación en UNVirtualLab

UNVL.1. Al seleccionar una planta de experimentación, el sistema busca la información correspondiente en la base de datos y despliega tres tipos de contenidos:

- La documentación de la planta
- El valor por defecto de los parámetros editables
- Los resultados de la simulación con los valores por defecto

UNVL.2. El usuario puede en este punto modificar el valor de los parámetros antes de lanzar la simulación.

UNVL.3. Una vez se lanza la simulación, el sistema lee los valores actuales de los parámetros y construye un archivo `init` que será leído por el

ejecutable. El sistema corre el ejecutable que genera el archivo res que contiene los resultados de la simulación.

UNVL.4. El sistema despliega los resultados en formato de gráficas, tablas y animaciones, siguiendo la información consignada en la base de datos.

### 6.1.3 Especificaciones funcionales

Para enumerar los requerimientos funcionales (RF), se consideran dos tipos de usuario: el experimentador y el administrador (véase sección 6.1.4). Las funciones que UNVirtualLab debe realizar son las siguientes:

- RF.1. Mostrar al experimentador la colección de plantas de experimentación disponibles.
- RF.2. Permitir que el experimentador seleccione la planta de experimentación sobre la que desea trabajar.
- RF.3. Mostrar al experimentador documentación útil para entender la planta y los modelos subyacentes.
- RF.4. Mostrar al experimentador los valores por defecto de los parámetros de la planta.
- RF.5. Permitir que el experimentador modifique los valores de los parámetros de la planta.
- RF.6. Permitir que el experimentador recupere los valores por defecto de los parámetros de la planta.
- RF.7. Ejecutar el experimento (es decir, simular el comportamiento de la planta de experimentación) con los valores de los parámetros que seleccione el experimentador. Este requerimiento implica los siguientes subrequerimientos:
  - RF 7.1. Crear los archivos temporales de entrada para cada simulación lanzada por el experimentador.
  - RF 7.2. Leer los archivos temporales generados por los ejecutables en cada simulación lanzada por el experimentador.
  - RF 7.3. Presentar los resultados de los experimentos usando varios formatos: gráficas, tablas y animaciones.

- RF.7.4. Borrar los archivos temporales una vez se presenten los resultados.
- RF.8. Permitir que el experimentador descargue los modelos subyacentes al experimento.
- RF.9. Permitir que el experimentador descargue los valores numéricos de las variables simuladas en un experimento.
- RF.10. Mostrar al experimentador la información sobre la autoría de los modelos y experimentos.
- RF.11. Controlar el acceso del administrador mediante una contraseña.
- RF.12. Permitir que el administrador seleccione el idioma en que se presentará la interfaz.
- RF.13. Permitir que el administrador modifique la apariencia de la interfaz a través de archivos CSS.
- RF.14. Mostrar al administrador la colección de plantas de experimentación disponibles.
- RF.15. Permitir que el administrador adicione y elimine plantas de experimentación.
- RF.16. Permitir que el administrador cree secciones de plantas de forma jerárquica.
- RF.17. Permitir que el administrador ubique las plantas en la sección que desee.
- RF.18. Permitir que el administrador habilite o inhabilite la presentación de cada planta de experimentación a los experimentadores.
- RF.19. Permitir que el administrador edite la información asociada a cada planta.
- RF.20. Permitir que el administrador adicione, edite y elimine experimentos sugeridos a los experimentadores para cada experimento.
- RF.21. Permitir que el administrador habilite e inhabilite los experimentos sugeridos a los experimentadores para cada planta de experimentación.
- RF.22. Permitir que el administrador seleccione los parámetros que serán modificables por el experimentador, para cada planta de experimentación.

- RF.23. Permitir que el administrador adicione, edite y elimine grupos de controles en cada planta de experimentación; estos contendrán los parámetros editables por los experimentadores.
- RF.24. Permitir que el administrador habilite e inhabilite grupos de controles en cada planta de experimentación.
- RF.25. Permitir que el administrador adicione y elimine los parámetros editables por los experimentadores en cada grupo de control y para cada planta de experimentación.
- RF.26. Permitir que el administrador habilite e inhabilite parámetros en cada grupo de control.
- RF.27. Permitir que el administrador edite la información asociada a cada parámetro editable por el experimentador, como: nombre, valor por defecto, valores mínimo y máximo permitidos, entre otros datos.
- RF.28. Permitir que el administrador adicione y elimine gráficas para desplegar los resultados de cada planta de experimentación.
- RF.29. Permitir que el administrador habilite e inhabilite gráficas en cada planta de experimentación.
- RF.30. Permitir que el administrador edite la información asociada a cada gráfica de cada planta de experimentación, tal como: título, escalas horizontal y vertical, número de divisiones verticales y horizontales.
- RF.31. Permitir que el administrador adicione y elimine curvas en cada gráfica para desplegar los resultados de cada experimento.
- RF.32. Permitir que el administrador habilite e inhabilite curvas en cada gráfica.
- RF.33. Permitir que el administrador edite información asociada a cada curva de cada gráfica de cada experimento, tal como: color de la curva y nombre de la variable graficada.
- RF.34. Permitir que el administrador adicione y elimine animaciones en 2D para presentar los resultados de las simulaciones.
- RF.35. Permitir que el administrador edite información asociada a cada animación de cada experimento, tal como: título, duración de la animación y tiempo de actualización.

- RF.36. Permitir que el administrador asocie variables de simulación a las propiedades geométricas de la animación.
- RF.37. Permitir que el administrador adicione, elimine y edite información sobre los autores de los modelos y experimentos.
- RF.38. Permitir que el administrador asocie autores de modelos y experimentos a cada planta de experimentación.
- RF.39. Permitir que el administrador cargue la documentación asociada a cada planta de experimentación para que esté a disposición de los experimentadores.
- RF.40. Permitir que el administrador cargue el código fuente asociado a cada planta de experimentación para que esté a disposición de los experimentadores.
- RF.41. Permitir que el administrador cargue el archivo ejecutable, junto con los archivos por defecto de entrada y salida de la simulación.
- RF.42. Crear los archivos de imágenes de gráficas y animaciones por defecto de cada experimento para que estén a disposición de los experimentadores.

#### **6.1.4 Casos de uso**

A partir de las especificaciones funcionales se han derivado los casos de uso que se muestran en las figuras 6.4, 6.5 y 6.6. Los tipos de usuario identificados son<sup>2</sup>:

- a. Experimentador: que utiliza UNVirtualLab para realizar experimentos virtuales. La figura 6.4 muestra los casos de uso asociados a este usuario y que corresponden a los requerimientos funcionales 1 al 10.
- b. Administrador: que gestiona el sistema y la información relacionada con los experimentos virtuales. La figura 6.5 resume los casos de uso asociados al administrador que corresponden a los requerimientos funcionales 11 al 42. Se destacan las funciones de inicio y cierre de sesión, y dos bloques

---

<sup>2</sup>Existe un tercer tipo de usuario, el modelador, que es quien crea los modelos y experimentos. No obstante, este usuario no interactúa directamente con UNVirtualLab, sino que emplea OpenModelica para el desarrollo de los modelos y experimentos. Una vez estos modelos están listos, se incorporan a UNVirtualLab a través de las funciones del administrador. Por esta razón, no es necesario considerar al usuario modelador en la explicación de la arquitectura del sistema.

de casos: uno asociado a las secciones de las plantas, y otro asociado a las plantas de experimentación en sí. La figura 6.6 amplía los casos de uso asociados a la edición de cada planta de experimentación.

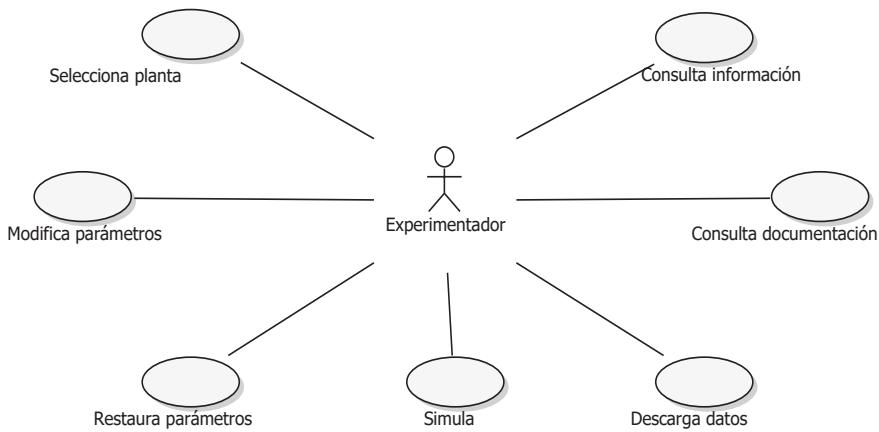


Figura 6.4 Diagrama UML de casos de uso para el experimentador

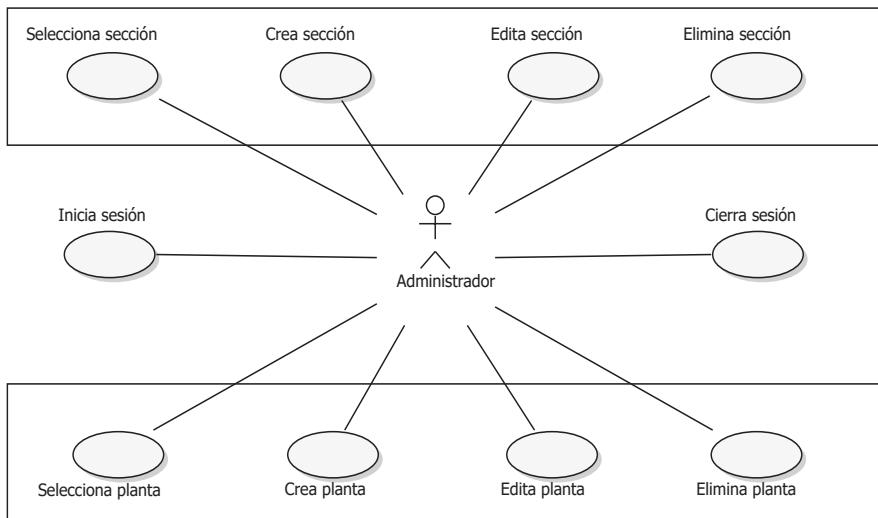


Figura 6.5 Diagrama UML de casos de uso para el administrador

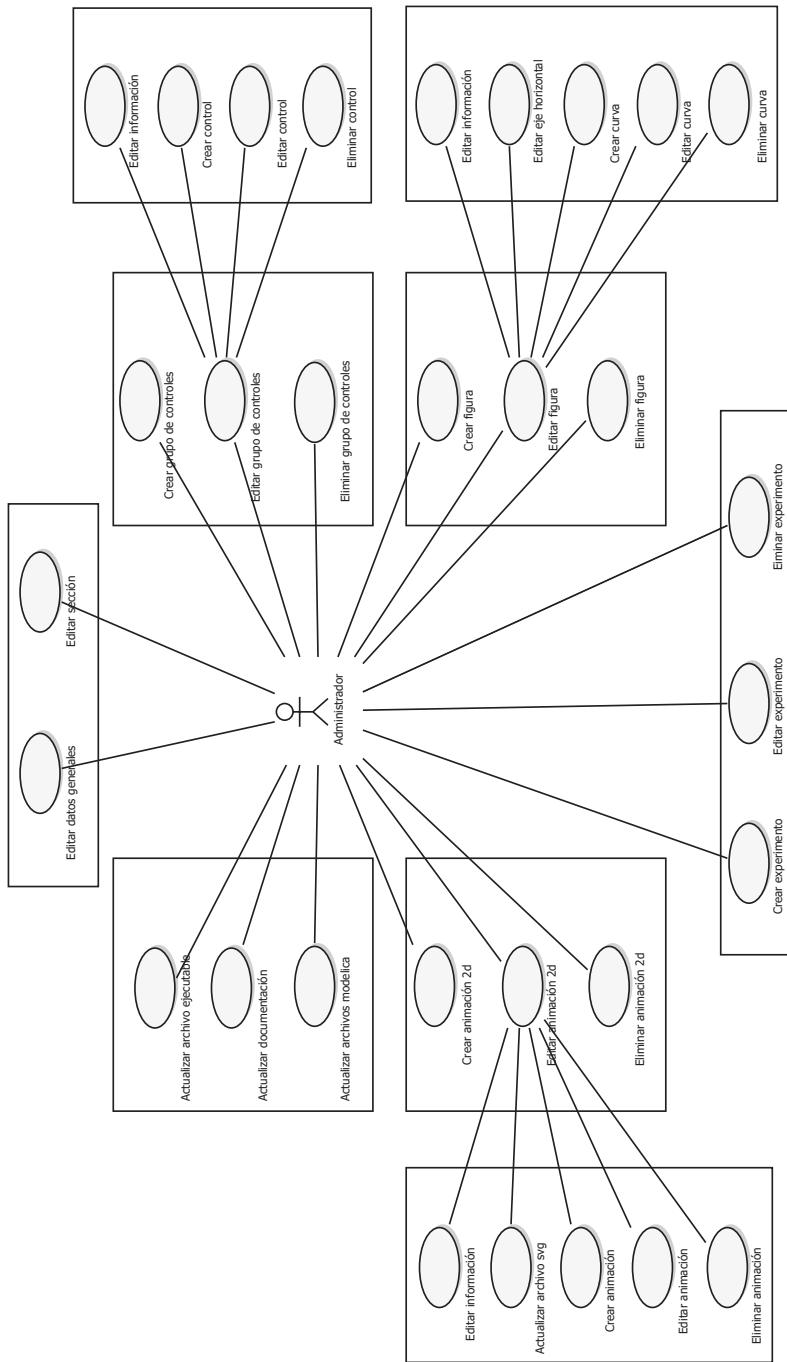


Figura 6.6 Diagrama UML de casos de uso para el administrador. Edición de experimentos

## 6.2 MODELO ENTIDAD-RELACIÓN

Para gestionar la información asociada a UNVirtualLab se ha diseñado una base de datos relacional sobre el motor mysql (véase [Cor22]). El diagrama Entidad-Relación de la base de datos se muestra en la figura 6.7, mientras que la estructura de cada una de las tablas se reseña en el apéndice B.

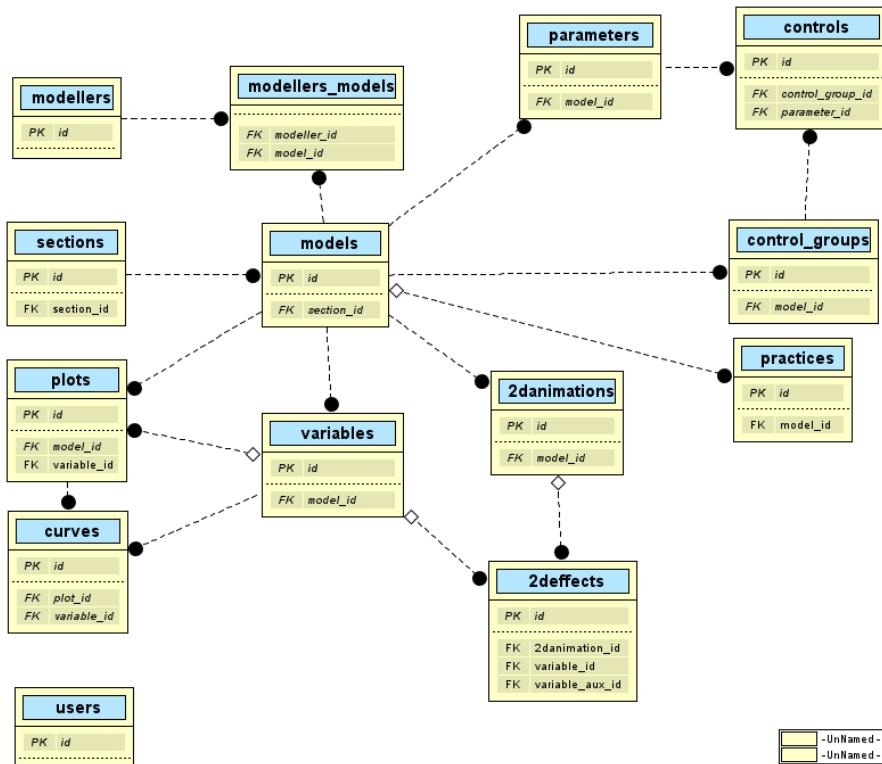


Figura 6.7 Diagrama Entidad-Relación de la base de datos

A continuación se explica el tipo de información que se aloja en cada una de las tablas:

- sections: cada registro de esta tabla describe una sección de experimentos. Las secciones pueden anidarse para constituir una estructura de árbol. Esto se logra con el campo section\_id, que funciona como apuntador al registro padre de la sección.

- **models:** cada registro de esta tabla describe el modelo ejecutable de un experimento, es decir, a una planta de experimentación. El campo `section_id` apunta a la sección a la que pertenece la planta.
- **modellers:** cada registro de esta tabla contiene información sobre un autor o coautor de un modelo o experimento.
- **modellers\_models:** esta tabla sirve para construir la relación  $\infty \leftrightarrow \infty$  entre los modelos (`models`) y sus autores (`modellers`).
- **practices:** cada registro de esta tabla describe un experimento sugerido para ser realizado con una determinada planta de experimentación. El campo `model_id` apunta al modelo de la planta con el que se realiza el experimento.
- **variables:** cada registro de esta tabla contiene información de una de las variables que se simulan al ejecutar un modelo específico. El campo `model_id` apunta al modelo al que pertenece la variable.
- **parameters:** cada registro de esta tabla contiene información de uno de los parámetros disponibles para ser modificados en los archivos de entrada de un modelo específico. El campo `model_id` apunta al modelo al que pertenece el parámetro.
- **control\_groups:** los parámetros que el experimentador puede modificar se pueden agrupar en grupos de controles. Cada registro de esta tabla describe uno de tales grupos asociados a un modelo específico. El campo `model_id` apunta al modelo al que pertenece el grupo de controles.
- **controls:** cada registro en esta tabla describe un parámetro que puede ser controlado (modificado) por el experimentador. El campo `control_group_id` identifica el grupo al que pertenece el control, mientras que el campo `parameter_id` identifica de qué parámetro se trata.
- **plots:** cada registro de esta tabla identifica una gráfica para presentar los resultados de la simulación de un determinado experimento. El campo `model_id` apunta al modelo cuyo resultado será graficado. El campo `variable_id` identifica la variable que estará en el eje horizontal (la abscisa).
- **curves:** en una misma gráfica se pueden dibujar varias curvas (con la misma abscisa). Cada registro de esta tabla describe una curva de una

gráfica determinada. El campo `plot_id` apunta a la gráfica correspondiente, mientras que el campo `variable_id` indica cuál es la variable que será graficada (la ordenada).

- `2danimations`: cada registro de esta tabla contiene información sobre una animación que se emplea para mostrar los resultados de la simulación de un modelo determinado. El campo `model_id` apunta al modelo cuyo resultado será animado.
- `2deffects`: las animaciones son alteraciones de propiedades geométricas o gráficas de un archivo gráfico. Dichas alteraciones se han denominado efectos de animación. Cada registro de esta tabla contiene un efecto de una animación determinada. El campo `2danimation_id` apunta a la animación a la que pertenece el efecto. El campo `variable_id` apunta a la variable que controla el efecto. Si el efecto requiere información de alguna variable auxiliar, el campo `variable_aux_id` se emplea para apuntar a ella.
- `users`: cada campo de esta tabla se utiliza para guardar información de los usuarios que tiene acceso con el rol de administrador.

### 6.3 COMPONENTES

UNVirtualLab es una aplicación web que emplea varios componentes para su operación. El conjunto de componentes requeridos no es el mismo para todos los usuarios y configuraciones. Pueden distinguirse los siguientes casos:

- a. **Experimentador**: utiliza los experimentos disponibles en alguna implementación de UNVirtualLab en la web; es decir, es fundamentalmente un cliente de UNVirtualLab. La figura 6.8 muestra los componentes requeridos, la cual básicamente consta de un navegador web con un intérprete de archivos SVG.
- b. **Experimentador avanzado**: además de utilizar como cliente los experimentos de alguna implementación web de UNVirtualLab, descarga los modelos para editarlos y compilarlos localmente. Como muestra la figura 6.9, este usuario requiere, además del navegador, una instalación local de OpenModelica.
- c. **Servidor mínimo**: una implementación mínima del servidor de UNVirtualLab requiere los módulos que se muestran en la figura 6.10. Se trata

de una aplicación web típica basada en mysql y php en la que el conjunto de *scripts* desarrollados se han agrupado en el módulo denominado UNVirtualLab. En esta implementación los archivos ejecutables de los modelos son compilados externamente y cargados al servidor.

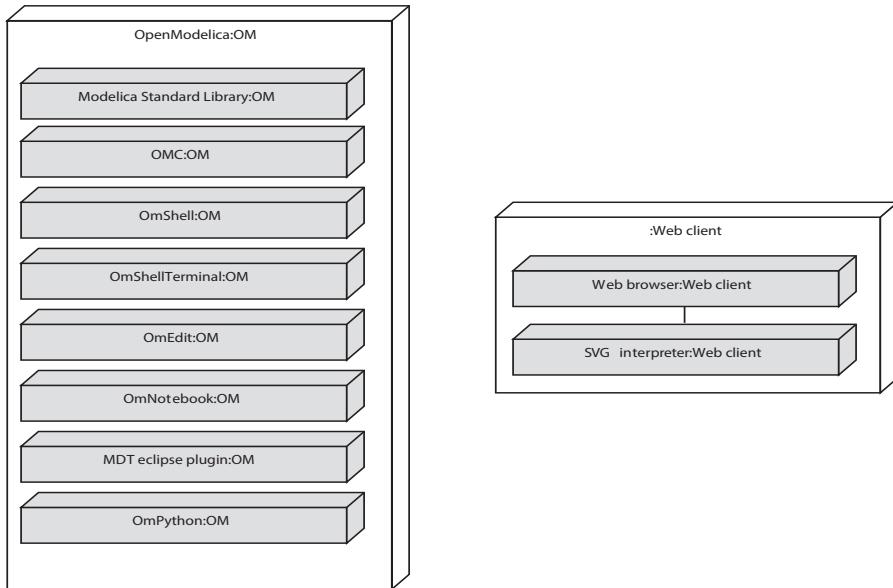


Figura 6.8 Diagrama UML de implementación. Módulos para el experimentador sin edición de modelos

- d. Servidor con compilación: en esta implementación el servidor se encarga de compilar los modelos desarrollados en lenguaje Modelica. Como se observa en la figura 6.11, además de los módulos de una implementación web típica, es necesario instalar un subconjunto de las aplicaciones de la *suite* OpenModelica. Esta implementación es necesaria cuando el sistema operativo del servidor es diferente del que existe en la plataforma externa donde se desarrollan los modelos.
- e. Servidor para el modelador: la implementación sugerida para el desarrollador de modelos y experimentos consta de una instalación completa de la *suite* OpenModelica, acompañada de una instalación local del un servidor UNVirtualLab. El conjunto completo se muestra en la figura 6.12.

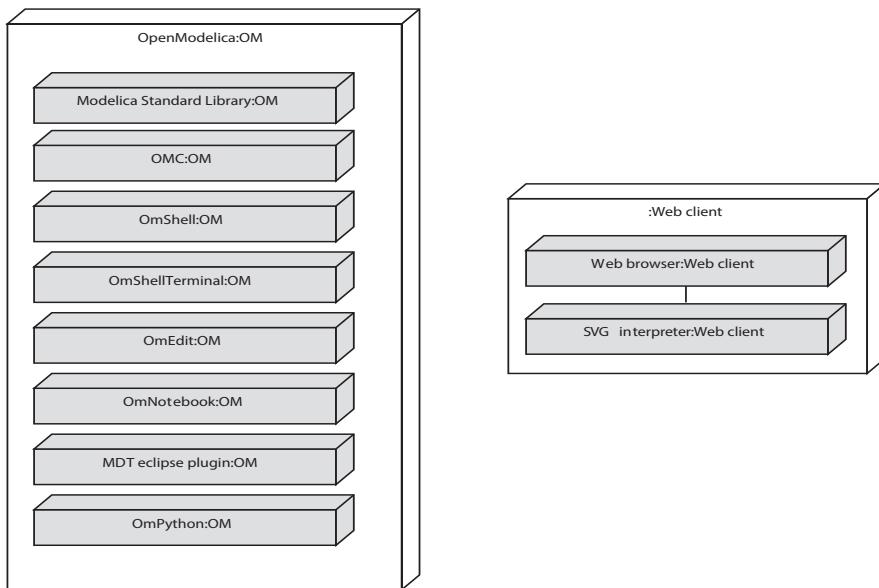


Figura 6.9 Diagrama UML de implementación. Módulos para el experimentador avanzado con edición de modelos

La tabla 6.1 muestra las características de los módulos empleados en el desarrollo de la versión 2.0 de UNVirtualLab<sup>3</sup>.

Tabla 6.1 Características de los módulos de la versión 2.0 de UNVirtualLab

Módulo	Nombre	Versión
Sistema operativo	Linux ubuntu	16.04 LTS sobre procesador AMD Athlon(tm) II P340 Dual-Core Processor x 2 de 64 bits
Servidor web	apache	2.4.18
Servidor mysql	mysql-server	5.7.21
Intérprete php	php7	7.0.28
<i>Suite</i> OpenModelica	OpenModelica	1.13.0.
Clientes web	firefox	59.0.2
	google-chrome-stable	51.0.2704
Scripts UNVirtualLab	UNVirtualLab	2.0

<sup>3</sup>Para obtener información sobre la versión 1.0, véase el apéndice E.

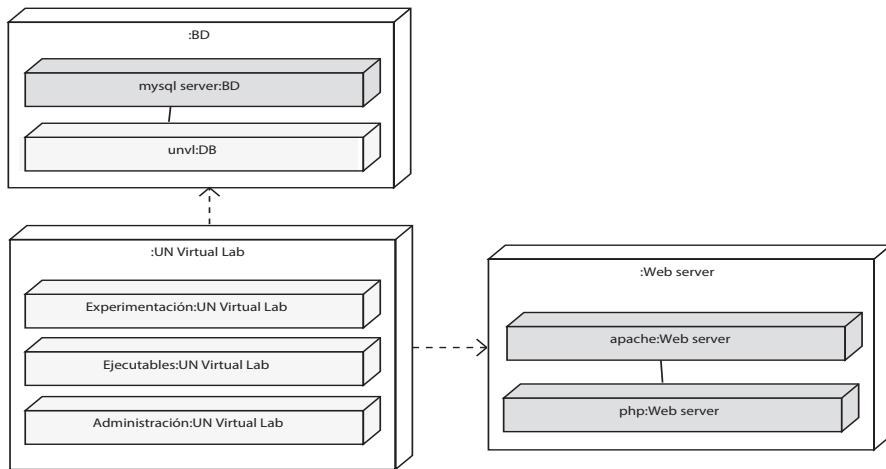


Figura 6.10 Diagrama UML de implementación. Módulos del servidor mínimo de UNVirtualLab

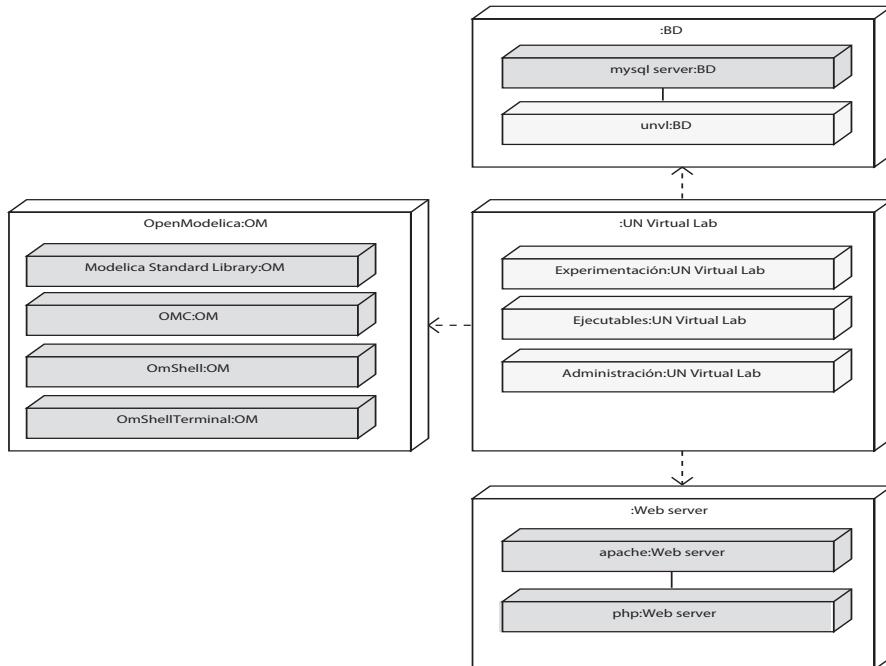


Figura 6.11 Diagrama UML de implementación. Módulos del servidor de UNVirtualLab con compilación de modelos

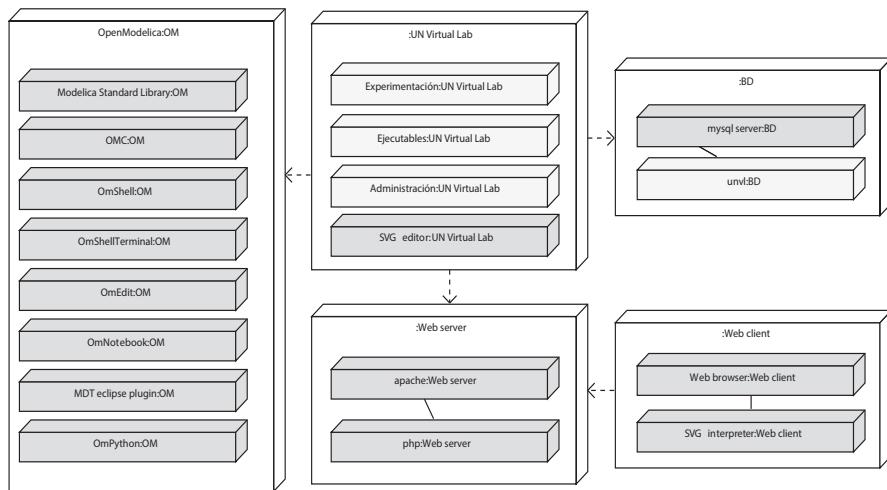


Figura 6.12 Diagrama UML de implementación. Módulos para el desarrollador de modelos y experimentos

## 6.4 INTERFAZ

### 6.4.1 Estructura de páginas

La interfaz web de UNVirtualLab se estructura como un conjunto de bloques anidados. Cada bloque es una región rectangular con una funcionalidad específica. La estructura de la interfaz se define en un archivo XML guardado en la carpeta `page_structure`. Este archivo consta de etiquetas anidadas `block` que contienen las siguientes propiedades:

- `id`: nombre que identifica el bloque.
- `type`: tipo de elemento HTML que contendrá el bloque.
- `class`: nombre de la clase CSS que controla la apariencia del bloque.
- `action`: nombre de la clase PHP encargada de desplegar el contenido interno del bloque.

A manera de ejemplo, considérese la figura 6.13 en la que se muestra la apariencia de la página *about* (subfigura 6.13(a)), su estructura de bloques (subfigura 6.13(b)) y el código del archivo XML (subfigura 6.13(c)) en el que

se define esa estructura<sup>4</sup>. En esta página se despliega la información de referencia de un modelo específico. La página consta de dos bloques, uno anidado dentro del otro:

- Un bloque del tipo body cuya apariencia se controla con la clase CSS `unvl`; el bloque no tiene contenidos específicos y, por tanto, no hay ninguna clase PHP asociada.
- Un bloque del tipo table cuya apariencia se controla con la clase CSS `about`; el contenido del bloque lo despliega la clase PHP `abouttable`.

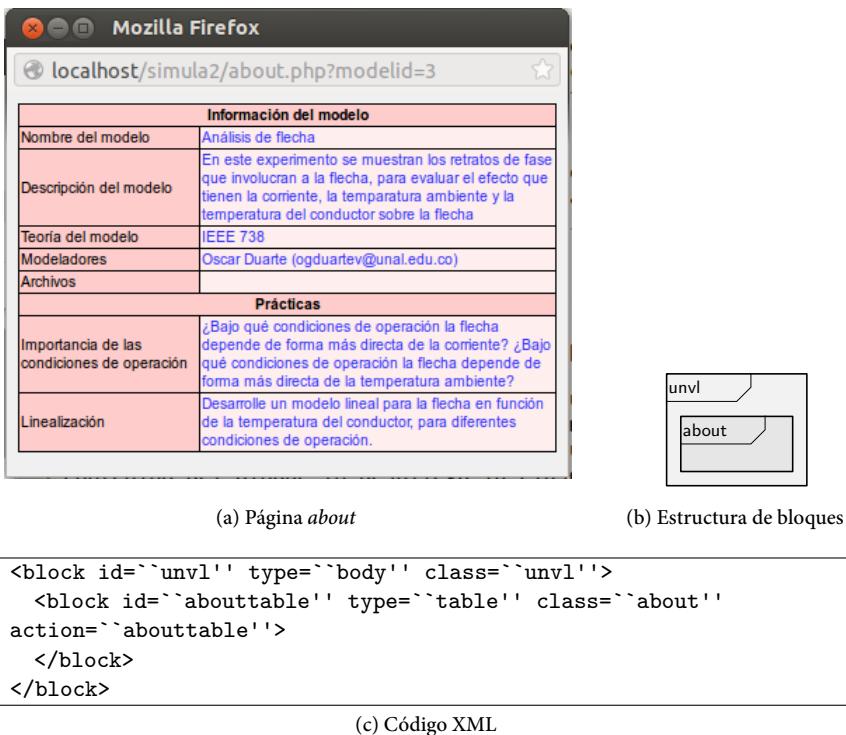


Figura 6.13 Página *about*. Estructura de bloques y código XML asociado

Al estructurar las páginas por bloques anidados, se obtienen varias ventajas en el desarrollo y la administración del sistema:

- El desarrollo de cada página es modular y ordenado.

<sup>4</sup>El código se encuentra en el archivo `page_structure/about.xml`

- Se puede obtener un prototipo rápido para evaluaciones preliminares.
- El posicionamiento de los bloques se puede controlar vía CSS.
- El control de la apariencia mediante archivos CSS es ordenado y modular.
- El desarrollo modular simplifica la depuración del código.
- Un mismo bloque puede reutilizarse en diferentes páginas.
- La presentación de un mismo bloque en páginas diferentes puede controlarse vía CSS.

Las figuras 6.14 y 6.15 muestran la manera como se anidan los bloques en las páginas de experimentación y administración, respectivamente. La función de cada uno de los bloques definidos en UNVirtualLab se presenta en la tabla 6.2. Por otra parte, la figura 6.16 muestra cómo se anidan los bloques de las páginas usadas para incrustar (embeber) tanto el modelo como la documentación dentro de páginas HTML externas.

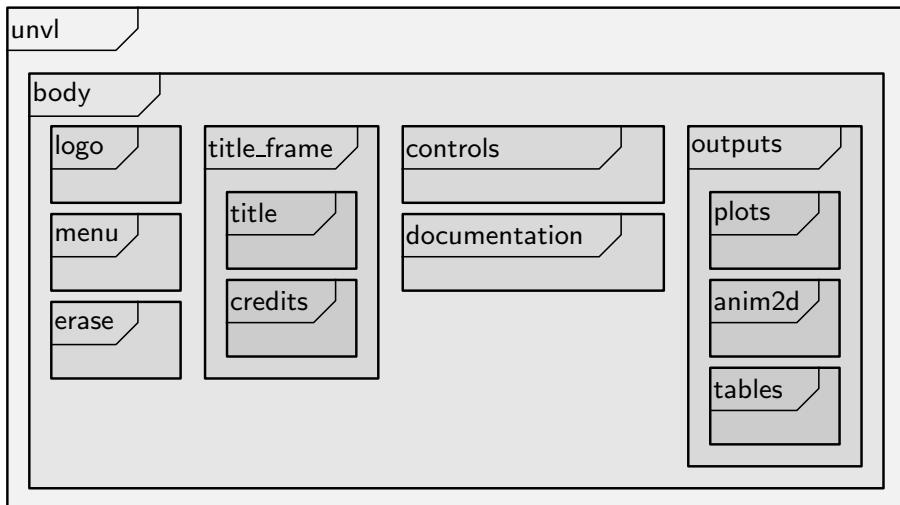


Figura 6.14 Estructura de la interfaz de experimentación

Tabla 6.2 Funcionalidad de los bloques de UNVirtualLab

Bloque (id)	Función
<b>Bloques generales y de experimentación</b>	
anim2d	Despliegue de las animaciones en 2D asociadas a la planta de experimentación.
body	Cuerpo principal de la página de experimentación.
controls	Despliegue de los controles de los parámetros de la planta de experimentación modificables por el usuario.
credits	Despliegue del enlace a la página de información de referencia de la planta de experimentación.
documentation	Despliegue del archivo (pdf) con la documentación de la planta de experimentación.
erase	Borrado de los archivos temporales del experimento; este bloque no genera código HTML y su efecto es invisible para el experimentador.
logo	Despliegue del logo de UNVirtualLab.
menu	Despliegue del menú de selección de secciones y plantas de experimentación en la página de experimentación.
outputs	Elemento contenedor de la presentación de los resultados del experimento.
plots	Despliegue de las curvas de resultados asociadas al experimento.
tables	Despliegue de las tablas de datos con los resultados del experimento.
title	Despliegue del título de la planta de experimentación.
titleframe	Elemento contenedor del título de la planta de experimentación y del enlace a la página de información de referencia del mismo.
unvl	Elemento contenedor de la página de experimentación.
<b>Bloques específicos de administración</b>	
admin_logout	Despliegue del botón para cerrar la sesión de administración.
admin_modelfiles	Despliegue de los controles para cargar los archivos asociados al modelo.
admin_2danim	Despliegue del espacio de edición de las animaciones 2D.
admin_2deffect	Despliegue del espacio de edición de los efectos de animación 2D.
admin_control	Despliegue del espacio de edición de los controles de parámetros.
admin_login	Despliegue de la página de inicio de sesión de administración.
admin_model	Despliegue del espacio de edición de los modelos de las plantas de experimentación.
admin_practice	Despliegue del espacio de edición de los experimentos sugeridos.
admin_plot	Despliegue del espacio de edición de las curvas de resultados de experimentos.
admin_section	Despliegue del espacio de edición de las secciones de plantas de experimentación.
body_admin	Cuerpo principal de las páginas de administración.
unvl_admin	Elemento contenedor de las páginas de administración.

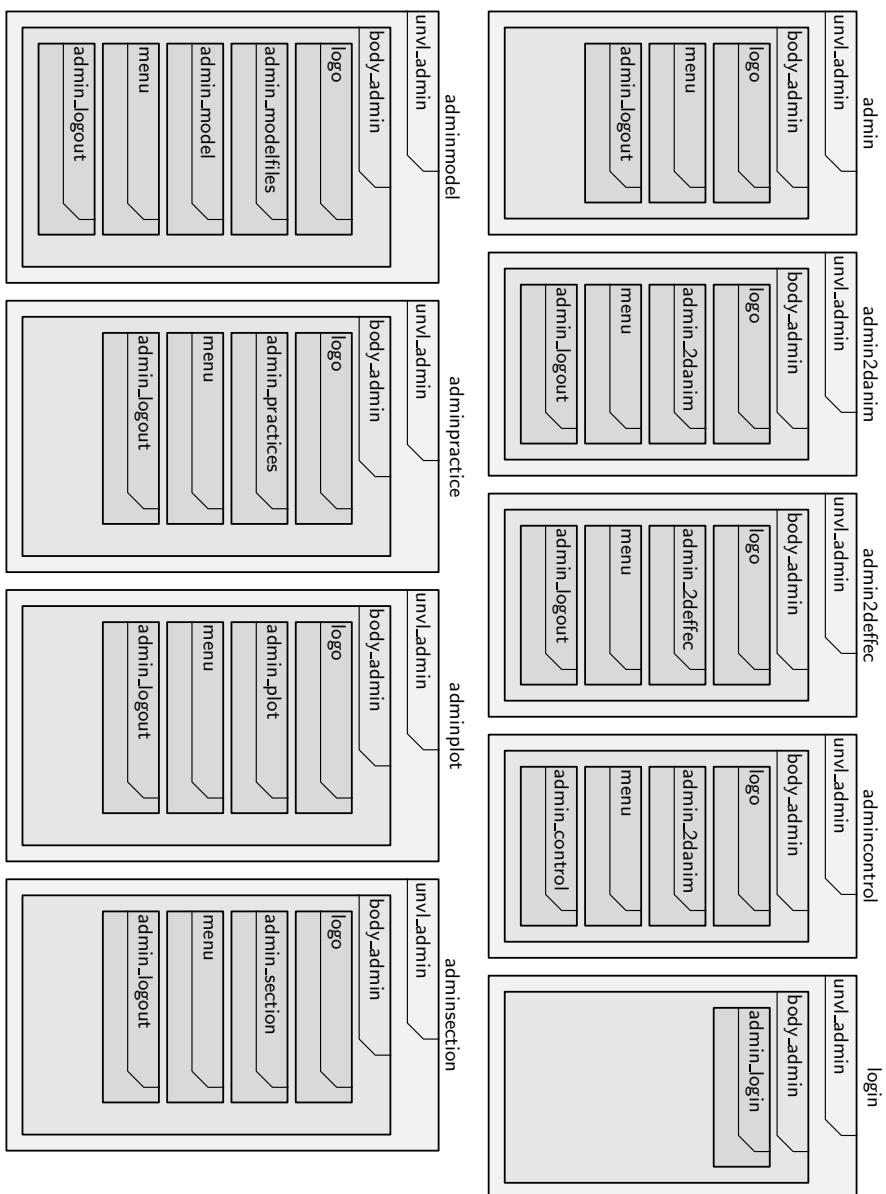


Figura 6.15 Estructura de las páginas de administración

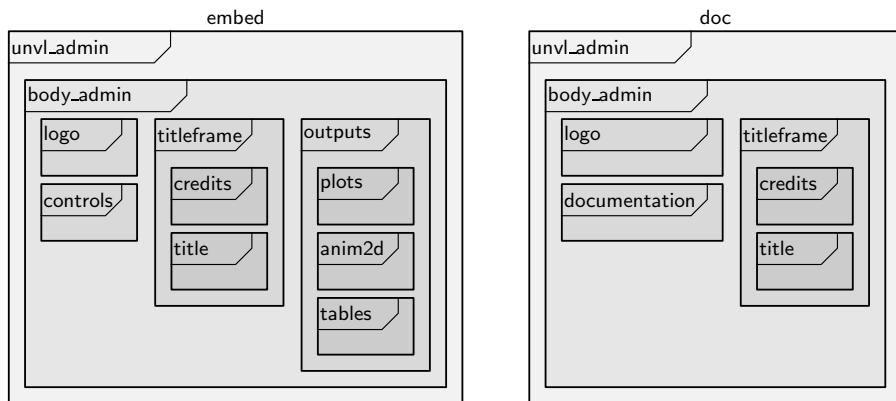


Figura 6.16 Estructura de la interfaz para embeber el modelo y la documentación

## 6.5 IMPLEMENTACIÓN EN PHP

Los *scripts* de UNVirtualLab 2.0 han sido escritos en lenguaje php siguiendo la metodología de programación orientada a objetos<sup>5</sup>. Las clases diseñadas se pueden agrupar en dos conjuntos:

- Clases enfocadas en el experimentador: implementan las funcionalidades requeridas por el usuario experimentador (véase figura 6.4); se presentan en la sección 6.5.1.
- Clases enfocadas en el administrador: implementan las funcionalidades requeridas por el usuario administrador (véase figura 6.5); se presentan en la sección 6.5.2.

La mayoría de las clases tiene por propósito desplegar y gestionar uno de los bloques de la interfaz. La colección de bloques y su disposición en pantalla se explican en la sección 6.4.

### 6.5.1 Clases enfocadas en el experimentador

La figura 6.17 muestra la jerarquía de clases enfocadas en el experimentador. Como puede observarse, *block* es la clase principal, que sirve de clase abstracta para las demás. Específicamente, *block* implementa las siguientes funciones:

---

<sup>5</sup>Algunos trozos de *software* utilizan también lenguaje javascript y ecmascript.

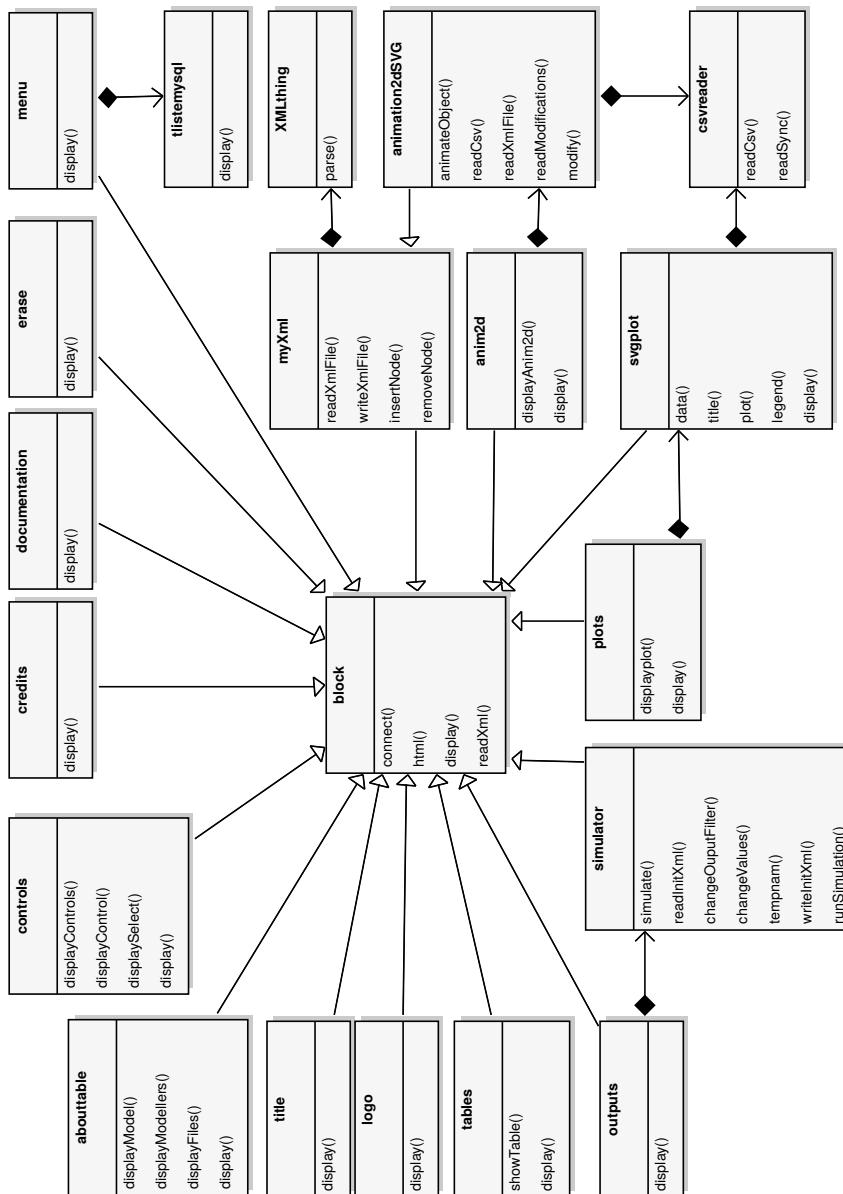


Figura 6.17 Diagrama UML de clases. Solo se muestran algunas funciones

Block 1. Realiza la conexión a la base de datos a través de la función `connect()`.

Block 2. Despliega la estructura básica de las páginas HTML a través de la función `html()`. El despliegue se realiza con los siguientes pasos:

1. Se lee el archivo XML que define la estructura de la página (función `readXml()`).
2. Se despliega el encabezado de la página HTML (función `opener()`).
3. Se despliega la información del bloque definido en el archivo XML (función `htmlBlock()`) a través de los siguientes pasos:
  - 3.1. Se despliega la apertura del bloque HTML.
  - 3.2. Se crea un objeto de la clase definida en el archivo XML para ese bloque.
  - 3.3. Se invoca la función `display()` del objeto creado. Esta clase es convenientemente redefinida en cada una de las clases herederas para desplegar la información pertinente. Los tres pasos anteriores se realizan a través de la función `htmlSimpleOpen()`.
  - 3.4. Se despliega la información de los bloques anidados llamando de forma iterativa la función `htmlBlock()`.
  - 3.5. Se despliega el cierre del bloque HTML (función `htmlSimpleClose()`).
4. Se despliega el cierre de la página HTML (función `closer()`).

Block 3. Despliega las cadenas de texto en el idioma adecuado a través de la función `text()` (ver sección 6.6.1).

Block 4. Captura el identificador del objeto que debe desplegarse, a través de la función `x_id()`.

Block 5. Valida que la información proveniente de los formularios corresponda al tipo declarado (función `validateValue()`).

La tabla 6.3 presenta una breve descripción de cada una de las demás clases enfocadas en el experimentador, explicando los aspectos relevantes de la lógica de implementación.

Tabla 6.3 Clases orientadas al experimentador

Clase	Descripción
<b>abouttable</b>	Despliega la información de referencia de un modelo específico. La información se presenta como una tabla HTML organizada así: <ul style="list-style-type: none"> <li>▪ Información general:] Nombre, descripción y bibliografía de referencia para el modelo (función <code>displayModel()</code>).</li> <li>▪ Modeladores: información sobre los desarrolladores del modelo (función <code>displayModellers()</code>).</li> <li>▪ Archivos: enlaces a los archivos descargables asociados al modelo (función <code>displayFiles()</code>).</li> <li>▪ Enlaces para incrustar: código HTML necesario para incrustar el modelo y la documentación. También se muestra el enlace directo al archivo pdf de documentación.</li> <li>▪ Prácticas: preguntas guías sugeridas como ideas para formular experimentos alrededor del modelo (función <code>displayPractices()</code>).</li> </ul>
<b>anim2d</b>	Despliega cada una de las animaciones 2D asociadas al modelo (función <code>displayAnim2d()</code> ), utilizando un objeto de la clase <code>animation2dSVG</code> .
<b>animation2dSVG</b>	Despliega una animación 2D con los resultados de una simulación. La animación se genera en lenguaje SVG. Los pasos que se ejecutan (en la función <code>animateObject()</code> ) son: <ol style="list-style-type: none"> <li>1. Se leen los resultados de la simulación. Los resultados están en un archivo temporal en formato CSV que se aloja en la carpeta work. Para la lectura se utiliza un objeto de la clase <code>csvreader</code>.</li> <li>2. Se lee el archivo SVG que será modificado con los datos de la simulación. Para la lectura se utilizan las funciones de la clase padre <code>MyXml</code>. La estructura se transforma en un árbol implementado con arreglos.</li> <li>3. Se leen desde la base de datos las animaciones que son controladas por los resultados de la simulación (función <code>readModifications()</code> y <code>sync()</code>).</li> <li>4. Se modifica la estructura del árbol con las animaciones correspondientes (funciones <code>modify()</code>, <code>modifyXml()</code> e <code>insertAnimation()</code>).</li> </ol>
<b>configuration</b>	Lee el archivo de configuración <code>config/unvlabconfig.txt</code> <sup>6</sup>
<b>controls</b>	Despliega el contenido del bloque que aloja los controles mediante los que el experimentador puede modificar los parámetros del modelo (función <code>displayControls()</code> ). Cada control se despliega mediante la función <code>displayControl()</code> ; si el control es del tipo 'select' se emplea la función <code>displaySelect()</code> . Los controles utilizan unas funciones en lenguajes JavaScript y ECMAScript que se escriben en línea a través de las funciones <code>javascript()</code> y <code>ecmascript()</code> .
<b>credits</b>	Despliega el contenido del bloque <code>credits</code> , que contiene un enlace que crea una nueva ventana con información de referencia del modelo específico.

<sup>6</sup>El autor de esta clase es Thomas Schmidt. La clase se distribuye bajo licencia libre Apache. Ha sido descargada de [link:5].

Tabla 6.3 Clases orientadas al experimentador

Clase	Descripción
<code>csvreader</code>	Lee un archivo de resultados de simulación en formato CSV y genera una arreglo PHP con los datos. La función <code>readCsv()</code> devuelve un arreglo de arreglos con los datos numéricos de cada variable simulada. La función <code>readSync</code> devuelve un arreglo de arreglos asociativos con los datos numéricos de cada variable simulada en los que la clave es el valor de la variable tiempo (time).
<code>documentation</code>	Despliega el archivo pdf de documentación asociado al modelo. El archivo debe tener por nombre <code>documentation.pdf</code> y debe alojarse en la carpeta <code>files/ID/doc</code> , en donde ID es el identificador del modelo.
<code>erase</code>	Borra los archivos temporales que se generan en cada simulación.
<code>logo</code>	Despliega el logo de la aplicación, con un enlace a la página del proyecto.
<code>menu</code>	Despliega el menú de selección de secciones y modelos. Para ello, utiliza un objeto de la clase <code>tlistemysql</code> .
<code>myXml</code>	Lee el contenido de archivos en formato XML. Esta clase ha sido construida a partir de la clase <code>XMLThing</code> , a la que se le han añadido las funciones de escritura ( <code>writeXmlFile()</code> , <code>writeXmlString</code> , <code>writeNodeAttributes</code> y <code>writeNodeString()</code> ), inserción de nodos ( <code>insertNode()</code> ) y eliminación de nodos ( <code>removeNode()</code> ).
<code>outputs</code>	Lanza una simulación a través de un objeto de la clase <code>simulator</code> .
<code>plots</code>	Despliega cada una de las gráficas asociadas al modelo (función <code>displayPlot()</code> ) utilizando un objeto de la clase <code>SVGplot</code> .
<code>simulator</code>	Ejecuta una simulación con los parámetros definidos por el usuario a través de los siguientes pasos (función <code>simulate()</code> ): <ol style="list-style-type: none"> <li>1. Se lee desde la base de datos el nombre del archivo ejecutable del modelo (función <code>exeFileName()</code>).</li> <li>2. Se lee el archivo de parámetros del modelo original, es decir, del modelo sin las modificaciones del usuario (función <code>readInitXml()</code>). La información se almacena en un árbol.</li> <li>3. Se leen desde la base de datos los nombres de las variables que deberán desplegarse después de realizar la simulación; con este conjunto se construye el filtro de salida de la simulación (función <code>changeOutputFilter()</code>).</li> <li>4. Se actualizan los valores del árbol de parámetros con los datos seleccionados por el usuario; estos datos se reciben desde un formulario (función <code>changeValues()</code>).</li> <li>5. Se asigna un nombre aleatorio (supóngase que es ABCDEF). Se construye el archivo de parámetros actualizado <code>work/INI_ABCDEF</code> (función <code>writeInitXml()</code>).</li> <li>6. Se corre el archivo ejecutable con el archivo de parámetros actualizado y los resultados se salvan en <code>work/INI_ABCDEF_res.csv</code>; el nombre de este archivo se almacena en el arreglo de sesión <code>\$_POST[]</code> (función <code>runSimulation()</code>).</li> </ol>

Tabla 6.3 Clases orientadas al experimentador

Clase	Descripción
svgplot	Despliega una figura asociada a un modelo con los resultados de una simulación. La figura se construye en formato SVG mediante los siguientes pasos: <ol style="list-style-type: none"> <li>1. Se leen los parámetros de dibujo tales como el tamaño de la figura (función <code>settings()</code>).</li> <li>2. Se lee desde la base de datos qué curvas deben dibujarse y a qué variables simuladas corresponden; los valores numéricos de esas variables se leen desde el archivo de resultados usando un objeto de la clase <code>csvreader</code> (función <code>data()</code>).</li> <li>3. Se despliega el código SVG de la figura (función <code>display()</code>).</li> </ol>
tables	Despliega la tabla con los valores numéricos de las variables simuladas (función <code>showTable()</code> ). Para descargar los valores se crea un enlace al script <code>send.php</code> , que recibe los datos serializados.
title	Lee desde la base de datos el nombre del modelo y lo despliega en el bloque <code>title</code> .
tlistemysql	Despliega el menú anidado de secciones y modelos. La clase ha sido construida a partir de la clase <code>PhpTliste</code> que se distribuye con licencia GNU <sup>7</sup> .
XMLThing	Lee un archivo con formato XML <sup>8</sup> .

### 6.5.2 Clases enfocadas en el administrador

Las figuras 6.18 y 6.19 muestran la jerarquía de clases enfocadas en el administrador. La clase principal es `adminblock`, que es heredera de la clase `block`.

El propósito fundamental de la clase `adminblock` es el de gestionar la interfaz entre el usuario y la base de datos. Esta clase se encarga de desplegar la información de la base de datos en formularios y de administrar las actualizaciones de la información. Cada clase hija se especializa en una tabla específica de la base de datos. Para ello, se han definido los siguientes miembros de clase que se redefinen en los hijos<sup>9</sup>:

- a. `table`: almacena el nombre de la tabla que gestiona.
- b. `fields`: arreglo que almacena información sobre los campos de la tabla. Cada elemento del arreglo es, a su vez, un arreglo con las siguientes claves:
  - `dbname`: nombre del campo.
  - `showname`: texto para desplegar en el formulario como nombre del campo.

<sup>7</sup>El autor de esta clase es Berthou ([link:6]), y se encuentra disponible en [link:7].

<sup>8</sup>Clase desarrollada por <wickedfather@hotmail.com> y disponible en [link:8].

<sup>9</sup>En el apéndice C se muestra el contenido de estos campos para las clases hijas.

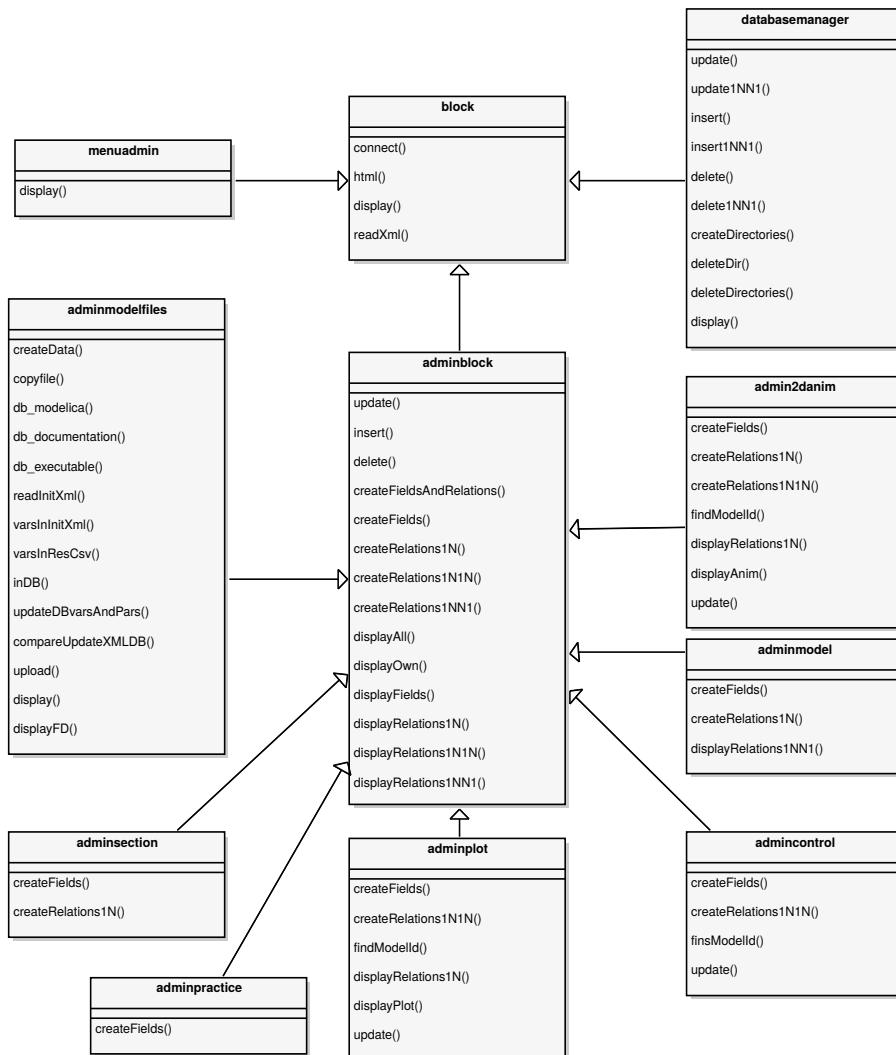


Figura 6.18 Diagrama UML de clases de administración. Solo se muestran algunas funciones

- **type**: tipo de dato. Según el tipo de dato se despliega un control HTML específico. Los tipos de datos válidos y los controles asociados se muestran en la tabla 6.4.
- c. **relations1N**: arreglo que almacena el listado de relaciones del tipo  $1 \leftrightarrow \infty$  asociadas a la tabla. Cada elemento del arreglo es, a su vez, un arreglo con las siguientes claves:

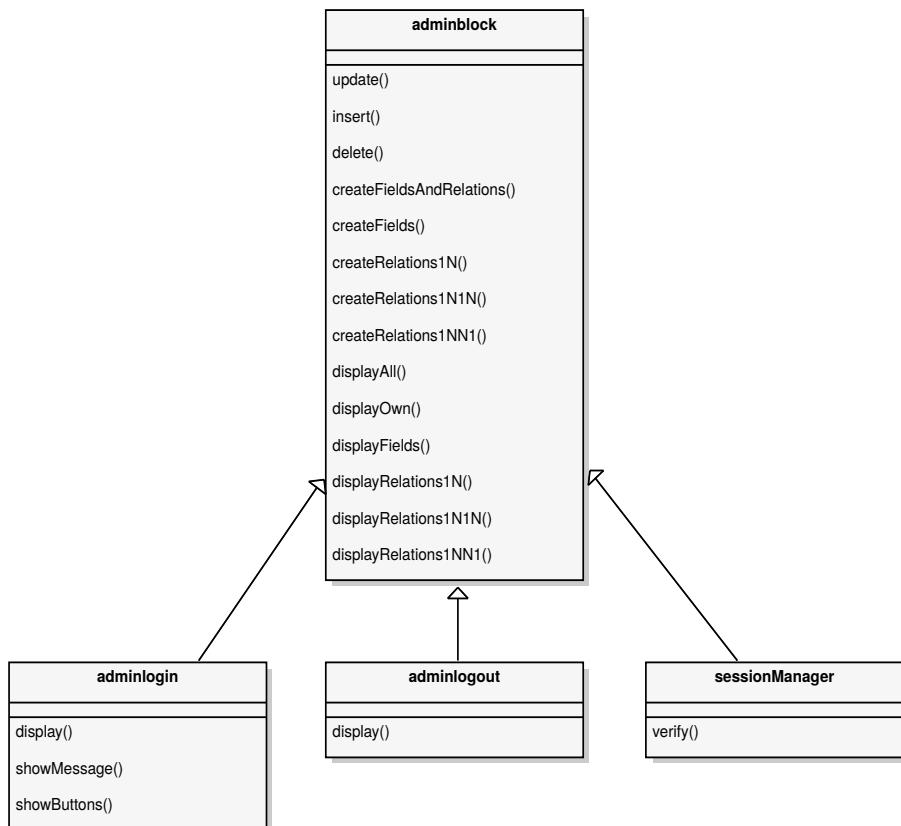


Figura 6.19 Diagrama UML de clases de manejo de sesiones. Solo se muestran algunas funciones

- **title**: texto para desplegar en el formulario como título del grupo.
- **table**: nombre de la tabla secundaria.
- **linkname**: nombre base del identificador para usar en el formulario.
- **id1**: nombre del campo en la tabla primaria de la relación.
- **id2**: nombre del campo en la tabla secundaria de la relación.
- **idvalue**: valor del campo en la tabla primaria de la relación.
- **showdbname**: texto para desplegar en el formulario como nombre de la relación.
- **show**: nombre del campo en la tabla secundaria que contiene el texto a desplegar en el formulario como valor actual de la relación.

Tabla 6.4 Tipo de datos válidos para editar mediante la clase `adminblock` y control HTML desplegado

Tipo de dato	Control HTML
<code>bool</code>	<code>&lt;select&gt;</code> con dos opciones
<code>check</code>	<code>&lt;input type="checkbox"&gt;</code>
<code>color</code>	<code>&lt;input type="color"&gt;</code> de la clase <code>color</code> <sup>10</sup>
<code>fixed</code>	texto no editable
<code>float</code>	<code>&lt;input type="float"&gt;</code>
<code>int</code>	<code>&lt;input type="int"&gt;</code>
<code>longtext</code>	<code>&lt;textarea&gt;</code>
<code>model_link</code>	<code>&lt;a href="#&gt;</code> enlace al modelo
<code>select options</code>	<code>&lt;select&gt;</code> con opciones definidas mediante un arreglo
<code>select section</code>	<code>&lt;select&gt;</code> las opciones son el contenido de la tabla <code>sections</code>
<code>select variable</code>	<code>&lt;select&gt;</code> las opciones son el contenido de la tabla <code>variables</code>
<code>select parameter</code>	<code>&lt;select&gt;</code> las opciones son el contenido de la tabla <code>parameters</code>
<code>text</code>	<code>input</code>

- d. `relations1N1N`: arreglo que almacena dos relaciones concatenadas, ambas del tipo  $1 \leftrightarrow \infty$  asociadas a la tabla. Cada elemento del arreglo es un arreglo semejante a los usados en el arreglo `relations1N`, con una clave más, denominada `subfields`. En esta clave se aloja un arreglo semejante a los usados en el arreglo `field`.
- e. `relations1NN1`: arreglo que almacena el listado de relaciones del tipo  $\infty \leftrightarrow \infty$  asociadas a la tabla. Cada elemento del arreglo es, a su vez, un arreglo con las siguientes claves:
- `title`: texto para desplegar en el formulario como título del grupo.
  - `tableLink`: nombre de la tabla de la relación.
  - `table2`: nombre de la tabla secundaria.
  - `linkname`: nombre base del identificador para usar en el formulario.
  - `id1`: nombre del campo en la tabla primaria de la relación.
  - `id2`: nombre del campo en la tabla secundaria de la relación.
  - `idLink1`: nombre del campo en la tabla primaria como aparece en la tabla de la relación.

---

<sup>10</sup>Utiliza el script `jscolor` desarrollado en lenguaje javascript por Jan Odvarko ([link:9]), distribuido con licencia libre GNU, y disponible en [link:10].

- idLink2: nombre del campo en la tabla secundaria como aparece en la tabla de la relación.
- showdbname: texto para desplegar en el formulario como nombre de la relación.
- show: nombre del campo en la tabla secundaria que contiene el texto que se desplegará en el formulario como valor actual de la relación.
- order: nombre de los campos de la tabla secundaria que sirven de criterio de ordenamiento ascendente.
- subfields: arreglo semejante a los usados en el arreglo field.

Especificamente, la clase adminblock implementa las siguientes funciones:

AdminBlock 1. Crea los arreglos que definen los campos y relaciones de cada tabla a través de las siguientes funciones que se redefinen en cada clase hija:

- createFieldsAndRelations()
- createFields()
- createRelations1N()
- createRelations1N1N()
- createRelations1NN1()

AdminBlock 2. Actualiza la base de datos con la información proveniente de un formulario, a través de las funciones update(), update1N1N() y update1NN1(). Estas funciones utilizan un objeto de la clase databasemanager.

AdminBlock 3. Crea nuevos campos en la base de datos con instrucciones provenientes de un formulario a través de las funciones insert() e insert1NN1(). Estas funciones utilizan un objeto de la clase databasemanager.

AdminBlock 4. Elimina campos en la base de datos con instrucciones provenientes de un formulario a través de las funciones delete() y delete1NN1(). Estas funciones utilizan un objeto de la clase databasemanager.

AdminBlock 5. Despliega los formularios para edición de información y adición y eliminación de elementos a través de las funciones:

- `displayAll()`
- `displayOwn()`
- `displayPostOwn()`
- `displayRelations1N()`
- `displayOneRelation1N()`
- `displayRelations1N1N()`
- `displayOneRelation1N1N()`
- `displayRelations1NN1()`
- `displayOneRelation1NN1()`

La tabla 6.5 presenta una breve descripción de cada una de las demás clases enfocadas en el administrador, explicando los aspectos relevantes de la lógica de implementación.

Tabla 6.5 Clases orientadas al administrador

Clase	Descripción
<code>admin2danim</code>	Clase especializada en la edición de la tabla <code>2danimations</code> que contiene las animaciones en 2D asociadas a un modelo.
<code>admin2deffect</code>	Clase especializada en la edición de la tabla <code>2deffects</code> que contiene los efectos asociados a una animación 2D.
<code>admincontrol</code>	Clase especializada en la edición de las tablas <code>control_groups</code> y <code>controls</code> que contienen los controles y grupos de controles mediante los que el usuario puede ajustar los parámetros de un modelo.
<code>adminlogin</code>	Despliega el formulario de acceso a las sesiones de administración.
<code>adminlogout</code>	Despliega el formulario de cierre de la sesión de administración.
<code>adminmodel</code>	Clase especializada en la edición de la tabla <code>models</code> que contiene los modelos de simulación.
<code>adminmodelfiles</code>	Clase especializada en cargar los archivos asociados a un modelo y en actualizar la información correspondiente en la base de datos. Los tipos de archivo que se gestionan son: <ul style="list-style-type: none"> <li>▪ Ejecutable: un archivo comprimido del tipo <code>.tar.gz</code> que debe contener tres archivos: el archivo ejecutable (<code>XXX</code>), el archivo con los valores de los parámetros de simulación <code>XXX_init.xml</code> y el archivo con los resultados de una simulación <code>XXX_res.csv</code>. Estos archivos se generan usando OpenModelica. Los archivos se guardan en la carpeta <code>files/ID/bin</code>, en donde <code>ID</code> es el identificador del modelo.</li> <li>▪ Documentación: un archivo del tipo <code>.pdf</code> con la documentación asociada al modelo. El archivo se renombra como <code>documentation.pdf</code> y se guardan en la carpeta <code>files/ID/doc</code>.</li> </ul>

<code>adminmodelfiles</code>	<ul style="list-style-type: none"> <li>Modelica: un archivo del tipo <code>.tar.gz</code> con el código fuente asociado al modelo (en lenguaje Modelica). El archivo se renombra como <code>source.tar.gz</code> y se guarda en la carpeta <code>files/ID/modelica</code>.</li> </ul>
<code>adminplot</code>	Clase especializada en la edición de las tablas <code>plots</code> y <code>curves</code> que contienen las figuras y curvas asociadas a un modelo.
<code>adminpractice</code>	Clase especializada en la edición de la tabla <code>practices</code> que contiene los experimentos sugeridos asociados a un modelo.
<code>adminsection</code>	Clase especializada en la edición de la tabla <code>sections</code> que contiene las secciones que sirven para organizar los modelos.
<code>menuadmin</code>	Despliega el menú de selección de secciones y modelos. Para ello, utiliza un objeto de la clase <code>tlistemysql</code> .
<code>databasemanager</code>	Clase especializada en gestionar a bajo nivel la base de datos. Implementa las funciones básicas de insertar, actualizar y eliminar registros. También se encarga de la gestión de los subdirectorios asociados a cada modelo, que se alojan dentro del directorio <code>files</code>
<code>sessionManager</code>	Verifica si los datos suministrados en el formulario de inicio de sesión corresponden o no a un usuario autorizado. En caso afirmativo inicia la sesión.

## 6.6 CONFIGURACIÓN DE LA INTERFAZ

### 6.6.1 Internacionalización - I18n

Puede cambiarse el idioma de los textos de la interfaz de UNVirtualLab<sup>11</sup>. Para modificar el idioma deben seguirse los siguientes pasos<sup>12</sup>:

1. Copiar el archivo `locale/es.inc` o el archivo `locale/en.inc` en uno nuevo de nombre `locale/XX.inc`, en donde XX es una cadena arbitraria que identifica el idioma. Se sugiere utilizar el estándar ISO 639 para seleccionar el valor de la cadena (véase, por ejemplo, [Int06]).
2. Traducir las cadenas de caracteres en el nuevo archivo al idioma seleccionado. La traducción debe hacerse en las cadenas que están a la derecha del carácter '='.
3. Modificar la opción `locale` en el archivo `config/unvlconfig.txt`. La línea correspondiente debe quedar así:

```
:locale = XX
```

<sup>11</sup>Esto no significa, sin embargo, que UNVirtualLab tenga la opción de manejar simultáneamente contenidos en diferentes idiomas. El contenido alojado en la base de datos es monolingüe.

<sup>12</sup>UNVirtualLab 2.0 incluye los archivos `es.inc` y `en.inc` para español e inglés, respectivamente. Por tanto, para estos dos idiomas, los dos primeros pasos deben obviarse.

## 6.6.2 Apariencia

UNVirtualLab utiliza temas especificados en hojas de estilo CSS para definir la apariencia de las páginas HTML. Es posible cambiar la presentación de los contenidos modificando las especificaciones CSS o creando un nuevo tema.

Los temas se alojan en el directorio `themes`. Cada tema debe tener un directorio propio cuyo nombre es igual al nombre del tema<sup>13</sup>. UNVirtualLab busca dentro de esa carpeta las definiciones CSS en dos archivos de nombre `style.css` y `styleSVG.css`. Se sugiere reservar el archivo `styleSVG.css` para las definiciones asociadas a las figuras SVG que muestran los resultados de una simulación.

Los temas CSS son inherentemente anidados. La anidación de los bloques CSS que se pueden personalizar en UNVirtualLab a través de los temas se muestra en las figuras 6.20 y 6.21. En la primera de esas figuras se presentan los bloques de la interfaz para el usuario experimentador, y en la segunda para el usuario administrador<sup>14</sup>.

Las definiciones de un bloque contenedor pasan automáticamente a los bloques contenidos<sup>15</sup>, en donde pueden ser redefinidas. En el apéndice D se consignan las definiciones de estilo presentes en el tema *unbasic*. No obstante, un nuevo tema puede definir otros parámetros de apariencia diferentes.

La organización de la interfaz puede modificarse también a partir de especificaciones CSS, ya que el tamaño y la posición de cada bloque pueden controlarse por este método. Las figuras 6.22 y 6.23 muestran la posición en pantalla de los principales bloques que visualizan los usuarios experimentador y administrador, respectivamente, cuando se utiliza el tema `unvlabasic`.

---

<sup>13</sup>A manera de ejemplo, UNVirtualLab 2.0 tiene por defecto definido el tema `unvlabasic` en la carpeta `themes/unvlabasic`.

<sup>14</sup>

Se han definido también los bloques `div.controlsEmbed` `div.outputsEmbed` y `div.documentationEmbed` para adecuar la apariencia de la interfaz cuando se incrusta en páginas HTML externas. La estructura de estos bloques es semejante a la de los bloques `div.controls` `div.outputs` y `div.documentation`, respectivamente.

<sup>15</sup>De allí el término cascada de la sigla CSS, Cascading Style Sheets.

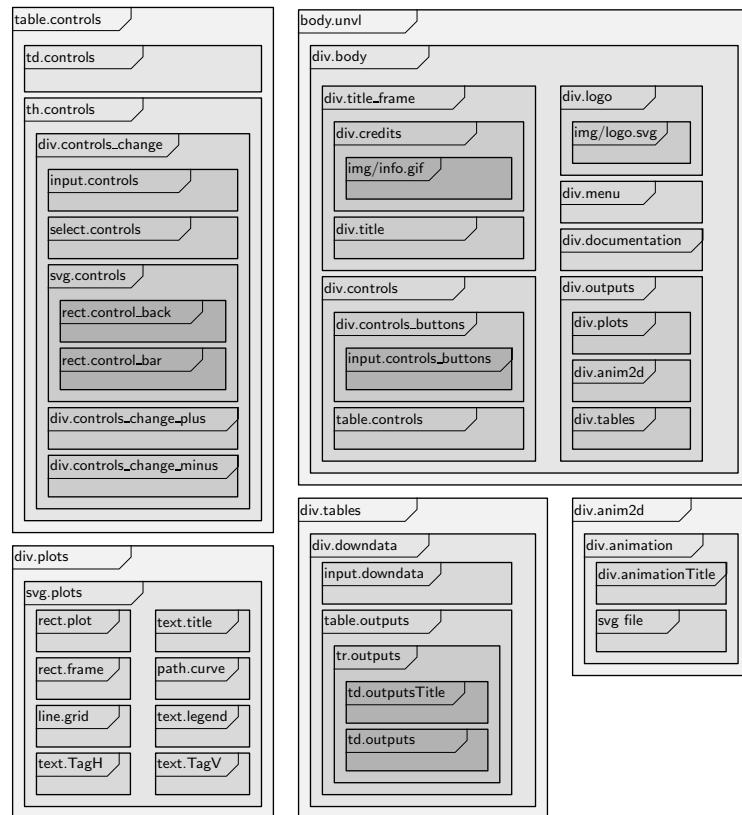


Figura 6.20 Anidación de los bloques CSS personalizables

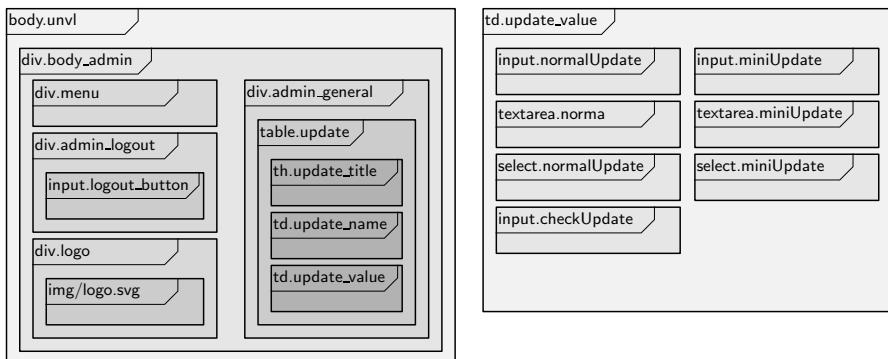


Figura 6.21 Anidación de los bloques CSS personalizables del módulo de administración.

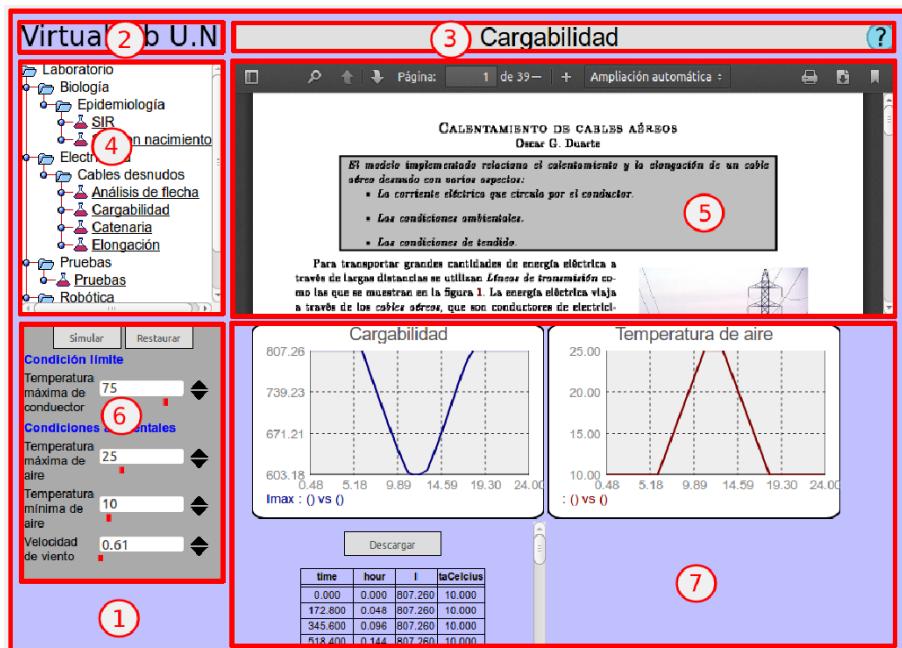


Figura 6.22 Bloques div principales: 1) div.body, 2) div.logo, 3) div.title\_frame, 4) div.menu, 5) div.documentation, 6) div.controls, y 7) div.outputs

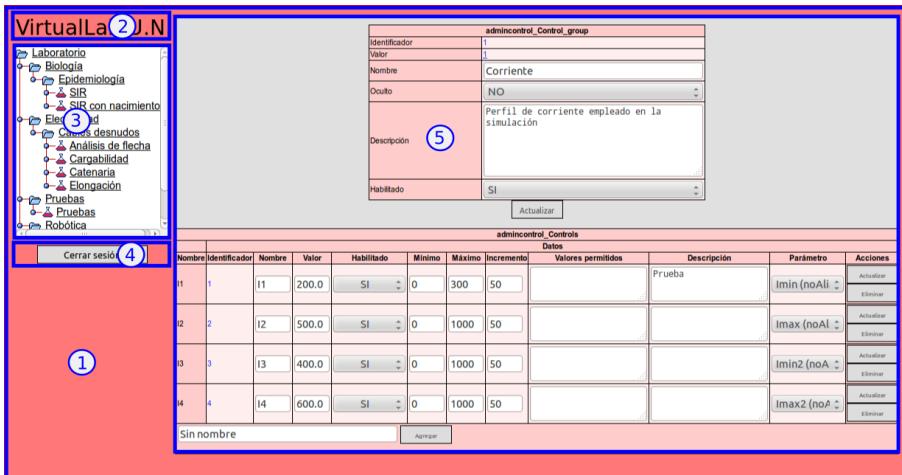


Figura 6.23 Bloques div principales del módulo de administración: 1) div.body\_admin, 2) div.logo, 3) div.menu, 4) div.admin\_logout, y 5) div.admin\_general

### 6.6.3 Estilo de las figuras

Las opciones para modificar la apariencia de las figuras que muestran los resultados de una simulación pueden clasificarse así:

- a. Definición del tamaño: el tamaño de las figuras se especifica en dos archivos diferentes (ambos deben modificarse para definir un nuevo tamaño):
  - En el archivo config/unvlconfig.txt, las opciones plotWidth y plotHeight definen el ancho y alto de la figura, respectivamente.
  - En el archivo themes/unvlabasic/styleSVG.css (o en el archivo correspondiente del tema en uso), las opciones width y height de la clase svg.plot definen el ancho y alto del bloque SVG que contiene la figura, respectivamente.
- b. Definiciones de cada figura: mediante la interfaz de administrador, es posible definir para cada figura de forma independiente los siguientes parámetros:
  - Rango de la escala horizontal: puede ser establecido de forma automática por UNVirtualLab o definido manualmente por el usuario.
  - Rango de la escala vertical: puede ser establecido de forma automática por UNVirtualLab o definido manualmente por el usuario.
  - Número de divisiones horizontales en la grilla.
  - Número de divisiones verticales en la grilla.
  - Color de cada una de las curvas graficadas.
- c. Definiciones del estilo: UNVirtualLab genera las figuras en formato SVG. Este formato es inherentemente anidado, lo que significa que las figuras se construyen a partir de regiones anidadas. Estas regiones tienen forma rectangular, y se muestran en la figura 6.24. El estilo de cada región, a su vez, tiene una especificación CSS<sup>16</sup>. De esta forma pueden modificarse simultáneamente para todas las figuras parámetros tales como:
  - Color de fondo.
  - Tipo y tamaño de letra.
  - Color y ancho de los trazos.

---

<sup>16</sup>En el tema unvlabasic estas especificaciones se encuentran en el archivo themes/unvlabasic/styleSVG.css.

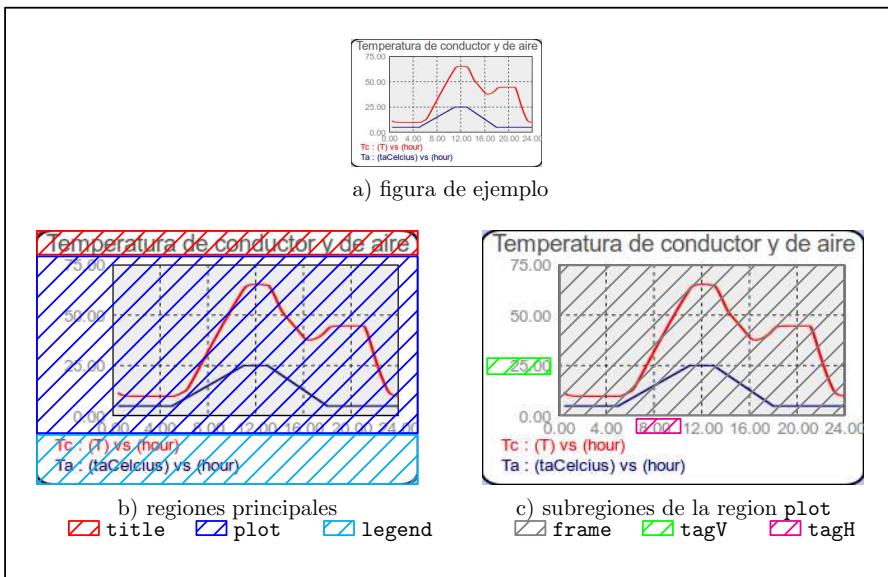


Figura 6.24 Regiones de las figuras asociadas a un modelo

- d. Definiciones de código: la clase `svgplot` (véase archivo `svgplot.php`) está encargada de dibujar las figuras. En ella se definen los tamaños y las posiciones de las regiones que se muestran en la figura 6.24. Estas dimensiones están dadas como porcentajes de la dimensión de la región contenedora. Las dimensiones que pueden modificarse se definen en las líneas de código:

```
var $titleHeight=10;
var $plotHeight=80;
var $legendHeight=5;
var $framePositionX=20;
var $framePositionY=5;
var $frameWidth=75;
var $frameHeight=85;
var $tagHeight=10;
var $tagWidth=20;
var $legendSeparation=20;
```



7

MANUALES



En este capítulo se presentan los tres manuales desarrollados para UNVirtualLab: un “manual del administrador” (sección 7.1) en el que se detallan las tareas de instalación y mantenimiento del sistema; un “manual del modelado” (sección 7.2) en el que se presenta una guía para la elaboración de plantas experimentales con enfoque pedagógico, y un “manual del experimentador” (sección 7.3) en el que se muestra cómo usar las plantas experimentales alojadas en UNVirtualLab. Este último manual está complementado con un conjunto de videos tutoriales disponibles en línea<sup>1</sup>.

## 7.1 MANUAL DEL ADMINISTRADOR

### 7.1.1 Instalación del servidor

UNVirtualLab está diseñado para operar bajo cualquier sistema operativo que soporte servidores http, php y mysql. Las instrucciones que se presentan a continuación han supuesto que el sistema operativo es Linux y el servidor http es apache (ver [Fou05]). Las pruebas se han realizado sobre Ubuntu 16.04. No obstante, el procedimiento de instalación es semejante al de cualquier aplicación web estándar basada en la combinación php y mysql, por lo que no es difícil realizar la instalación en otros sistemas operativos.

Como se explica en la sección 6.3, es posible distinguir tres tipos de instalación del servidor:

- Servidor mínimo (véase figura 6.10).
- Servidor con compilación (véase figura 6.11).
- Servidor para el modelador (véase figura 6.12).

En la figura 7.1 se muestra la secuencia de pasos necesaria para instalar un servidor UNVirtualLab. Estos pasos de instalación se explican a continuación.

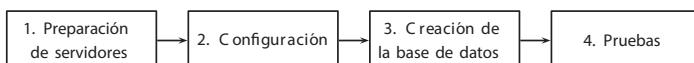


Figura 7.1 Proceso de instalación de UNVirtualLab

<sup>1</sup>A este conjunto de videos se accede haciendo clic sobre el logo de UNVirtualLab y seleccionando la opción “Manual del experimentador”.

## IN.1. Preparación de servidores:

IN 1.1 Efectuar una instalación estándar del servidor mysql ([Cor22]).

En este proceso de instalación se define el nombre del usuario de administración y su clave de usuario. Archivar estos datos para usarlos en el paso IN 3.1.

IN 1.2 Realizar una instalación estándar del servidor php.

IN 1.3 Realizar una instalación estándar del servidor http apache<sup>2</sup>.

IN 1.4 Configurar el servidor http apache:

IN 1.4.1 Crear o seleccionar un directorio en el que se instalarán los *scripts* de UNVirtualLab. En el contexto de este manual, la ruta del directorio se denomina DIRUNVL.

IN 1.4.2 Configurar apache para que tenga acceso al directorio DIRUNVL. Si el directorio raíz de apache es /var/www esto se consigue con los siguientes pasos:

IN 1.4.2.1 Crear un enlace simbólico:

```
ln -s DIRUNVL /var/www/unvl
```

IN 1.4.2.2 Editar el archivo sites-available/default, que usualmente está en el directorio /etc/apache2/, para adicionar el directorio DIRUNVL y bloquear el acceso al archivo DIRUNVL/config. Esto se consigue adicionando los bloques:

```
<Directory DIRUNVL/>
    Options Indexes FollowSymLinks MultiViews
    AllowOverride None
    Order allow,deny
    allow from all
</Directory>
<Directory DIRUNVL/config/>
    Allow from None
    Order allow,deny
</Directory>
```

IN 1.4.2.3 Reiniciar el servidor apache

```
sudo /etc/init.d/apache2 restart
```

IN 1.5 Si se desea realizar una instalación con compilación, es necesario instalar el compilador de OpenModelica<sup>3</sup>:

---

<sup>2</sup>Para algunas distribuciones de Linux, este paso está incluido en la instalación misma del sistema operativo.

<sup>3</sup>Las instrucciones actualizadas pueden consultarse en [link:11].

IN 1.5.1 Editar el archivo `/etc/apt/sources.list` para adicionar la línea:

```
deb http://build.openmodelica.org/apt
`lsb_release -cs` nightly
```

IN 1.5.2 Desde una línea de comando, importar la llave<sup>4</sup>:

```
wget -q http://build.openmodelica.org/apt/
openmodelica.asc -O- | sudo apt-key add -
```

IN 1.5.3 Desde una línea de comando, actualizar el listado de repositorios:

```
sudo apt-get update
```

IN 1.5.4 Desde una línea de comando instalar el compilador:

```
sudo apt-get install omc
```

IN 1.6 Si se desea realizar una instalación para el modelador, la instrucción del paso IN 1.5.3 debe remplazarse por

```
sudo apt-get install openmodelica
```

**IN.2. Configuración:** se instalan los *scripts* php de UNVirtualLab y se editan los archivos de configuración. Los pasos a seguir son:

IN 2.1 Copiar el archivo `unvl.tar.gz` en el directorio DIRUNVL

IN 2.2 Descomprimir el archivo `unvl.tar.gz`

```
cd DIRUNVL
gunzip unvl.tar.gz
tar -xf unvl.tar
```

Como resultado, el archivo `unvl.php` debe estar ubicado como `DIRUNVL/unvl.php`

IN 2.3 Editar el archivo de configuración `DIRUNVL/config/unvlconfig.txt`. El significado de cada una de las variables de configuración se consigna en la tabla 7.1.

IN 2.4 Asignar los permisos de lectura adecuados para el archivo de configuración. Con las siguientes instrucciones se otorga permiso de lectura solo al usuario dueño del archivo.

```
cd DIRUNVL/config
chmod 444 unvlconfig.txt
```

**IN.3. Creación de la base de datos:** el *script* `sql/crearSchema.php` se encarga de crear la base de datos `mysql`, los usuarios y los permisos necesarios. Para ello es necesario conocer el nombre y la clave de acceso

---

<sup>4</sup>La instrucción es un único comando. Se ha escrito en dos líneas por su extensión.

mysql de un usuario con permisos de creación y gestión de bases de datos (típicamente root). Por seguridad, esta información no debe quedar disponible en ningún archivo del servidor. Los pasos a seguir son los siguientes:

IN 3.1 Editar el script `sql/crearSchema.php`. En las líneas 4 y 5 deben editarse las variables `$username` y `$userpass` para indicar el nombre de usuario mysql y la clave de acceso con permisos de creación y gestión de bases de datos. Por ejemplo:

```
$username="root";
$userpass="rootpassword";
```

IN 3.2 Correr el script `sql/crearSchema.php`.

```
cd DIRUNVLL/sql
php crearSchema.php
```

Este *script* lee la información de configuración del archivo `config/unvlconfig.txt` y realiza las siguientes tareas:

- a. Se conecta con el servidor de la base de datos DBserver usando el usuario y la clave de acceso definidas en las variables `$username` y `$userpass` (archivo `sql/crearSchema.php`).
- b. Crea una base de datos cuyo nombre está definido por la variable `DBname` (archivo `config/unvlconfig.txt`).
- c. Corre el *script* `sql/unvl.sql` que se encarga de crear el *schema* de la base de datos creada en el paso b.
- d. Crea un usuario mysql cuyo nombre y clave de acceso están definidos por las variables `DBuser` y `DBuserpass`, respectivamente (archivo `config/unvlconfig.txt`).
- e. Le asigna al usuario creado en el paso d permiso para leer registros (SELECT) de todas las tablas de la base de datos creada en el paso b.
- f. Crea un usuario mysql cuyo nombre y clave de acceso están definidos por las variables `DBadmin` y `DBadminpass`, respectivamente (archivo `config/unvlconfig.txt`).
- g. Le asigna al usuario creado en el paso f permiso para leer, crear, editar y eliminar registros (SELECT, UPDATE, INSERT y DELETE) de todas las tablas de la base de datos creada en el paso b.
- h. Crea un usuario UNVirtualLab con rol de administrador cuyo nombre y clave de acceso están definidos en las variables

UNVLadmin y UNVLadminpass, respectivamente (archivo config/unvlconfig.txt).

- i. Crea una sección de modelos de nombre unvl en la base de datos creada en el paso b.

**IN 3.3 Borrar el archivo sql/crearSchema.php.**

```
rm DIRUNVL/sql/crearSchema.php
```

**IN 4. Pruebas:** las pruebas iniciales verifican que UNVirtualLab se encuentre disponible en la red, y que el acceso al usuario administrador esté habilitado.

**IN 4.1** Utilizando un programa navegador web, cargar la dirección definida por la variable URLbase (archivo config/unvlconfig.txt). El resultado debe ser la pantalla de navegación para el experimentador que se muestra en la figura 7.2. El mismo resultado debe obtenerse si se carga la dirección seguida de unvl.php.



Figura 7.2 Pantalla de UNVirtualLab para el usuario experimentador, inmediatamente después de una instalación fresca

**IN 4.2** Utilizando un programa navegador web, cargar la dirección definida por la variable URLbase seguida de admin.php. El resultado debe ser la pantalla de inicio de sesión de administración que se muestra en la figura 7.3.



Figura 7.3 Pantalla de UNVirtualLab para inicio de sesión de administrador, inmediatamente después de una instalación fresca.

IN 4.3 A partir de la página del numeral anterior, iniciar una sesión utilizando como nombre de usuario y clave de acceso las variables:

UNVLadmin

UNVLadminpass

Estas variables están definidas en el archivo config/unvconfig.txt. El resultado debe ser la pantalla de navegación para el administrador que se muestra en la figura 7.4. Seleccionar la única opción del menú de navegación (unvl), agregar una sección hija de nombre “Pruebas” y seleccionarla en el menú.

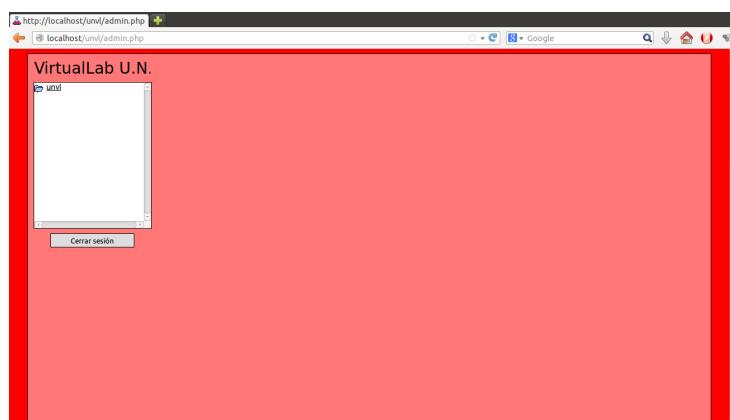


Figura 7.4 Pantalla de UNVirtualLab para el usuario administrador con sesión abierta, inmediatamente después de una instalación fresca

**IN 4.4 Importar el modelo de prueba de un Motor DC:**

```
cd DIRUNVL/samples
php importModel.php motorDC/motorDCExport.txt
```

**IN 4.5 En la página de edición del modelo “Motor DC”, hacer click sobre el ícono que está a la derecha del letrero “Documentación (.pdf)”, seleccionar el archivo DIRUNVL/samples/motorDC/MotorDC.pdf y enviarlo.**

**IN 4.6 Pruebas de simulación**

**Para una instalación mínima:**

**IN 4.6.1 En la página de edición del modelo “Motor DC”, hacer click sobre el ícono que está a la derecha del letrero “Ejecutable (.tar.gz)”, seleccionar y enviar el archivo: DIRUNVL/samples/Min/DCmotorMinimal.tar.gz**

**IN 4.6.2 En la página de edición del modelo ‘Motor DC’, hacer click sobre el ícono que está a la derecha del letrero “Modelica (.tar.gz)”, seleccionar y enviar el archivo DIRUNVL/samples/Min/DCmotorMinimalSource.tar.gz**

**Para una instalación con compilación:**

**IN 4.6.1 En la página de edición del modelo “Motor DC”, hacer click sobre el ícono que está a la derecha del letrero “Ejecutable (.tar.gz)”, seleccionar y enviar el archivo DIRUNVL/samples/Compila/DCmotorCompila.tar.gz**

**IN 4.6.2 En la página de edición del modelo “Motor DC”, hacer click sobre el ícono que está a la derecha del letrero “Modelica (.tar.gz)”, seleccionar y enviar el archivo DIRUNVL/samples/Compila/DCmotorCompilaSource.tar.gz**

**Para una instalación para modelador:**

**IN 4.6.1 Utilizando OMShell, compilar el modelo del archivo DIRUNVL/samples/Modelador/DCMotor.mo y simular el modelo con salida en formato CSV mediante las siguientes instrucciones:**

```
loadModel(Modelica)
cd DIRUNVL/samples/Modelador
loadFile("DCmotor.mo");
```

```
simulate(DCmotor, startTime=0, stopTime=10,
          outputFormat="csv");
```

IN 4.6.2 Crear un archivo comprimido en formato .tar.gz con los archivos:

- DCmotor
- DCmotor\_init.xml
- DCmotor\_res.csv

Estos archivos son creados por OMShell como parte del proceso de simulación. Para efectos de esta explicación, se supondrá que el nombre dado al archivo comprimido es DCmotorModelador.tar.gz.

IN 4.6.3 Crear un archivo comprimido en formato .tar.gz con el archivo DCmotor.mo. Para efectos de esta explicación, se supondrá que el nombre dado al archivo comprimido es: DCmotorModeladorSource.tar.gz.

IN 4.6.4 En la página de edición del modelo ‘Motor DC’, hacer click sobre el ícono que está a la derecha del letrero ‘Ejecutable (.tar.gz)’, seleccionar y enviar el archivo DIRUNVL/samples/Modelador/DCmotorModelador.tar.gz.

IN 4.6.5 En la página de edición del modelo ‘Motor DC’, hacer click sobre el ícono que está a la derecha del letrero ‘Modelica (.tar.gz)’, seleccionar y enviar el archivo DIRUNVL/samples/Modelador/DCmotorModeladorSource.tar.gz.

IN 4.7 Utilizando un programa navegador web, cargar la dirección definida por la variable URLbase (archivo config/unvlconfig.txt) y seleccionar el modelo “Motor DC”. El resultado debe ser la pantalla de navegación para el experimentador que se muestra en la figura 7.5.

IN 4.8 Modificar el valor del parámetro “Resistencia” con la interfaz, y lanzar una simulación.

Tabla 7.1 Variables de configuración de UNVirtualLab en el archivo config/unvlconfig.txt

Variable	Significado	Valor por defecto
<code>locale</code>	Idioma de la interfaz. Es el nombre del archivo .inc con los mensajes que se despliegan. El archivo debe estar ubicado en el directorio <code>locale</code> .	es
<code>theme</code>	Configuración de la apariencia. Es el nombre del subdirectorio que contiene las especificaciones CSS. El subdirectorio debe estar ubicado dentro del directorio <code>themes</code>	unvlbasic
<code>plotWidth</code>	Ancho en pixeles de las figuras que muestran los resultados de las simulaciones.	300
<code>plotHeight</code>	Altura en pixeles de las figuras que muestran los resultados de las simulaciones.	200
<code>aboutWidth</code>	Ancho en pixeles de la ventana que despliega información de referencia de los modelos.	400
<code>aboutHeight</code>	Altura en pixeles de la ventana que despliega información de referencia de los modelos.	400
<code>DBserver</code>	Dirección del servidor de la base de datos <code>mysql</code> .	localhost
<code>DBname</code>	Nombre de la base de datos.	unvl
<code>DBuser</code>	Nombre del usuario <code>mysql</code> con acceso a lectura de registros en la base de datos.	unvlvisitor
<code>DBuserpass</code>	Clave de acceso del usuario <code>mysql</code> con acceso a lectura de registros en la base de datos.	unvlvisitor
<code>DBadmin</code>	Nombre del usuario <code>mysql</code> con acceso a edición, inserción y eliminación de registros en la base de datos.	unvladmin
<code>DBadminpass</code>	Clave de acceso del usuario <code>mysql</code> con acceso a edición, inserción y eliminación de registros en la base de datos.	unvladmin
<code>UNVLadmin</code>	Nombre del usuario UNVirtualLab con rol de administrador.	adminUNVL
<code>UNVLadminpass</code>	Clave de acceso del usuario UNVirtualLab con rol de administrador.	adminpassword
<code>URLbase</code>	Dirección URL de acceso a UNVirtualLab.	localhost/unvl
<code>unvlDir</code>	Directorio en que está alojado UNVirtualLab.	/var/www/unvl
<code>Compilation</code>	Si esta variable toma el valor TRUE, entonces los archivos que se suban como ejecutables serán descomprimidos, compilados y copiados en el directorio del modelo. De lo contrario, solo serán descomprimidos y copiados.	TRUE
<code>CompilationOrder</code>	Instrucción de compilación que se utiliza si la variable <code>Compilation</code> toma el valor TRUE.	omc
<code>CompilationPostOrder</code>	Complemento a la instrucción de compilación.	>/dev/null

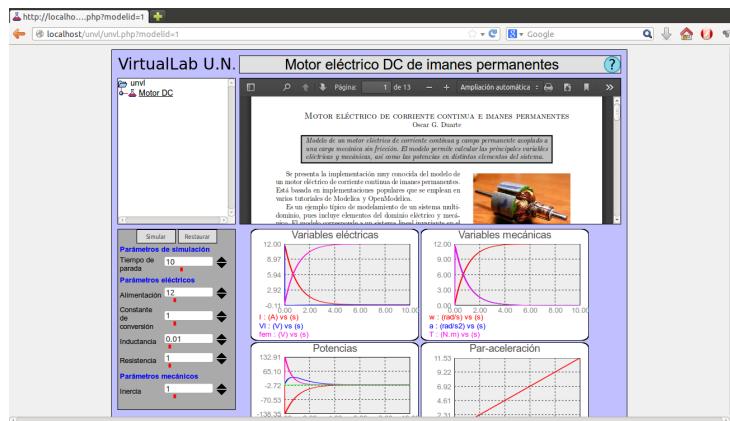


Figura 7.5 Pantalla de UNVirtualLab para el usuario experimentador, después de la instalación del ejemplo de Motor DC

### 7.1.2 Copias de seguridad

Una copia de seguridad (BK) de UNVirtualLab incluye dos aspectos: un volcado de la base de datos y una copia de los archivos:

- BK 1. El volcado de la base de datos puede efectuarse desde consola mediante la utilidad mysqldump:

```
mysqldump -u DBadmin -pDBadminpass --add-drop-table DBname > dumpfile
```

En donde DBadmin, DBadminpass y DBname están definidos en el archivo config/unvlconfig.txt, y dumpfile es el nombre del archivo que contendrá el volcado de la base de datos.

- BK 2. La copia de los archivos puede realizarse desde consola así:

```
cd DIRUNVL
tar -cf backUnvl.
gzip backUnvl.tar
```

en donde DIRUNVL es el directorio de instalación, y backUnvl es la base del nombre del archivo de backup. El archivo generado tendrá por nombre backUnvl.tar.gz.

### 7.1.3 Exportación e importación de modelos

UNVirtualLab cuenta con dos *scripts* para exportar (EX) e importar (IM) la información que reside en la base de datos de un modelo específico. El proceso de exportación es el siguiente:

EX 1. Desde la ventana de administración del modelo, averiguar cuál es el identificador del modelo. Este es el número que aparece en la tabla de información general del modelo en la casilla “Identificador”.

EX 2. Desde consola, exportar el modelo con la siguiente secuencia de instrucciones (modID es el identificador del modelo, y filename un nombre arbitrario para el modelo):

```
cd DIRUNVL/samples
php exportModel.php modID > filename
```

EX 3. Crear el archivo comprimido del ejecutable. Los archivos que se deben comprimir varían según el tipo de instalación de destino (mínima, con compilación o para modelador. Véase sección 7.2.4):

DIRUNVL/files/modID/bin/

EX 4. Obtener una copiar del archivo de documentación:

DIRUNVL/files/modID/doc/documentation.pdf

EX 5. Obtener una copiar del archivo del código fuente:

DIRUNVL/files/modID/modelica/source.tar.gz

EX 6. Obtener una copiar de los archivos SVG de animación que estén en la carpeta:

DIRUNVL/files/modID/graphics

Para importar el modelo en otra instalación de UNVirtualLab, el procedimiento es el siguiente:

IM.1. Desde la consola, importar el modelo con la siguiente secuencia de instrucciones (filename es el nombre del archivo en el que se ha exportado el modelo):

```
cd DIRUNVL/samples
php importModel.php filename
```

IM.2. Desde la ventana de administración de modelos, cargar los archivos ejecutables, fuente y de documentación.

IM.3. Desde la ventana de administración de modelos, cargar los archivos SVG para animaciones.

## 7.2 MANUAL DEL MODELADOR

El modelador tiene una responsabilidad múltiple en la producción del material alojado en UNVirtualLab. Los tres productos que debe elaborar son:

- El modelo en Modelica que implementa la simulación.
- Las plantas de experimentación en UNVirtualLab que permiten la interacción del usuario experimentador con el modelo.
- La documentación en formato PDF, tanto del modelo como de las plantas de experimentación, destinada al usuario experimentador.
- Un listado de experimentos sugeridos para ser desarrollados por el usuario experimentador.

La figura 7.6 ilustra el proceso sugerido al modelador: la producción del modelo, de las plantas experimentales y de los experimentos se realiza en ciclos iterativos de mejoramiento; en cada etapa se actualiza la documentación correspondiente.

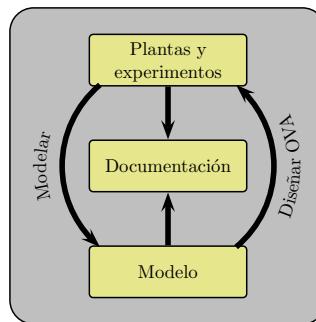


Figura 7.6 Proceso iterativo de modelado

En cada uno de los ciclos de modelado esbozados en la figura 7.6, el modelador debe realizar varias actividades que generan ciertos productos, tal como se ilustra en la figura 7.7. Estos productos se explican a continuación. El proceso puede iniciarse desde cualquiera de las entradas marcadas por líneas punteadas; a efectos de la siguiente explicación, se supondrá que se inicia desde la definición de objetivos del OVA. Para facilitar la comprensión, dicha explicación se acompaña de un mismo ejemplo que evoluciona a cada paso.

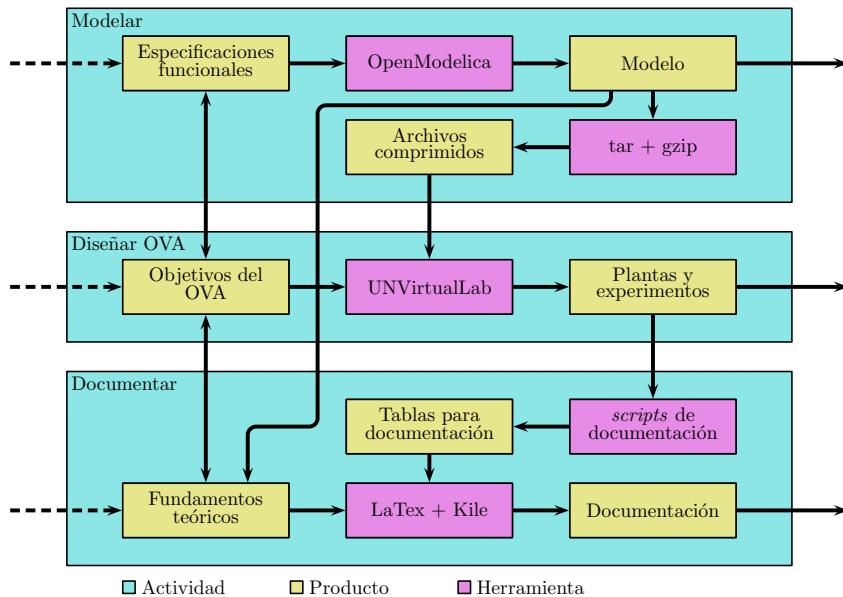


Figura 7.7 Un ciclo de modelado

### 7.2.1 Objetivos del OVA

Independientemente del uso final que tenga una instalación específica de UNVirtualLab, cada uno de los experimentos y cada una de las prácticas en esa instalación se pueden considerar como Objetos Virtuales de Aprendizaje (véase [Dua09]). El modelador debe especificar cuál es el propósito para el que se diseña dicho OVA.

#### Ejemplo 7.1: Objetivos de aprendizaje.

Se desea ilustrar mediante una simulación el funcionamiento de un motor eléctrico DC. Específicamente, se formulan los siguientes objetivos de aprendizaje:

- Reconocer el impacto de la carga mecánica en las variables eléctricas.
- Identificar el efecto de los parámetros eléctricos y mecánicos en los tiempos de respuesta.
- Identificar los procesos de transformación de potencia y energía.



### 7.2.2 Especificaciones funcionales

El modelador debe definir lo que debe hacer el modelo. Esta definición incluye la identificación de las principales variables y parámetros involucrados.

#### *Ejemplo 7.2: Especificaciones funcionales.*

---

El modelo debe simular el comportamiento de un motor DC acoplado a una carga mecánica. Específicamente debe simularse el comportamiento de las siguientes variables físicas:

- Diferencia de potencial eléctrico en cada uno de los elementos eléctricos del sistema.
- Corriente eléctrica en cada uno de los elementos eléctricos del sistema.
- Torque en cada uno de los elementos mecánicos del sistema.
- Velocidad angular en cada uno de los elementos mecánicos del sistema.
- Potencia y energía en cada uno de los elementos del sistema.

El modelo debe incorporar los siguientes parámetros físicos:

- Resistencia eléctrica e inductancia eléctrica del motor.
- Factor de conversión electromagnética.
- Momento de inercia de la carga mecánica.

El modelo debe incorporar los siguientes parámetros de simulación:

- Tiempo de simulación.



### 7.2.3 Modelo

Utilizando OpenModelica, el modelador debe implementar la simulación en un modelo ejecutable. Debe tenerse especial atención en la definición de las unidades de las variables para que estas sean adecuadamente desplegadas por UNVirtualLab. El módulo SIunits de la Modelica Standard Library incluye una gran colección de unidades listas para su incorporación en los modelos. Si el modelador requiere una definición propia de sus unidades en una variable específica, puede usar la propiedad unit del tipo de datos Real. Por ejemplo, si el usuario requiere definir la variable t para medir el tiempo en años, puede emplear el siguiente código

```
Real t (unit="Year") "Tiempo";
```

Una vez simulado el modelo, es necesario ubicar los archivos ejecutable, de entrada y de salida generados por OpenModelica.

### *Ejemplo 7.3: Modelado.*

---

El modelo propuesto es muy similar al tutorial que incluye OpenModelica. Los pasos a seguir son:

1. Crear el modelo DCmotor a partir del código que se muestra en el cuadro correspondiente al archivo DCmotor.mo
2. Simular el modelo, empleando como formato de salida csv. Por ejemplo, utilizando la herramienta OMShell de OpenModelica la secuencia de instrucciones será (debe sustituirse path por el directorio en el que se encuentra el archivo DCmotor.mo):

```
loadModel(Modelica);
cd("path")
loadFile("DCmotor.mo");
simulate(DCmotor,outputFormat="csv",stopTime=10);
plot(Pj);
```

3. Copiar en alguna carpeta los siguientes archivos generados por OpenModelica<sup>5</sup>:
  - DCmotor (o DCmotor.exe si el sistema operativo es windows)
  - DCmotor\_init.xml
  - DCmotor\_res.csv




---

<sup>5</sup>En una instalación estándar de UNVirtualLab sobre linux, los archivos generados por OMEedit se ubican en la carpeta temporal /tmp/OpenModelica/OMEdit; los archivos generados por OMShell se ubican en la misma carpeta del archivo fuente Modelica.

## Archivo 7.1 DCmotor.mo

```

model DCmotor
  Modelica.Electrical.Analog.Basic.Resistor R(R = 1);
  Modelica.Electrical.Analog.Basic.Inductor L(L = 0.01);
  Modelica.Electrical.Analog.Basic.EMF emf(k = 1);
  Modelica.Electrical.Analog.Basic.Ground g;
  Modelica.Blocks.Sources.Step step(height = 12);
  Modelica.Mechanics.Rotational.Components.Inertia J(J = 1);
  Modelica.Electrical.Analog.Sources.SignalVoltage SV;
  Modelica.SIunits.Power Psv "Potencia en la fuente";
  Modelica.SIunits.Power Pr "Potencia en la resistencia";
  Modelica.SIunits.Power Pl "Potencia en la inductancia";
  Modelica.SIunits.Power Pj "Potencia en la carga mecánica";
  Modelica.SIunits.Power PT "Potencia total";
  Modelica.SIunits.AngularAcceleration alfa "Aceleración angular";
equation
  connect(L.n,emf.p);
  connect(R.n,L.p);
  connect(SV.p,R.p);
  connect(SV.n,g.p);
  connect(emf.n,g.p);
  connect(step.y,SV.v);
  connect(emf.flange,J.flange_b);
  Psv = SV.v * SV.i;
  Pr = R.v * R.i;
  Pl = L.v * L.i;
  Pj = J.flange_b.tau * J.w;
  PT=Psv+Pr+Pl+Pj+PT;
  alfa=der(J.w);
end DCmotor;

```

## 7.2.4 Archivos comprimidos

UNVirtualLab requiere dos archivos comprimidos para cada experimento. En ambos casos, el formato de compresión requerido es .tar.gz. Los archivos requeridos son:

- Archivo comprimido del código fuente: contiene todos los archivos con código fuente del modelo. Este archivo sólo se utiliza como fuente de información para el usuario, en caso de que desee simular por su cuenta el modelo, o realizarle modificaciones. Por tanto, la forma en que se organice el contenido del archivo comprimido no es crítica para el funcionamiento de UNVirtualLab.
- Archivo comprimido del ejecutable: contiene todos los archivos necesarios para que UNVirtualLab corra la simulación. El contenido que debe tener depende del tipo de instalación de UNVirtualLab, así:

- Instalación mínima: el archivo comprimido debe contener tres archivos:
  - a. El archivo ejecutable del modelo.
  - b. El archivo de entrada, cuyo nombre termina en `_init.xml`.
  - c. El archivo de salida, cuyo nombre termina en `_res.csv`.
- Instalación con compilación: el archivo comprimido debe contener:
  - a. Un archivo con extensión `.mos` con la secuencia de instrucciones OpenModelica para compilar y simular el modelo. Debe haber sólo un archivo con extensión `.mos`.
  - b. Todos los archivos con código fuente necesarios para compilar el modelo.
- Instalación para modelador: el contenido debe ser igual al de una instalación con compilación.

*Ejemplo 7.4: Preparación de archivos.*

---

Crear un archivo de texto de nombre `DCmotorUNVL.mos` con el siguiente contenido:

```
loadModel(Modelica);
loadFile("DCmotor.mo");
simulate(DCmotor, startTime=0, stopTime=10, outputFormat="csv");
```

Crear un archivo comprimido en formato `.tar.gz` con los archivos:

- `DCmotor.mo`
- `DCmotorUNVL.mos`

Para efectos de este ejemplo, se supondrá que el nombre dado al archivo comprimido es `DCmotorSource.tar.gz`

Crear un archivo comprimido en formato `.tar.gz` con el archivo `DCmotor.mo`. Para efectos de este ejemplo, se supondrá que el nombre dado al archivo comprimido es: `DCmotorSource.tar.gz`



### 7.2.5 Plantas de experimentación

En el contexto de UNVirtualLab, una planta de experimentación está asociada a un único modelo ejecutable. De manera análoga a las plantas físicas de experimentación, las plantas en UNVirtualLab están dotadas de instrumentos que permiten controlarlas y visualizar sus comportamientos. Es responsabilidad del modelador el diseño de tales instrumentos.

La figura 7.8 muestra en un diagrama el tipo de instrumentos asociados a una planta de experimentación UNVirtualLab. Hay una primera clasificación en instrumentos de control y de visualización. Los primeros permiten modificar las condiciones de la simulación, mientras los segundos permiten visualizar los resultados de esta.

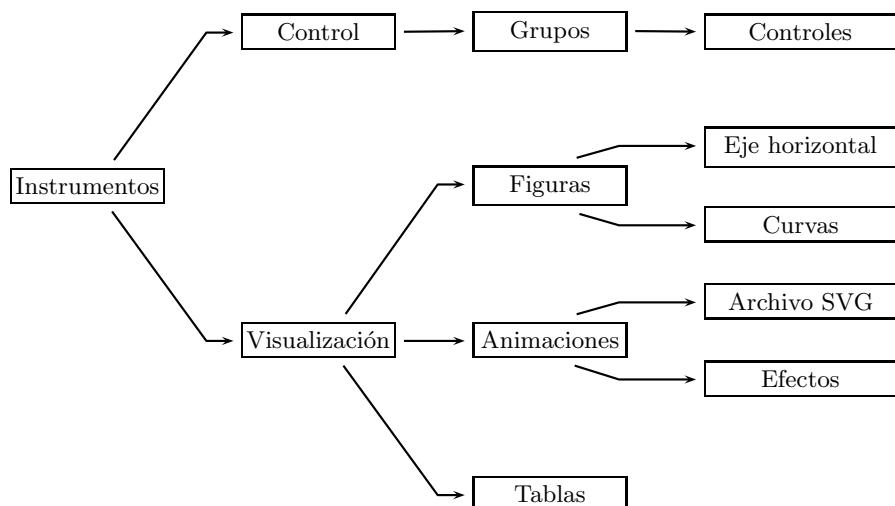


Figura 7.8 Instrumentos asociados a un experimento

El modelador debe decidir el número de controles, así como un conjunto de propiedades asociadas a cada uno de ellos. La tabla 7.2 muestra las propiedades asociadas a cada tipo de control con una descripción de su significado.

Tabla 7.2 Propiedades de los controles de cada experimento

Grupos de Controles	
Propiedad	Explicación
Nombre	Nombre asignado al grupo de controles.
Descripción	Descripción que se visualiza al llevar el cursor al nombre del grupo de controles.

Habilitado	Variable que permite mostrar u ocultar al experimentador el grupo de controles.
<b>Controles</b>	
Propiedad	Explicación
Nombre	Nombre asignado al control.
Valor	Valor por defecto que toma la variable asociada al control.
Habilitado	Variable que permite mostrar u ocultar al experimentador el control.
Mínimo	Valor mínimo permitido para el parámetro asociado al control.
Máximo	Valor máximo permitido para el parámetro asociado al control.
Incremento	Valor en que se incrementa/descuenta el parámetro asociado al control al momento de pulsar los botones de incremento/- decremento
Valores permitidos	Listado de valores y etiquetas para utilizar controles de tipos <b>select</b> . Cada opción debe terminar en el carácter ';' y contener una pareja 'valor' - 'etiqueta' separados por el carácter '/'. El valor es el dato que leerá el modelo, y la etiqueta es lo que se visualizará en pantalla. Por ejemplo: "1.0/uno;2.0/dos;3.0/tres". Para que se active el control tipo <b>select</b> es necesario que haya al menos una opción válida.
Descripción	Descripción que se visualiza al llevar el cursor al nombre del grupo de controles
Parámetro	Parámetro del modelo asociado al control.
<b>Figuras</b>	
Propiedad	Explicación
Nombre	Nombre asignado a la figura y título de la misma
Descripción	Descripción que se visualiza al llevar el cursor sobre la figura.
Habilitado	Variable que permite mostrar u ocultar al experimentador la figura.
Variable horizontal	Variable del modelo asociada al eje horizontal de la figura.
Valor mínimo de la variable horizontal	Valor mínimo en la escala horizontal (la opción de autoescala horizontal debe estar deshabilitada).
Valor máximo de la variable horizontal	Valor máximo en la escala horizontal (la opción de autoescala horizontal debe estar deshabilitada).
Divisiones horizontales	Número de divisiones horizontales para la grilla de la figura.
Auto-escala horizontal	Si esta opción se habilita, entonces la escala horizontal toma un rango calculado a partir de los resultados de la simulación.
Valor mínimo de la variable vertical	Valor mínimo en la escala vertical (la opción de autoescala vertical debe estar deshabilitada).
Valor máximo de la variable vertical	Valor máximo en la escala vertical (la opción de autoescala vertical debe estar deshabilitada).
Divisiones verticales	Número de divisiones verticales para la grilla de la figura.
Auto-escala vertical	Si esta opción se habilita, entonces la escala vertical toma un rango calculado a partir de los resultados de la simulación.
Primer dato	Número de datos de la simulación que serán ignorados para el trazado de las curvas.
<b>Curvas</b>	
Propiedad	Explicación
Nombre	Nombre asignado a la figura. No se visualiza para el experimentador.
Leyenda	Nombre corto que aparecerá como leyenda de la curva en la figura.

Habilitado	Variable que permite mostrar u ocultar al experimentador la curva.
Descripción	Descripción que se visualiza al llevar el cursor al nombre de la curva en la leyenda.
Variable vertical	Variable del modelo asociada al eje vertical de la figura para la curva.
Color	Color con el que se trazará la curva y con el que se escribirá su leyenda.
<b>Animaciones</b>	
<b>Propiedad</b>	<b>Explicación</b>
Nombre	Nombre asignado a la animación y título de esta.
Descripción	Descripción que se visualiza al llevar el cursor sobre la animación.
Habilitado	Variable que permite mostrar u ocultar al experimentador la animación.
Duración	Tiempo de duración de la animación. Si toma un valor no positivo, la duración será igual al tiempo de simulación.
Tiempo de muestreo	Tiempo de muestreo para generar la animación.
Archivo SVG	Nombre del archivo SVG de la simulación.
<b>Efecto</b>	
<b>Propiedad</b>	<b>Explicación</b>
Nombre	Nombre del efecto. No se despliega.
Descripción	Descripción del efecto. No se despliega.
Habilitado	Variable que permite mostrar u ocultar al experimentador el efecto.
Tipo	Tipo de efecto de animación. 'single', una variable controla el efecto; 'double', dos variables controlan el efecto; 'path', trayectoria en 2D en la que dos variables controlan las coordenadas x,y respectivamente; 'text', cambio de un texto (no implementada); 'color', un color es controlado por una variable.
Identificador de animación SVG	id interno del efecto del archivo SVG que será controlado.
Variable de control	Primera variable de control del efecto.
Variable auxiliar	Segunda variable de control para los efectos tipo 'double' y 'path'.
Offset	Valor de offset para control del efecto.
Escala	Valor de amplificación de la variable simulada para controlar el efecto.
Color RGB mínimo	Color correspondiente al valor mínimo de la variable. Válido para efectos tipo 'color'.
Color RGB máximo	Color correspondiente al valor máximo de la variable. Válido para efectos tipo 'color'.
Valor mínimo de color	Valor de la variable asociado a la variable 'Color RGB mínimo'. Válido para efectos tipo 'color'.
Valor máximo de color	Valor de la variable asociado a la variable 'Color RGB máximo'. Válido para efectos tipo 'color'.

### **Ejemplo 7.5: Planta de experimentación.**

---

Para satisfacer los objetivos del OVA, se ha diseñado una planta de experimentación con varios instrumentos. Las propiedades específicas de cada instrumento se muestran en la tabla 7.3, mientras que el diseño de las animaciones se explica en la sección 7.2.6. Los instrumentos implementados son los siguientes:

- a. Instrumentos de control: tres grupos de controles con un total de cinco controles:
  - Parámetros mecánicos: inercia de la carga mecánica.
  - Parámetros eléctricos: resistencia del motor, inductancia del motor, constante de conversión electromagnética.
  - Parámetros de simulación: tiempo de simulación.
- b. Instrumentos de visualización: tres figuras y una animación. En las figuras se grafican un total de 10 curvas, y en la animación se visualiza el comportamiento de dos variables:
  - Figuras:
    - Variables eléctricas: se grafica el comportamiento de la corriente, la tensión en la inductancia y la tensión electromagnética.
    - Variables mecánicas: se grafica el comportamiento de la velocidad angular, la aceleración angular y el par mecánico.
    - Potencias: se grafica el comportamiento de la potencia en la fuente eléctrica, en la carga mecánica, en la resistencia y en la inductancia.
  - Animación:
    - Mediante un sensor de aguja se visualiza el comportamiento de la velocidad angular.
    - Mediante un sensor de barra se visualiza el comportamiento de la potencia mecánica.



Tabla 7.3 Propiedades de los controles en el ejemplo

Grupos de Controles						
Nombre	Parámetros mecánicos	Parámetros eléctricos	Parámetros de simulación			
Descripción	Parámetros mecánicos del modelo	Parámetros eléctricos del modelo	Parámetros de las condiciones de simulación			
Habilitado	SI	SI	SI			
Nombre	Inercia	Resistencia	Inductancia	Constante de conversión	Alimentación	Tiempo de simulación
Valor	1	1	0.01	1	12	10
Habilitado	SI	SI	SI	SI	SI	SI
Mínimo	0	0	0	0	0	0
Máximo	5	10	0.1	5	60	30
Incremento	0.5	0.5	0.01	0.5	5	1
Valores permitidos	-	-	-	-	-	-
Descripción	Momento de inercia de la carga mecánica ( $kg \cdot m^2$ )	Resistencia eléctrica del motor (Ohm)	Inductancia del motor (H)	Constante de conversión electromagnética (N.m/A)	Tensión de alimentación (V)	Tiempo de parada de la simulación (s)
Parámetro	IJ	R.R	L.L	emf/k	stepheight	stopTime
Figuras						
Nombre	Variables eléctricas	Variables mecánicas	Variables mecánicas del modelo	Variables mecánicas	Potencias	Par-aceleración
Descripción	Variables eléctricas del modelo	Variables mecánicas	Variables mecánicas del modelo	Variables mecánicas	Potencias en distintos elementos del sistema	Curva para las aceleraciones en la carga mecánica
Habilitado	SI	SI	SI	SI	SI	SI
Variable horizontal	time	time	time	time	time	alfa
Valor mínimo de la variable horizontal	0	0	0	0	0	0
Valor máximo de la variable horizontal	0	0	0	0	0	0
Divisiónes horizontales	5	5	5	5	5	5
Auto-escala horizontal	SI	SI	SI	SI	SI	SI
Valor mínimo de la variable vertical	0	0	0	0	0	0
Valor máximo de la variable vertical	0	0	0	0	0	0
Divisiónes verticales	4	4	4	4	4	5

Auto-escala vertical	SI		SI		SI		SI
Primer dato	0		0		0		0
<b>Curvas</b>							
Nombre	Corriente	Tensión en inducción	Fem	Velocidad angular	Par	Potencia en resistencia	Potencia en inducción
Levenda	I	Vl	fem	w	a	Pr	Par mecánico
Habilitado	SI	SI	SI	SI	SI	Pr	T vs alfa
Descripción	Corriente en el motor	Tensión en la inductancia del motor	Fuerza electromotriz	Velocidad angular	Par mecánico	SI	SI
Variable vertical	R.i	L.v	enm.v	J.w	alfa	SI	SI
Color	FF0000	0000FF	FF00FF	FF0000	Jflange_btatu	SI	SI
Animaciones							
Nombre	Sensores						
Descripción	Animación con sensores de variables del modelo						
Habilitado	SI						
Duración	-						
Tiempo de muestreo	-						
Archivo SVG	sensores.svg						
Efecto							
Nombre	Aguja_w	Aguja_w	Barra_Pm				
Descripción	Sensor de velocidad angular	Sensor de velocidad angular	Sensor de potencia mecánica				
Habilitado	SI	SI	SI				
Tipo	single	single	barra				
Identificador de animación SVG	aguja	aguja	barra				
Variable de Control	J.w		Pj				
Variable Auxiliar	-	-	-				
Offset	90	0					
Escala	10	1					
Color RGB mínimo	-	-					
Color RGB máximo	-	-					
Valor mínimo de Color	-	-					
Valor máximo de Color	-	-					

### 7.2.6 Animaciones

Para implementar las animaciones en 2D, UNVirtualLab aprovecha el potencial de animación (AN) de los archivos gráficos en formato SVG<sup>6</sup>. Para implementar una animación en UNVirtualLab, el modelador debe seguir los siguientes pasos:

- AN 1. Construir un gráfico en formato SVG. Debido a que el formato SVG es una extensión del formato XML, el archivo puede construirse desde un editor de texto. No obstante, existen herramientas gráficas de dibujo que facilitan esta tarea. Dentro de las herramientas de *software libre*, se destaca *inkscape* (véase [dI31]). Otra herramienta útil de *software libre* es *scour* (ver [Sch31]), que simplifica el contenido de archivos SVG creados con herramientas gráficas.
- AN 2. Identificar el elemento gráfico que se desea animar. Si son varios elementos los que se desean animar con una misma variable de control, conviene agruparlos como un único elemento. Para ello puede emplearse la marca `<g></g>` del lenguaje SVG.
- AN 3. Construir el efecto de animación del elemento con datos ficticios y asignar a esa animación un identificador. A manera de ejemplo, se supondrá aquí que el identificador asignado es `id_efecto`. En la sección 7.2.6.1 se muestra como construir algunos efectos de animación.
- AN 4. En la página de administración del modelo, crear una nueva animación 2D.
- AN 5. En la página de administración de la animación, cargar el archivo SVG.
- AN 6. En la página de administración de la animación, crear un nuevo efecto.
- AN 7. En la página de administración del efecto, editar la propiedad *Identificador de animación SVG* para asignar el identificador del efecto de animación (`id_efecto`).
- AN 8. En la página de administración del efecto, ajustar las demás propiedades hasta tener el efecto de animación deseado. En especial, es necesario ajustar los valores de *Offset* y *Escala* para traducir los resultados

---

<sup>6</sup>Scalable Vector Graphics. Para una introducción al tema pueden consultarse los numerosos tutoriales disponibles en la red, por ejemplo [DHH02]. Las animaciones SVG son implementadas por los navegadores web; por esta razón, para poder visualizar las animaciones de UNVirtualLab es necesario emplear un navegador web que soporte las animaciones SVG. Hoy en día, esto no constituye una limitación importante, debido a que incluso la mayoría de los navegadores para dispositivos móviles lo hacen.

de la simulación en valores gráficos de la animación. Los valores gráficos de la animación serán pixeles o grados, según el tipo de animación. Si se denota por  $X_{res}$  los resultados de la simulación, y por  $X_{ani}$  los valores gráficos de la animación, estos se calculan así:

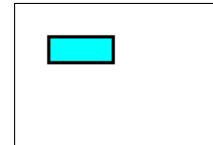
$$X_{ani} = \text{Offset} + (\text{Escala} \times X_{res}) \quad (7.1)$$

#### 7.2.6.1 Algunos efectos de animación SVG

La tabla 7.4 muestra algunos efectos de animación con gráficos SVG. A partir de un rectángulo base, se presentan los siguientes efectos:

- a. Desplazamientos: pueden realizarse desplazamientos en cada uno de los dos ejes, o de forma combinada a lo largo de una trayectoria.
- b. Rotaciones: la animación de rotación por defecto es alrededor del punto  $(0, 0)$ . El efecto de rotar alrededor de un punto  $p$  se puede conseguir en tres pasos: 1) trasladando el elemento al punto  $(0, 0)$ ; 2) efectuando la rotación deseada  $y$ ; 3) trasladando el objeto rotado al punto  $p$ .
- c. Cambio de atributos: pueden modificarse el largo y ancho del rectángulo, así como su color y otras propiedades gráficas.
- d. Animación de textos: el estándar SVG no prevee una animación de textos. Sin embargo, puede emplearse la animación del atributo ‘opacity’ (‘opacidad’) para conseguir este efecto. UNVirtualLab 2.0 no implementa este efecto, pero se presenta aquí a título informativo.

Tabla 7.4 Ejemplo de implementación de algunos tipos de animación SVG

Rectángulo base del ejemplo	Imagen base:
<pre>&lt;?xml version="1.0" encoding="UTF-8" standalone="no"?&gt; &lt;svg xmlns="http://www.w3.org/2000/svg" width="300" height="200" version="1.1"&gt;   &lt;rect x="40" y="40" width="100" height="40"         stroke="black" stroke-width="5" fill="cyan"&gt;   &lt;/rect&gt; &lt;/svg&gt;</pre>	<p>Imagen base:</p> 

**Desplazamiento horizontal**

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<svg xmlns="http://www.w3.org/2000/svg" width="300" height="200" version="1.1">
<rect x="40" y="40" width="100" height="40"
      stroke="black" stroke-width="5" fill="cyan">
<animate
  attributeName="x"
  begin="0"
  dur="2"
  values="20.0; 30.0; 80.0; 100.0"
  keyTimes="0.0; 0.25; 0.50; 1.0"
  repeatCount="1"
  fill="freeze">
/>
</rect>
</svg>
```

Imagen base:

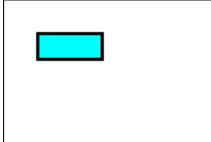
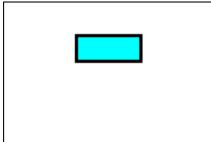


Imagen final:

**Desplazamiento vertical**

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<svg xmlns="http://www.w3.org/2000/svg" width="300" height="200" version="1.1">
<rect x="40" y="40" width="100" height="40"
      stroke="black" stroke-width="5" fill="cyan">
<animate
  attributeName="y"
  begin="0"
  dur="2"
  values="20.0; 30.0; 80.0; 100.0"
  keyTimes="0.0; 0.25; 0.50; 1.0"
  repeatCount="1"
  fill="freeze">
/>
</rect>
</svg>
```

Imagen base:

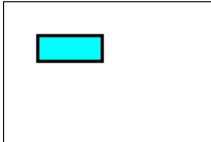
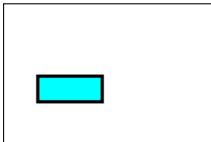


Imagen final:



### Desplazamiento a lo largo de una trayectoria

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<svg xmlns="http://www.w3.org/2000/svg" width="300" height="200" version="1.1">
<rect x="-50" y="-20" width="100" height="40"
      stroke="black" stroke-width="5" fill="cyan">
<animateMotion
  begin="0s"
  dur="2s"
  values="0,0;60,60;100,60;140,80"
  keyTimes="0.0;0.33;0.66;1.0"
  rotate="auto"
  repeatCount="1"
  fill="freeze"
/>
</rect>
</svg>
```

Imagen base:

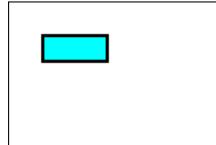
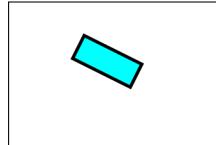


Imagen final:



### Cambio de la altura

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<svg xmlns="http://www.w3.org/2000/svg" width="300" height="200" version="1.1">
<rect x="40" y="40" width="100" height="40"
      stroke="black" stroke-width="5" fill="cyan">
<animate
  attributeName="height"
  begin="0"
  dur="2"
  values="20.0; 30.0; 60.0; 80.0"
  keyTimes="0.0; 0.25; 0.50; 1.0"
  repeatCount="1"
  fill="freeze"
/>
</rect>
</svg>
```

Imagen base:

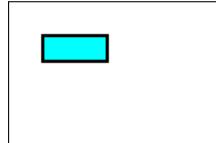
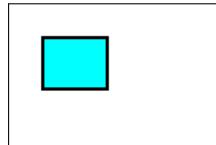


Imagen final:



**Cambio del ancho**

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<svg xmlns="http://www.w3.org/2000/svg" width="300" height="200" version="1.1">
<rect x="40" y="40" width="100" height="40"
      stroke="black" stroke-width="5" fill="cyan">
<animate
    attributeName="width"
    begin="0"
    dur="2"
    values="20.0; 30.0; 80.0; 200.0"
    keyTimes="0.0; 0.25; 0.50; 1.0"
    repeatCount="1"
    fill="freeze">
/>
</rect>
</svg>
```

Imagen base:

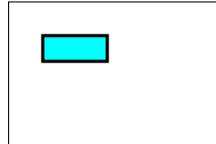


Imagen final:

**Rotación alrededor del punto (0,0)**

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<svg xmlns="http://www.w3.org/2000/svg" width="300" height="200" version="1.1">
<rect x="40" y="40" width="100" height="40"
      stroke="black" stroke-width="5" fill="cyan">
<animateTransform
    attributeName="transform"
    begin="0s"
    dur="2s"
    type="rotate"
    values="5.0; 10.0; 20.0; 25.0"
    keyTimes="0.0; 0.25; 0.50; 1.0"
    repeatCount="1"
    fill="freeze">
/>
</rect>
</svg>
```

Imagen base:

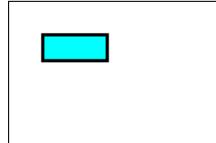
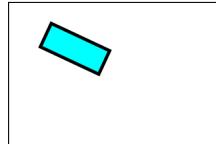


Imagen final:



### Rotación alrededor de un punto arbitrario

```

<?xml version="1.0" encoding="UTF-8" standalone
  ="no"?>
<svg xmlns="http://www.w3.org/2000/svg" width
  ="300" height="200" version="1.1">
<g>
  <rect x="-50" y="-20" width="100" height
    ="40" stroke="black" stroke-width="5"
    fill="cyan">
    <animateTransform
      attributeName="transform"
      begin="0s"
      dur="2s"
      type="rotate"
      values="5.0; 10.0; 20.0; 25.0"
      keyTimes="0.0; 0.25; 0.50; 1.0"
      repeatCount="1"
      fill="freeze"
    />
  </rect>
  <animateTransform
    attributeName="transform"
    type="translate"
    begin="0"
    dur="2s"
    values="90.0,60.0; 90.0,60.0"
    keyTimes="0.0; 1.0"
    repeatCount="1"
    fill="freeze"
  />
</g>
</svg>

```

Imagen base:

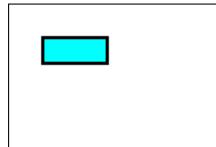
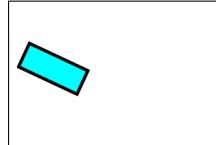


Imagen final:



**Cambio de color**

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<svg xmlns="http://www.w3.org/2000/svg" width="300" height="200" version="1.1">
<rect x="40" y="40" width="100" height="40"
      stroke="black" stroke-width="5" fill="cyan">
<animate
  attributeName="fill"
  begin="0"
  dur="2"
  values="#000000;# ff0000;#00 ff00"
  keyTimes="0;0.5;1.0"
  repeatCount="1"
  fill="freeze"
/>
</rect>
</svg>
```

Imagen base:

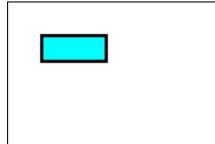
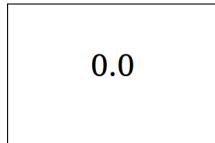


Imagen final:

**Texto base del ejemplo**

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<svg xmlns="http://www.w3.org/2000/svg" width="300" height="200" version="1.1">
<text x="150" y="100" text-anchor="middle"
      font-size="48">0.0
</text>
</svg>
```

Imagen base:



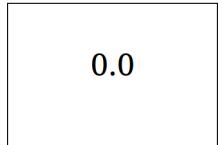
**Cambio de texto**

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<svg xmlns="http://www.w3.org/2000/svg" width="300" height="200" version="1.1">
  <text x="150" y="100" text-anchor="middle" font-size="48">0.0
    <animate attributeName="opacity" begin="0.0" dur="2.0" values="1.0;0.0;0.0;" keyTimes="0.0;0.5;1.0" repeatCount="1" fill="freeze"
      />
  </text>
  <text x="150" y="100" text-anchor="middle" font-size="48">0.1
    <animate attributeName="opacity" begin="0.0" dur="2.0" values="0.0;1.0;0.0;" keyTimes="0.0;0.5;1.0" repeatCount="1" fill="freeze"
      />
  </text>
  <text x="150" y="100" text-anchor="middle" font-size="48">0.2
    <animate attributeName="opacity" begin="0.0" dur="2.0" values="0.0;0.0;1.0;" keyTimes="0.0;0.5;1.0" repeatCount="1" fill="freeze"
      />
  </text>
</svg>

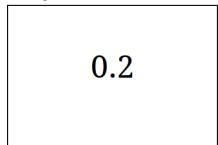
```

Imagen base:



0.0

Imagen final:



0.2

**Ejemplo 7.6: Animación 2D.**

La figura 7.9 muestra el gráfico diseñado para la animación del ejemplo. Se trata de dos sensores diferentes:

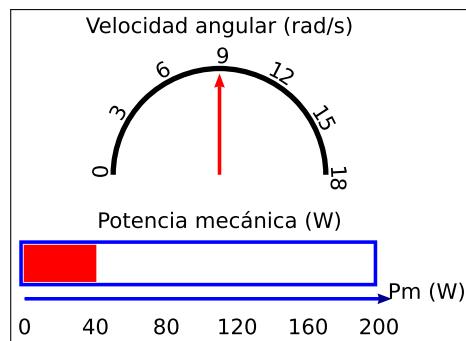


Figura 7.9 Gráfico SVG para la animación del ejemplo

- Un sensor de aguja (parte superior del gráfico) para mostrar el valor de la velocidad angular. Se trata de una línea con una marca final, que rota alrededor del punto 150,100. El ángulo de rotación se controla con el resultado de la simulación para la variable  $J.w$ , que corresponde a la velocidad angular.
- Un sensor de barra (parte inferior del gráfico) para mostrar el valor de la potencia mecánica. Se trata de un rectángulo en una posición fija, cuyo ancho se controla con el resultado de la simulación para la variable  $Pj$ , que corresponde a la potencia mecánica.

La aguja del sensor (en rojo) está en las coordenadas (0, 0).



### 7.2.7 Experimentos sugeridos

Una misma planta de experimentación puede permitir el desarrollo de muchos experimentos diferentes. El modelador puede sugerir algunos de ellos. En la construcción de las sugerencias es conveniente tener en cuenta las recomendaciones que se formulan en [Dav08], algunas de las cuales se han condensado en la sección 5.4. Especial atención debe darse al nivel de autonomía necesario para realizar el experimento según el tipo de actividad (véase tabla 5.4).

Tabla 7.5 Posibles usos y consideraciones de diseño para diferentes aproximaciones a las actividades de laboratorio (tomado de [Dav08])

<b>Demostración</b>
<ul style="list-style-type: none"> <li>▪ Mostrar algunas características de una pieza de un equipo y cómo funciona.</li> <li>▪ Mostrar un concepto particular de una teoría.</li> <li>▪ Utilizar una estrategia de mostrar y contar pero incluyendo mayor interacción con los estudiantes al formular preguntas que activen la reflexión.</li> </ul>
<b>Ejercicio</b>
<ul style="list-style-type: none"> <li>▪ Utilizar una técnica o habilidad de forma precisa.</li> <li>▪ Enfocarse en el uso adecuado de un procedimiento, más que en la investigación.</li> <li>▪ Los estudiantes deben conocer el propósito del trabajo práctico y su importancia potencial en el desempeño profesional.</li> <li>▪ Este tipo de actividad puede mejorarse mediante un contexto motivacional, tal como permitirles a los estudiantes la selección del equipo adecuado.</li> </ul>
<b>Investigación estructurada</b>
<ul style="list-style-type: none"> <li>▪ Fomentar una aproximación más profunda al aprendizaje de laboratorio, impulsando a los estudiantes a tomar iniciativas personales tales como planificar y diseñar el experimento, elegir las variables, seleccionar los métodos y materiales, etc.</li> <li>▪ A los estudiantes se les presenta un problema o una serie de preguntas de investigación que pueden basarse en la vida real, junto con sugerencias sobre los materiales de investigación y el tipo de equipo y material que deben elegir.</li> <li>▪ El rango de posibles resultados genera soluciones individuales, de tal manera que se reducen las posibilidades de plagio.</li> </ul>
<b>Investigación abierta</b>
<ul style="list-style-type: none"> <li>▪ Se tienen en cuenta las consideraciones de la investigación estructurada, pero con un número mayor de decisiones y decisiones de diseño a cargo del estudiante.</li> </ul>
<b>Proyecto</b>
<ul style="list-style-type: none"> <li>▪ Se tienen en cuenta las consideraciones de la investigación abierta, pero con una mayor cantidad de piezas de trabajo que simulen la investigación y desarrollo en el 'mundo real'.</li> </ul>

Para UNVirtualLab un experimento sugerido es un texto breve que sirve como guía al experimentador sobre posibles usos de la planta. En [Dav08] se lista un conjunto de posibles usos que cada tipo de actividad puede tener, desde la perspectiva del aprendizaje. La tabla 7.5 reproduce ese listado. UNVirtualLab permite desarrollar en línea actividades de los primeros tres tipos, mientras que para los últimos dos tipos el experimentador debe descargar el código fuente y manipularlo en el ambiente de OpenModelica.

### Ejemplo 7.7: Experimentos sugeridos.

---

La tabla 7.6 muestra seis experimentos sugeridos que se han preparado para el ejemplo, con los que se cubren todos los tipos de actividad a que se refiere la tabla 5.4. Estos experimentos buscan facilitar el aprendizaje de los objetivos formulados en la sección 7.2.1.



### 7.2.8 Documentación

La documentación asociada a cada planta experimental debe elaborarse en formato PDF. La ventana en la que se presenta tiene un tamaño que depende del diseño del tema seleccionado<sup>7</sup>. El formato en el que se despliega el archivo depende del módulo de visualización que tenga el navegador del usuario para archivos PDF.

El contenido de la documentación es muy importante como soporte al aprendizaje. Se sugiere que la documentación incluya, al menos, una explicación de los siguientes aspectos:

- a. El modelo: explicación de los aspectos teóricos asociados a la planta de experimentación, y descripción de los fenómenos a simular y las leyes que los gobiernan. En general, este aspecto se refiere a todas aquellas explicaciones que permitan ingresar al ciclo de aprendizaje de Kolb a través de la *conceptualización abstracta* (véase figura 6.1).
- b. Plantas de experimentación: descripción de la planta de experimentación que ilustre la función de cada uno de los instrumentos disponibles, tanto los de control como los de visualización. Esta explicación debe ayudar a ingresar al ciclo de aprendizaje de Kolb a través de la *experiencia concreta* o de la *experimentación activa* (véase figura 6.1).
- c. Experimentos sugeridos: explicación de los experimentos sugeridos para realizar con la planta. Esta explicación busca ayudar a ingresar al ciclo de aprendizaje de Kolb a través de la *observación reflexiva* (véase figura 6.1).

---

<sup>7</sup>En el tema `unvlabasic` el tamaño de la ventana es de 680 × 260 pixeles.

Tabla 7.6 Experimentos sugeridos para el ejemplo de la sección 7.2

Nombre	Tipo	Encabezado	Descripción
Segunda ley de Newton de la rotación	Demostración	Verifique el cumplimiento de la segunda ley de Newton de la rotación.	Analice el tipo de relación que hay entre el par mecánico y la aceleración angular. Pruebe para distintos valores del momento de inercia.
Suma de potencias	Demostración	Verifique que la suma total de potencias en el sistema es nula.	Utilice la tabla de datos para verificar en una hoja electrónica cuánto suman las potencias en los diferentes elementos del sistema. Repita el ejercicio para diferentes condiciones de simulación.
Tiempo de asentamiento de la velocidad angular	Ejercicio	¿Cuál es el tiempo de asentamiento de la velocidad angular? ¿Qué relación tiene con el momento de inercia, con la resistencia eléctrica y con la inductancia eléctrica?	Obtenga el tiempo en el que la velocidad angular llega al 95 % de su valor final. Realice el cálculo para diferentes condiciones de momento de inercia, resistencia eléctrica e inductancia eléctrica.
Velocidad angular estacionaria	Investigación estructurada	¿Qué relación hay entre el valor estacionario de la velocidad angular y la tensión de alimentación?	Encuentre el efecto que tiene el cambio en la tensión de alimentación sobre la velocidad angular estacionaria. Utilice el modelo matemático para deducir una expresión sobre esta relación y contrástela con el comportamiento del modelo.
Efectos de la carga mecánica en el circuito eléctrico	Investigación abierta	¿Cómo incide la carga mecánica sobre las variables eléctricas del motor?	Explore la relación entre la carga mecánica y las diferentes variables de tipo eléctrico que intervienen en el motor.
Simulación de dispositivos	Proyecto	Aplique el modelo para el diseño de un elemento rotacional activado por un motor DC.	Obtenga un modelo de software para simular el comportamiento de un dispositivo real que emplee un motor DC.

- d. La implementación: descripción de la forma en que fue implementado el modelo teórico en lenguaje Modelica para ser simulado. Esta descripción debe contener los detalles que permitan modificar o complementar la planta en el ambiente OpenModelica. Se busca que esta explicación ayude a ingresar al ciclo de aprendizaje de Kolb a través de la *experimentación activa* (véase figura 6.1).

### **Ejemplo 7.8: Documentación.**

---

La documentación asociada al modelo se presenta en el capítulo 8. Los aspectos teóricos que se han incluído son los siguientes:

- Principios físicos: una explicación de las leyes físicas que explican el comportamiento del motor DC.
- Forma constructiva y principios de operación: una descripción de una de las formas constructivas del motor más comunes, y una explicación de cómo gracias a ese diseño se logra transformar la energía eléctrica en energía mecánica rotacional.
- Modelo matemático: a partir de las leyes físicas y teniendo en cuenta la forma constructiva se deriva el modelo matemático que describe el comportamiento del motor DC.

La documentación de las plantas de experimentación, los experimentos sugeridos y la implementación se realiza utilizando los *scripts* que se presentan en la sección 7.2.9.



#### **7.2.9 Ayudas para la documentación**

El *script* `latex/esqueleto.php` es una herramienta auxiliar para la tarea de documentación (DC). El script lee desde la base de datos la información de una o más plantas experimentales y genera varios archivos en formato  $\text{\LaTeX}$  con dicha documentación. Específicamente, el *script* construye:

- DC 1. Una carpeta de nombre definido por el usuario, en la que se incluirán los demás archivos y subcarpetas creados.
- DC 2. Un archivo de nombre definido por el usuario, con el código  $\text{\LaTeX}$  del archivo principal y que enlaza los demás archivos. En especial, enlaza el archivo `latex/preambulo.tex` que contiene las definiciones necesarias para la compilación del documento.
- DC 3. Varios archivos en formato  $\text{\LaTeX}$ :
  - `archivos.tex`: contiene información relativa al código fuente de la implementación en lenguaje Modelica.

- `experimentos.tex`: contiene información relativa a las plantas de experimentación.
- `implementación.tex`: archivo vacío. Se espera que el usuario redacte en este archivo la explicación sobre cómo se realizó la implementación de la planta de experimentación.
- `modelo.tex`: archivo vacío. Se espera que el usuario redacte en este archivo la explicación sobre los fundamentos teóricos del modelo de las plantas de experimentación.
- `presentacion.tex`: archivo vacío. Se espera que el usuario redacte en este archivo una breve presentación de las plantas de experimentación.
- `referencias.tex`: archivo vacío. Se espera que el usuario incluya en este archivo las instrucciones  $\text{\LaTeX}$  relacionadas con las citas bibliográficas.
- `resumen.tex`: contiene la descripción de la planta de experimentación.

- DC 4. Una colección de archivos  $\text{\LaTeX}$  de nombre `exp/expXXX.tex` con información general de cada planta de experimentación, y el listado de experimentos sugeridos asociados.
- DC 5. Una colección de archivos gráficos en formato `ps` de nombre `expFig/expXXX.ps` con imágenes de las figuras y animaciones de cada planta de experimentación.
- DC 6. Una colección de tablas en formato  $\text{\LaTeX}$  de nombre `expTab/expTabXXX.tex` con información de todos los parámetros, gráficas, animaciones y tablas de salida de datos de cada planta de experimentación.
- DC 7. Una colección de archivos  $\text{\LaTeX}$  de nombre `expTex/expXXX.tex`. Se espera que el usuario redacte en estos archivos la información específica que desee incorporar a la documentación de cada planta de experimentación.
- DC 8. Una carpeta vacía de nombre `figuras`. Se espera que el usuario adicione en esta carpeta las figuras que se incluyan en la documentación.

### Ejemplo 7.9: Ayuda para la generación de documentación.

La documentación asociada al modelo, que se presenta en el capítulo 8 (al igual que toda la documentación contenida en la parte III) ha sido elaborada a partir de los archivos generados por el script `latex/esqueleto.php`. Específicamente las tablas 8.1, 8.2, 8.3, 8.4 y 8.5 así como el listado de experimentos sugeridos de la sección 8.2 han sido construidos automáticamente a partir de la información administrada por UNVirtualLab.



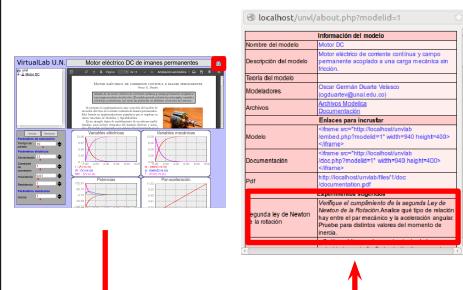
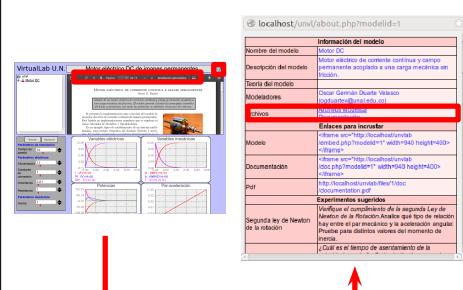
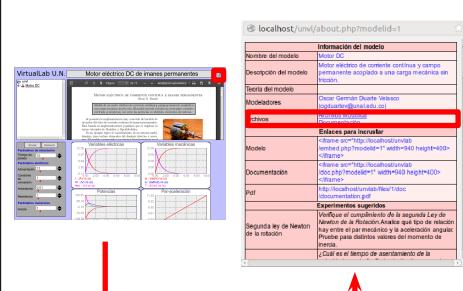
### 7.3 MANUAL DEL EXPERIMENTADOR

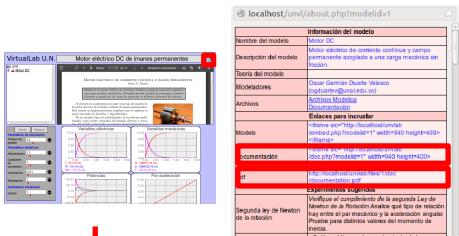
El manual del experimentador de UNVirtualLab se ha elaborado como una colección de preguntas frecuentes, que se consignan en la tabla 7.7. En la implementación en línea del manual cada pregunta se acompaña de un breve video que muestra cada procedimiento. En este documento los videos se han sustituido por las imágenes de la tabla 7.7.

Tabla 7.7 Listado de preguntas frecuentes relativas a UNVirtualLab

Pregunta	Respuesta	Imagen
1. ¿Cómo selecciono una planta de experimentación?	El menú de selección muestra el listado de plantas de experimentación disponibles. Las plantas están agrupadas en secciones. Para seleccionar una planta, navegue en el menú a lo largo de las secciones y haga clic sobre la planta deseada.	The screenshot shows the UNVirtualLab interface. On the left, there's a navigation menu with sections like Biología, Física, Química, Electrónica, and Robótica. Below it, there are dropdown menus for parameters such as 'Día del año' (57), 'Dirección del viento' (0), 'Temperatura mínima de aire' (25), 'Temperatura máxima de aire' (5), 'Velocidad del viento' (0.61), and 'Corriente' (11). The main area displays a simulation titled 'Calentamiento de un cable aéreo desnudo'. It includes a text box with a warning about overheating due to sunlight and wind, and two graphs: one for 'Flujo de calor' (heat flow) and another for 'Corriente' (current) over time.

Pregunta	Respuesta	Imagen
2. ¿Cómo lanzo una simulación?	Haga clic sobre el botón “Simular” que está en el re cuadro de los controles	
3. ¿Cómo modifico las condiciones de la simulación?	Utilice los controles disponibles para ello. Tiene tres alternativas: 1) escribir el valor deseado; 2) utilizar las flechas para aumentar o disminuir el valor; y 3) utilizar el botón deslizante para ajustar el valor. El botón “Restaurar” le permite recuperar los valores por defecto de todos los parámetros.	
4. ¿Cómo descargo los resultados de la simulación?	Haga clic sobre el botón “Descargar” que está sobre la tabla de resultados. El archivo de descarga está en formato CSV, separado por tabulador. El separador decimal empleado es la coma.	

Pregunta	Respuesta	Imagen
5. ¿Dónde encuentro los experimentos sugeridos para realizar con la planta?	Haga clic sobre el logo de ayuda. Se desplegará una ventana que contiene información sobre la planta experimental, entre la que se encuentra el listado de experimentos sugeridos.	
6. ¿Cómo descargo la documentación?	Haga clic sobre el logo de ayuda. Se desplegará una ventana que contiene información sobre la planta experimental. En la línea marcada como "Archivos" encontrará el enlace a la "Documentación" que está en formato PDF. También puede utilizar las opciones de descarga del visualizador de PDF de su navegador, si este las tiene.	
7. ¿Cómo descargo el código fuente de la simulación?	Haga clic sobre el logo de ayuda. Se desplegará una ventana que contiene información sobre la planta experimental. En la línea marcada como "Archivos" encontrará el enlace a los "Archivos Modelica", que aparecen comprimidos en un único archivo en formato tar.gz.	

Pregunta	Respuesta	Imagen
8. ¿Cómo inserto la planta de experimentación en un documento SCORM o HTML?	Haga clic sobre el logo de ayuda. Se desplegará una ventana que contiene información sobre la planta experimental. En la línea marcada como “Modelo”, bajo la marca “Enlaces para incrustar” encontrará el código en lenguaje HTML que necesita para incrustar en su documento el experimento. Este código genera un <i>frame</i> interno que contiene todos los instrumentos de la planta, es decir, que contiene tanto los controles como los visualizadores.	
9. ¿Cómo inserto la documentación en un documento SCORM o HTML?	Haga clic sobre el logo de ayuda. Se desplegará una ventana que contiene información sobre la planta experimental. En la línea marcada como “Documentación”, bajo la marca “Enlaces para incrustar” encontrará el código en lenguaje HTML que necesita para incrustar en su documento la documentación de la planta. También puede emplear el enlace de la línea “Pdf” para enlazar directamente el archivo pdf correspondiente.	
10. ¿Cómo incorporo más controles, gráficas o animaciones?	Esta opción sólo está disponible para el administrador del sitio.	



## **Parte III**

# **Plantas experimentales, modelos y experimentos**



# 8

## MOTOR ELÉCTRICO DE CORRIENTE CONTINUA E IMANES PERMANENTES

*Modelo de un motor eléctrico de corriente continua y campo permanente acoplado a una carga mecánica sin fricción. El modelo permite calcular las principales variables eléctricas y mecánicas, así como las potencias en distintos elementos del sistema.*

Se presenta la implementación muy conocida del modelo de un motor eléctrico de corriente continua de imanes permanentes. Está basada en implementaciones populares que se emplean en varios tutoriales de Modelica y OpenModelica. Es un ejemplo típico de modelamiento de un sistema multidominio, pues incluye elementos del dominio eléctrico y mecánico. El modelo corresponde a un sistema lineal invariante en el tiempo de segundo orden, que no incluye fenómenos de fricción mecánica ni pérdidas magnéticas.



Figura 8.1 Rotor de un motor DC de imanes permanentes<sup>1</sup>

<sup>1</sup>Tomada de [link:12] con licencia de Creative Commons.

El modelo se ha seleccionado por su simplicidad como el sistema mínimo de ejemplo que acompaña la instalación de UNVirtualLab.

## 8.1 EL MODELO

### 8.1.1 Principios físicos

Los motores eléctricos aprovechan el fenómeno físico conocido como fuerza de Lorentz. Este puede enunciarse de la siguiente forma (véase, por ejemplo, [Cha05], pág. 33):

Una carga  $q$  que se mueve con velocidad  $\mathbf{v}$  en presencia de un campo magnético cuya densidad de flujo magnético es  $\mathbf{B}$ , experimenta una fuerza  $\mathbf{f}$  causada por dicho campo, calculada como<sup>2</sup>:

$$\mathbf{f} = q(\mathbf{v} \times \mathbf{B}) \quad (8.1)$$

Si, en lugar de una carga eléctrica en movimiento, se analiza una corriente eléctrica  $I$  que circula a lo largo de un hilo de longitud  $L$  (y elemento diferencial  $d\mathbf{l}$ ), la ecuación 8.1 se convierte en:

$$\mathbf{f} = \int_L (\mathbf{I} \cdot d\mathbf{l} \times \mathbf{B}) \quad (8.2)$$

Otro de los fenómenos involucrados en el funcionamiento del motor eléctrico es la *inducción electromagnética*. Este fenómeno se explica por la ley de Faraday, que puede enunciarse como (véase, por ejemplo, [Cha05], pág. 35):

La variación de flujo magnético al interior de un circuito cerrado induce una fuerza electromotriz igual al negativo de la rata de cambio del flujo magnético. En el caso de una espira atravesada por un flujo magnético  $\psi$ , la tensión inducida  $v_e$  se calcula como:

$$v_e = -\frac{d\psi}{dt} \quad (8.3)$$

En particular, si un hilo conductor de longitud  $L$  (y vector de dirección  $\mathbf{l}$ ) que forma parte de un circuito cerrado que atraviesa un campo magnético de densidad de flujo  $\mathbf{B}$  a una velocidad  $\mathbf{v}$ , entonces dicho hilo experimenta entre sus extremos una diferencia de tensión  $v_e$

$$v_e = (\mathbf{v} \times \mathbf{B}) \cdot \mathbf{l} \quad (8.4)$$

---

<sup>2</sup>Las expresiones en las ecuaciones 8.1 y 8.2 son vectoriales; el operador  $\times$  representa el producto vectorial.

### 8.1.2 Forma constructiva y principio de operación

La figura 8.2 muestra el esquema de una forma constructiva posible para un motor DC de imanes permanentes que tiene las siguientes características principales<sup>3</sup>:

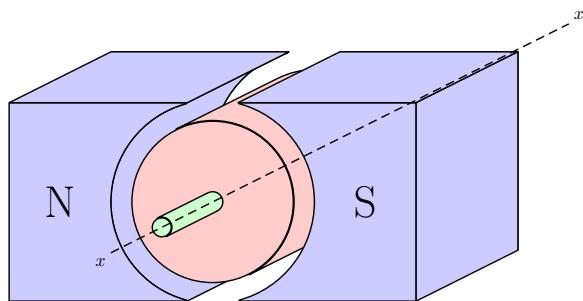


Figura 8.2 Esquema de una forma constructiva de un motor DC de imanes permanentes

- Existe un elemento fijo denominado *estator*. Adosado al estator se encuentra un imán que provee un campo magnético de densidad  $\mathbf{B}$ , cuya magnitud es  $B$ .
- Existe un elemento móvil denominado *rotor*. El rotor puede girar alrededor de un eje  $x - x'$ .
- El rotor afecta las líneas de campo magnético, tal como se observa en la figura 8.3: se destaca en la figura 8.3(c) que las líneas de campo magnético son prácticamente radiales en el entrehierro, es decir perpendiculares al eje de rotación del motor.
- Adosado al rotor hay un arrollamiento que consta de  $N$  espiras. El largo de cada espira es  $L$  y el ancho es  $r$ ; por tanto, tiene un área  $A = 2rL$ . La figura 8.4 ilustra la geometría de una espira.
- A través del arrollamiento circula la corriente  $I$ , cuya magnitud es  $i$ .
- La corriente ingresa (y sale) del arrollamiento a través de un juego de escobillas y colector. Este elemento realiza una comutación, de tal forma que el sentido de la corriente por el lado cercano a cada polo no se altera aunque la espira gire.

---

<sup>3</sup>Una descripción más extensa puede encontrarse en el capítulo 9 de [Cha05].

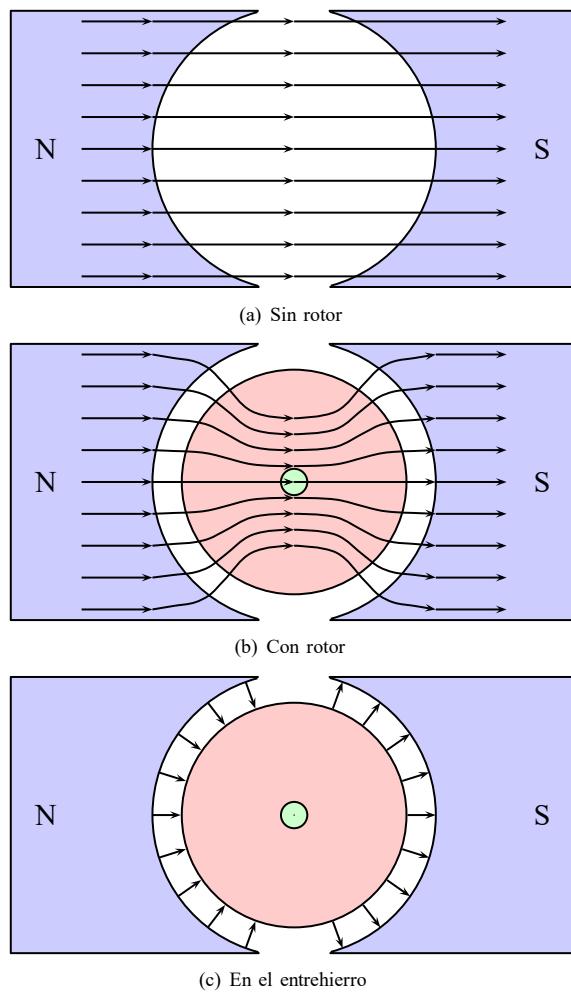
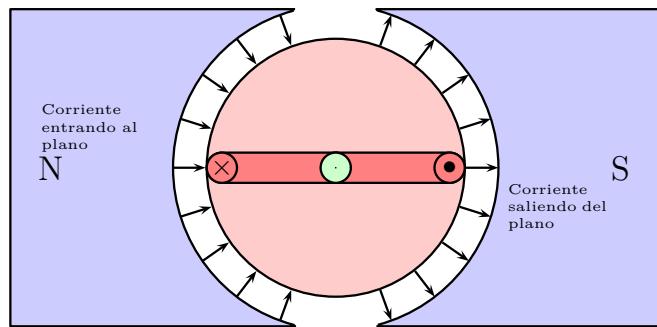


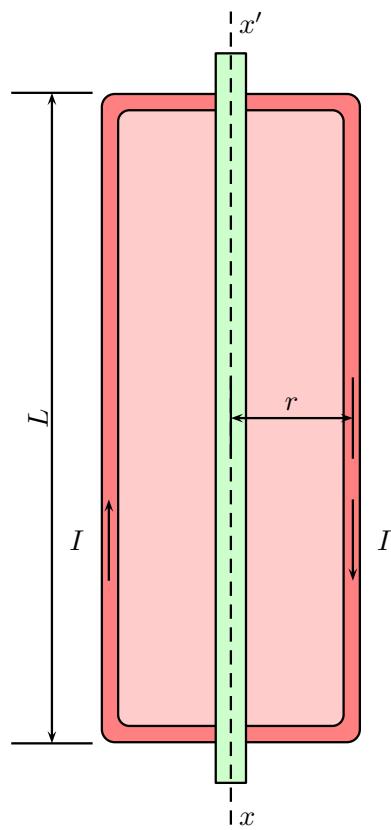
Figura 8.3 Líneas de campo magnético en un motor DC de imanes permanentes

La corriente que circula por el lado derecho de la espira interactúa con el campo magnético, produciendo una fuerza de Lorentz hacia arriba de valor  $f$  (véase figura 8.5). Al pasar por el lado izquierdo se produce otra fuerza de Lorentz del mismo valor  $f$ , pero hacia abajo. El valor de  $f$  se puede calcular como:

$$f = iLB \quad (8.5)$$



(a) Una espira. Vista de frente



(b) Una espira. Vista de planta

Figura 8.4 Geometría de una espira

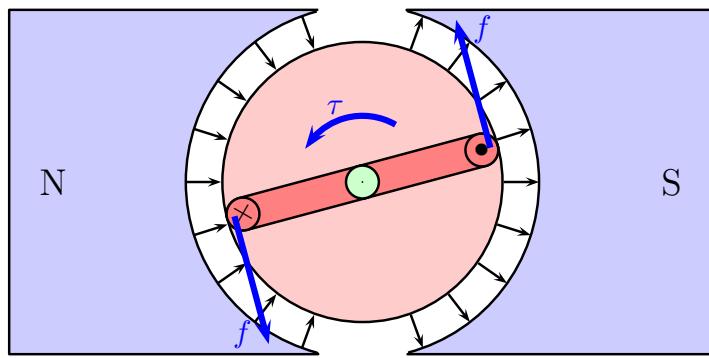


Figura 8.5 Fuerzas y torques en una espira

Cada una de esas fuerzas genera un par rotacional  $\tau = fr$  en el sentido antihorario. El par total recibido por el arrollamiento es  $T$ :

$$\begin{aligned} T &= 2N\tau \\ T &= 2NriLB \\ T &= NABi \\ T &= Ki \end{aligned} \tag{8.6}$$

en donde se ha definido  $K = NAB$ , que resulta ser una constante que depende de aspectos constructivos del motor. La ecuación 8.6 muestra que el par rotacional es directamente proporcional a la corriente que circula en el arrollamiento.

El par generado produce una aceleración angular  $\alpha$  en el rotor. Si el momento de inercia total del rotor y su carga mecánica acoplada es  $J$ , la segunda ley de Newton de la rotación establece que

$$T = J\alpha \tag{8.7}$$

Por otra parte, en cada uno de los dos segmentos de las espiras que son paralelos al eje de rotación se induce una tensión  $V_e$  (véase figura 8.6). La tensión en cada espira será  $2V_e$ , y la tensión en el arrollamiento completo será la fuerza electromotriz  $fem = 2NV_e$ . Si el rotor gira a una velocidad angular  $\omega$  la magnitud de la velocidad tangencial será  $v = wr$ . En esas condiciones:

$$\begin{aligned}
 fem &= 2NV_e \\
 fem &= -2NvBL \\
 fem &= -2N\omega rLB \\
 fem &= -NAB\omega \\
 fem &= -K\omega
 \end{aligned} \tag{8.8}$$

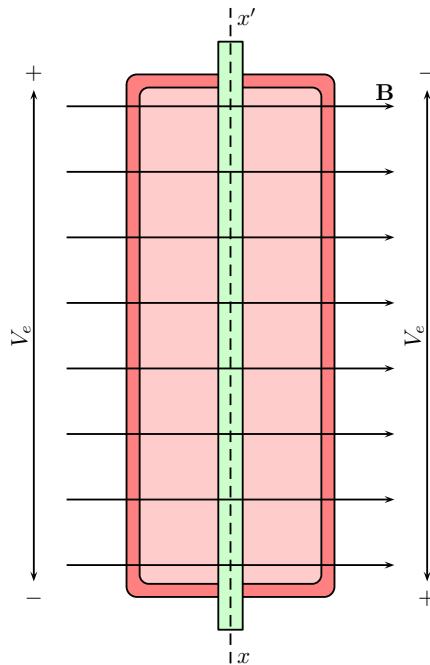


Figura 8.6 Fuerza electromotriz inducida en una espira

### 8.1.3 Modelo matemático

#### 8.1.3.1 Ecuación diferencial

La figura 8.7 muestra el diagrama del sistema modelado. Una fuente de tensión de valor  $v_s(t)$  suministra la energía al sistema. El circuito eléctrico del motor se ha modelado como un arreglo en serie de tres elementos:

- Un resistor lineal cuya resistencia es  $R$ .
- Un inductor lineal cuya inductancia es  $L$ .

- Una fuerza electromotriz  $fem$  de constante  $K$ . La tensión en la  $fem$  se representa por  $V_{fem}$ .

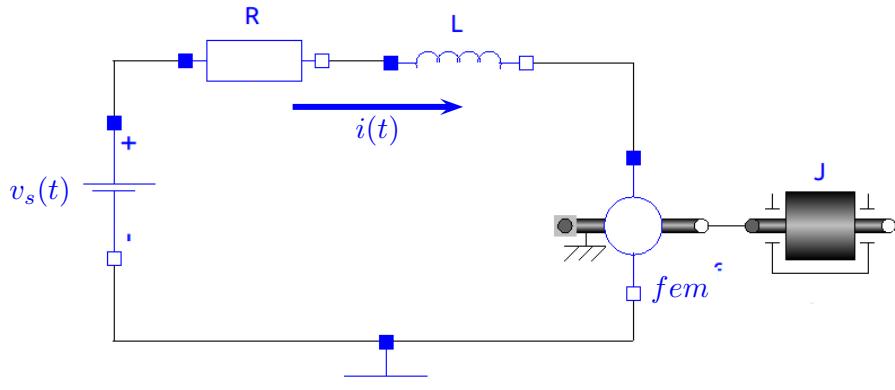


Figura 8.7 Diagrama de un motor DC de imanes permanentes acoplado a una carga mecánica

Utilizando la ley de tensiones de Kirchhoff, y haciendo uso de la ecuación 8.5 se puede escribir la ecuación del circuito eléctrico:

$$\begin{aligned} v_s(t) &= Ri(t) + L \frac{di(t)}{dt} + V_{fem} \\ v_s(t) &= Ri(t) + L \frac{di(t)}{dt} + K\omega(t) \end{aligned} \quad (8.9)$$

A partir de las ecuaciones 8.6 y 8.7 se puede escribir:

$$\begin{aligned} Ki(t) &= J\alpha(t) \\ Ki(t) &= J \frac{d\omega(t)}{dt} \\ i(t) &= \frac{J}{K} \frac{d\omega(t)}{dt} \end{aligned} \quad (8.10)$$

Lo que permite reescribir la ecuación 8.9 como

$$\begin{aligned} v_s(t) &= \frac{RJ}{K} \frac{d\omega(t)}{dt} + \frac{LJ}{K} \frac{d^2\omega(t)}{dt^2} + K\omega(t) \\ v_s(t) &= \frac{LJ}{K} \frac{d^2\omega(t)}{dt^2} + \frac{RJ}{K} \frac{d\omega(t)}{dt} + K\omega(t) \end{aligned} \quad (8.11)$$

### 8.1.3.2 Función de transferencia

La ecuación 8.11 corresponde a un sistema lineal de segundo orden. Para realizar un análisis en el dominio de Laplace, podemos tomar la velocidad angular  $\omega(t)$  como la salida del sistema (véase, por ejemplo, [Dua05], pág.47). En esas condiciones, la función de transferencia resulta ser  $F(s) = \frac{W(s)}{V_s(s)}$  y puede obtenerse de la siguiente forma:

$$\begin{aligned}
 V_s(s) &= \frac{LJ}{K}W(s)s^2 + \frac{RJ}{K}W(s)s + KW(s) \\
 V_s(s) &= \left( \frac{LJ}{K}s^2 + \frac{RJ}{K}s + K \right) W(s) \\
 V_s(s) &= \frac{LJs^2 + RJs + K^2}{K}W(s) \\
 \frac{W(s)}{V_s(s)} &= \frac{K}{LJs^2 + RJs + K^2} \\
 \frac{W(s)}{V_s(s)} &= \frac{K}{LJs^2 + RJs + K^2} \\
 F(s) &= \frac{K}{LJs^2 + RJs + K^2} \\
 F(s) &= \frac{K/LJ}{s^2 + (R/L)s + K^2/LJ}
 \end{aligned} \tag{8.12}$$

### 8.1.3.3 Estabilidad

Los polos de la función de transferencia obtenida en la ecuación 8.12 son

$$p_{1,2} = -\frac{R}{L} \pm \sqrt{\frac{R^2}{L^2} - \frac{4K^2}{LJ}} \tag{8.13}$$

De la ecuación 8.13 se desprende que el sistema no puede tener polos con parte real positiva, y por tanto su comportamiento es estable (véase, por ejemplo, [Dua05], pág.71).

### 8.1.3.4 Estado estacionario

A partir de la ecuación 8.12 es posible conocer la velocidad angular de estado estacionario ( $w_{ee}$ ) ante una entrada de tensión escalón de altura  $V$  (véase, por ejemplo, [Dua05], pág. 93):

$$\begin{aligned}
 w_{ee} &= \lim_{s \rightarrow 0} s \frac{V}{s} F(s) \\
 w_{ee} &= \lim_{s \rightarrow 0} s \frac{V}{s} \frac{K/LJ}{s^2 + R/Ls + K^2/LJ} \\
 w_{ee} &= V \frac{K/LJ}{0^2 + R/L0 + K^2/LJ} \\
 w_{ee} &= V \frac{K/LJ}{K^2/LJ} \\
 w_{ee} &= \frac{V}{K}
 \end{aligned} \tag{8.14}$$

### 8.1.3.5 Comportamiento transitorio

De la ecuación 8.13 se desprende que el comportamiento transitorio del sistema puede ser subamortiguado, sobreamortiguado o en amortiguamiento crítico (véase, por ejemplo, [Dua05], pág.73). La condición de subamortiguamiento resulta ser:

$$\begin{aligned}
 \frac{4K^2}{LJ} &> \frac{R^2}{L^2} \\
 \frac{4K^2}{J} &> \frac{R^2}{L} \\
 K^2 &> \frac{JR^2}{4L} \\
 K &> \frac{R}{2} \sqrt{\frac{J}{L}}
 \end{aligned} \tag{8.15}$$

## 8.2 PLANTAS DE EXPERIMENTACIÓN Y EXPERIMENTOS SUGERIDOS

### Planta de experimentación 1: Motor eléctrico DC de imanes permanentes.

**Presentación:** Esta planta implementa el modelo de un motor eléctrico de corriente continua y campo permanente acoplado a una carga mecánica sin fricción. La planta permite modificar los parámetros eléctricos y mecánicos principales, y visualizar el comportamiento de las variables mecánicas, eléctricas y las potencias en los diferentes elementos del sistema.

**Instrumentación<sup>4</sup>:** el modelo cuenta con 6 parámetros ajustables organizados en 3 grupos de controles (véase tabla 8.1). Como resultado del experimento, el programa despliega:

- 11 curvas organizadas en 4 gráficos (véase tabla 8.2).
- 1 animación en 2D (véase tabla 8.3).
- Una tabla de datos del comportamiento de 11 variables (véase tabla 8.4).

**Experimentos sugeridos:** el siguiente es el listado de experimentos sugeridos:

***Experimento 1.1: Segunda ley de Newton de la rotación.***

Verifique el cumplimiento de la segunda Ley de Newton de la Rotación. Analice qué tipo de relación hay entre el par mecánico y la aceleración angular. Pruebe para distintos valores del momento de inercia.

***Experimento 1.2: Tiempo de asentamiento de la velocidad angular.***

¿Cuál es el tiempo de asentamiento de la velocidad angular? ¿Qué relación tiene con el momento de inercia, con la resistencia eléctrica, con la inductancia eléctrica? Obtenga el tiempo en el que la velocidad angular llega al 95 % de su valor final. Realice el cálculo para diferentes condiciones de momento de inercia, resistencia eléctrica e inductancia eléctrica.

***Experimento 1.3: Velocidad angular estacionaria.***

¿Qué relación hay entre el valor estacionario de la velocidad angular y la tensión de alimentación? Encuentre el efecto que tiene el cambio en la tensión de alimentación sobre la velocidad angular estacionaria. Utilice el modelo matemático para deducir una expresión sobre esta relación y contrástela con el comportamiento del modelo.

***Experimento 1.4: Efectos de la carga mecánica en el circuito eléctrico.***

¿Cómo incide la carga mecánica sobre las variables eléctricas del motor? Explore la relación entre la carga mecánica y las diferentes variables de tipo eléctrico que intervienen en el motor.

***Experimento 1.5: Simulación de dispositivos.***

Aplique el modelo para el diseño de un elemento rotacional activado por un motorDC. Obtenga un modelo de software para simular el comportamiento de un dispositivo real que emplee un motor DC.

***Experimento 1.6: Suma de potencias.***

Verifique que la suma total de potencias en el sistema es nula. Utilice la tabla de datos para verificar en una hoja electrónica cuánto suman las potencias en los diferentes elementos del sistema. Repita el ejercicio para diferentes condiciones de simulación.

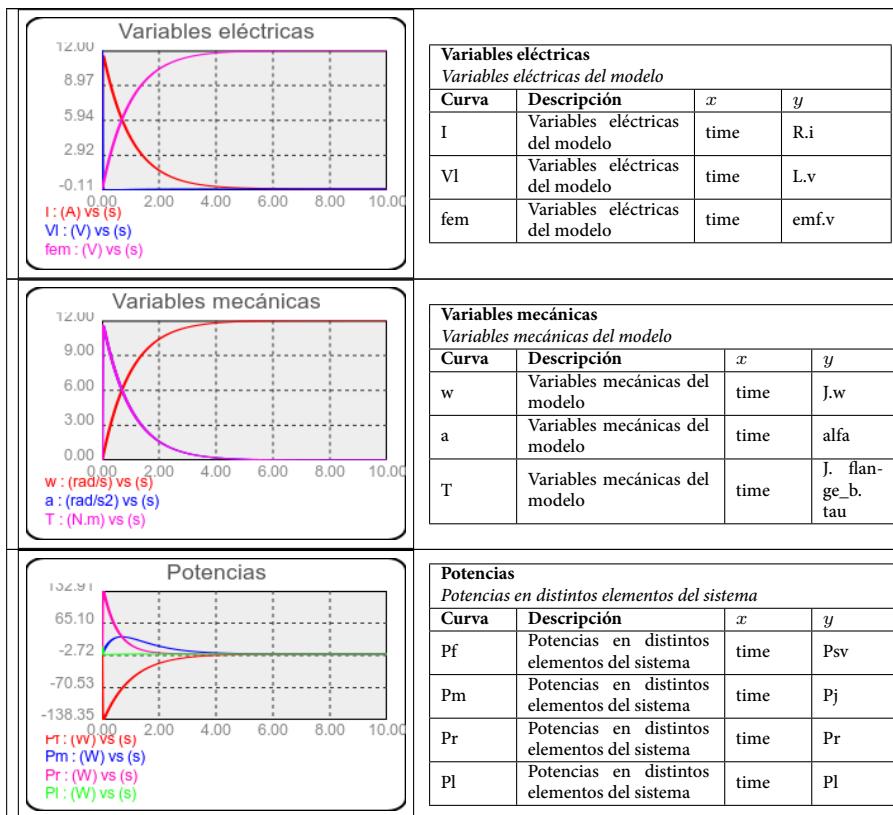
Tabla 8.1 Parámetros del experimento 1, “Motor eléctrico DC de imanes permanentes”<sup>5</sup>

Título:	Motor DC		
Descripción:	Motor eléctrico de corriente continua y campo permanente acoplado a una carga mecánica sin fricción.		
Créditos	Implementación	Oscar Germán Duarte Velasco	
	e-mail	ogduartev@unal.edu.co	
Parámetros			
Grupo	nombre Modelica	nombre	descripción
Parámetros mecánicos	J.J	Inercia	Momento de inercia de la carga mecánica (kg.m <sup>2</sup> )

<sup>4</sup>La información en las tablas siguientes se muestra tal como aparece en la base de datos. Por esta razón hay datos tanto en español como en inglés.

<b>Parámetros eléctricos</b>	<i>R.R</i>	Resistencia	Resistencia eléctrica del motor (Ohm)
	<i>L.L</i>	Inductancia	Inductancia del motor (H)
	<i>emf.k</i>	Constante de conversión	Constante de conversión electromagnética (N.m/A)
	<i>step.height</i>	Alimentación	Tensión de alimentación (V)
<b>Parámetros de simulación</b>	<i>stopTime</i>	Tiempo de parada	Tiempo de parada de la simulación

Tabla 8.2 Figuras del experimento 1, "Motor eléctrico DC de imanes permanentes"



<sup>5</sup>La información en las tablas siguientes se muestra tal como aparece en la base de datos. Por esta razón hay datos tanto en español como en inglés.

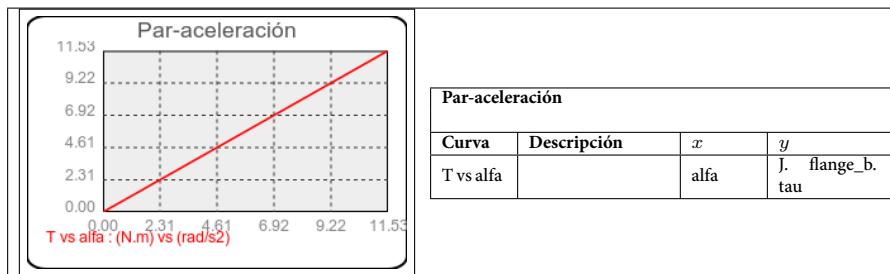


Tabla 8.3 Animaciones del experimento 1, “Motor eléctrico DC de imanes permanentes”

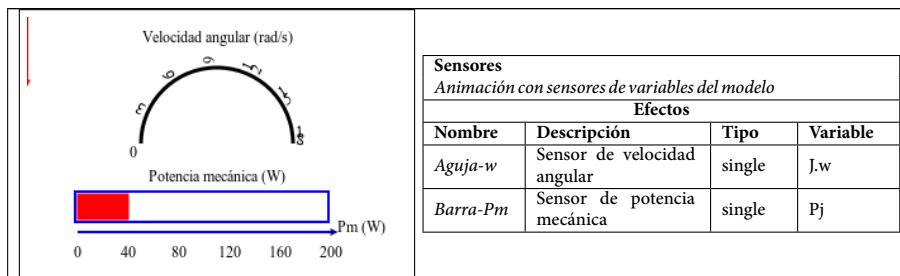


Tabla 8.4 Variables en la tabla de resultados del experimento 1, “Motor eléctrico DC de imanes permanentes”

Variable	Descripción	Unidades
time	time	s
R.i	Current flowing from pin p to pin n	A
L.v	Voltage drop between the two pins (= p.v - n.v)	V
emf.v	Voltage drop between the two pins	V
J.w	Absolute angular velocity of component (= der(phi))	rad/s
alfa		rad/s <sup>2</sup>
J.flange_b.tau	Cut torque in the flange	N.m
Psv		W
Pj		W
Pr		W
Pl		W

### 8.3 LA IMPLEMENTACIÓN

La implementación del modelo se basa en la propuesta en [FPA<sup>+</sup>13a] (pág. 18). A este modelo se la han adicionado las variables de aceleración angular y la potencia transformada en varios elementos del sistema.

El modelo utiliza una única clase DCmotor, que se construye interconectando elementos disponibles en la Modelica Standard Library (MSL). El archivo DCmotor.mo muestra el código fuente de la implementación.

No se ha hecho uso de los elementos del paquete Electrical.Machines de la Modelica Standard Library, pese a que allí se incluye una implementación más detallada de las máquinas de corriente continua en general. Esta decisión se ha tomado para que el ejemplo resulte más ilustrativo en relación con el uso de UNVirtualLab.

### 8.3.1 Listado de archivos

La tabla 8.5 muestra el listado de los archivos fuente de la implementación del modelo.

Tabla 8.5 Archivos del modelo

Número	Archivo
8.1	DCmotor.mo

Archivo 8.1 DCmotor.mo

```

model DCmotor
  Modelica.Electrical.Analog.Basic.Resistor R(R = 1);
  Modelica.Electrical.Analog.Basic.Inductor L(L = 0.01);
  Modelica.Electrical.Analog.Basic.EMF emf(k = 1);
  Modelica.Electrical.Analog.Basic.Ground g;
  Modelica.Blocks.Sources.Step step(height = 12);
  Modelica.Mechanics.Rotational.Components.Inertia J(J = 1);
  Modelica.Electrical.Analog.Sources.SignalVoltage SV;
  Modelica.SIunits.Power Psv "Potencia en la fuente";
  Modelica.SIunits.Power Pr "Potencia en la resistencia";
  Modelica.SIunits.Power Pl "Potencia en la inductancia";
  Modelica.SIunits.Power Pj "Potencia en la carga mecánica";
  Modelica.SIunits.Power PT "Potencia total";
  Modelica.SIunits.AngularAcceleration alfa "Aceleración angular";
equation
  connect(L.n,emf.p);
  connect(R.n,L.p);
  connect(SV.p,R.p);
  connect(SV.n,g.p);
  connect(emf.n,g.p);
  connect(step.y,SV.v);
  connect(emf.flange,J.flange_b);
  Psv = SV.v * SV.i;
  Pr = R.v * R.i;
  Pl = L.v * L.i;
  Pj = J.flange_b.tau * J.w;
  PT=Psv+Pr+Pl+Pj+PT;
  alfa=der(J.w);
end DCmotor;
```



# 9

## CALENTAMIENTO DE CABLES AÉREOS

*El modelo implementado relaciona el calentamiento y la elongación de un cable aéreo desnudo con varios aspectos:*

- *La corriente eléctrica que circula por el conductor.*
- *Las condiciones ambientales.*
- *Las condiciones de tendido.*

Para transportar grandes cantidades de energía eléctrica a través de largas distancias se utilizan líneas de transmisión como las que se muestran en la figura 9.1. La energía eléctrica viaja a través de los cables aéreos, que son conductores de electricidad que están suspendidos en el aire gracias a las torres que los soportan. El peso de los cables causa una deflexión en estos. Tal deflexión varía con el calentamiento que sufren los cables, debido al paso de la energía eléctrica y a las condiciones ambientales.



Figura 9.1 Línea de transmisión<sup>1</sup>

---

<sup>1</sup>Tomada de [link:13] con licencia de Creative Commons.

La deflexión de los cables es un factor que incide en la seguridad de la línea. El diseñador debe prever que no se violen las distancias mínimas de seguridad. Para ello, debe estudiar la manera como se afecta la geometría de la línea en diferentes condiciones de operación.

## 9.1 EL MODELO

El modelo implementado permite simular simultáneamente dos fenómenos:

- El calentamiento de un cable aéreo por el que circula una corriente eléctrica.
- El efecto que el calentamiento tiene sobre la geometría de la curva que describe el cable.

### 9.1.1 Modelo térmico estático

El modelo térmico implementado es el sugerido por la norma técnica IEEE 738 (véase [IEE07]). Se trata de un balance de calor en el que intervienen los siguientes fenómenos:

- Calentamiento por efecto Joule.
- Calentamiento por radiación solar.
- Enfriamiento por convección.
- Enfriamiento por radiación.

La ecuación 9.1 representa la condición de equilibrio del calentamiento (balance de calor) de un conductor eléctrico al aire libre:

$$0 = \frac{1}{mC_p} [R(T_c)I^2 + q_s - q_c - q_r] \quad (9.1)$$

en donde:

- $T_c$  es la temperatura del conductor.
- $mC_p$  es la capacitancia térmica del conductor.
- $R$  es la resistencia por unidad de longitud del conductor, que es función de la temperatura.
- $I$  es la corriente por el conductor. El término  $R(T_c)I^2$  es la ganancia de calor por efecto Joule.

- $q_s$  es la ganancia de calor por radiación solar.
- $q_c$  es la pérdida de calor por convección.
- $q_r$  es la pérdida de calor por radiación.

La norma IEEE 738 especifica los procedimientos para la estimación de  $q_s$ ,  $q_c$  y  $q_r$ .

### 9.1.2 Modelo térmico dinámico

La ecuación 9.2 representa la dinámica del calentamiento de un conductor eléctrico al aire libre:

$$\frac{dT_c}{dt} = \frac{1}{mC_p} [R(T_c)I^2 + q_s - q_c - q_r] \quad (9.2)$$

Se estudia aquí la solución de la ecuación con  $I$  constante, y con condición inicial  $T_c(0) = T_0$ . Aunque los términos  $q_c$  y  $q_r$  dependen de  $T_c$ , en los métodos de solución que se presentan se suponen constantes; esto significa que los métodos propuestos son válidos para cortos intervalos de tiempo (típicamente de 1 minuto).

#### 9.1.2.1 Método 1: aproximación de resistencia constante

En este método se supone que  $R$  es independiente de  $T_c$ . En estas condiciones se tiene:

$$\frac{dT_c}{dt} = \alpha \quad (9.3)$$

$$\alpha = \frac{1}{mC_p} [RI^2 + q_s - q_c - q_r] \quad (9.4)$$

cuya solución es

$$T_c(t) = T_0 + \alpha t \quad (9.5)$$

#### 9.1.2.2 Método 2: aproximación de resistencia lineal con la temperatura

En este método se supone una variación lineal de  $R$  con  $T_c$ . Se supone, además, que se conoce el valor de  $R$  para dos temperaturas:

$$R(T_L) = R_L \quad R(T_H) = R_H \quad (9.6)$$

de tal manera que la resistencia a una temperatura  $T_c$  está dada por

$$R(T_c) = R_L + (T_c - T_L) \frac{(R_H - R_L)}{(T_H - T_L)} \quad (9.7)$$

o lo que es igual:

$$R(T_c) = \sigma T_c + \beta \quad (9.8)$$

$$\sigma = \frac{(R_H - R_L)}{(T_H - T_L)} \quad \beta = R_L - \sigma T_L \quad (9.9)$$

En estas condiciones, la ecuación 9.2 se convierte en

$$\frac{dT_c}{dt} = \frac{1}{mC_p} [(\sigma T_c + \beta)I^2 + q_s - q_c - q_r] \quad (9.10)$$

$$\frac{dT_c}{dt} = \frac{\sigma I^2}{mC_p} T_c + \frac{1}{mC_p} [\beta I^2 + q_s - q_c - q_r] \quad (9.11)$$

que puede escribirse en forma resumida así:

$$\frac{dT_c}{dt} = a_0 T_c + b_0 \quad (9.12)$$

$$a_0 = \frac{\sigma I^2}{mC_p} \quad b_0 = \frac{1}{mC_p} [\beta I^2 + q_s - q_c - q_r] \quad (9.13)$$

La solución de la ecuación 9.12 es

$$T_c(t) = -\frac{b_0}{a_0} + \frac{a_0 T_0 + b_0}{a_0} e^{a_0 t} \quad (9.14)$$

### 9.1.2.3 Estimación de corriente. Método 1

Supóngase que son conocidos  $T_0$ ,  $T_c(t)$  y  $t$  en la ecuación 9.5. Se requiere estimar el valor de  $I$ . Para ello se despeja  $\alpha$ :

$$\alpha = \frac{T_c(t) - T_0}{t} \quad (9.15)$$

y posteriormente se despeja  $I$  de 9.4

$$I = \sqrt{\frac{\alpha m C_p - q_s + q_c + q_r}{R}} \quad (9.16)$$

### 9.1.2.4 Estimación de corriente. Método 2

Supóngase que son conocidos  $T_0$ ,  $T_c(t)$  y  $t$  en la ecuación 9.14. Se requiere estimar el valor de  $I$ .

**Caso 1. Cambio de temperatura:** de la ecuación 9.13 se obtiene:

$$I^2 = \frac{a_0 m C_p}{\sigma} \quad b_0 = \frac{\beta a_0}{\sigma} + \frac{1}{m C_p} [q_s - q_c - q_r] \quad (9.17)$$

que en forma resumida se pueda expresar como:

$$b_0 = \theta a_0 + \eta \quad (9.18)$$

con

$$\theta = \frac{\beta}{\sigma} \quad \eta = \frac{1}{m C_p} [q_s - q_c - q_r] \quad (9.19)$$

Lo que permite expresar 9.14 como

$$T_c(t) = T_0 e^{a_0 t} - \frac{\theta a_0 + \eta}{a_0} (1 - e^{a_0 t}) \quad (9.20)$$

La ecuación 9.20 se resuelva para  $a_0$  mediante métodos numéricos. Posteriormente se calcula  $I$  a partir de 9.17:

$$I = \sqrt{\frac{a_0 m C_p}{\sigma}} \quad (9.21)$$

**Caso 2. Temperatura constante:** debe asumirse la condición de estado estacionario y emplear:

$$R I^2 + q_s = q_c + q_r \quad (9.22)$$

### 9.1.3 Modelo mecánico

El modelo mecánico del cable está ampliamente documentado en textos de ingeniería mecánica y eléctrica (véase, por ejemplo, [FB13]). Se trata de una catenaria, tal como la que se muestra en la figura 9.2: está apoyada en  $A$  y  $B$ , con un desnivel  $\Delta y$ . La separación horizontal entre apoyos es  $S$ . La tensión longitudinal es  $T_{en}$ . La tensión horizontal es  $H$ . La longitud del conductor es  $L$ . El peso por unidad de longitud es  $W$ . La temperatura es  $T$ . El punto más bajo ( $O$ ) se ubica a una distancia  $S_A$  del apoyo  $A$  y a una distancia  $S_B$  del apoyo  $B$ . La longitud del conductor desde el apoyo  $A$  hasta el punto más bajo es  $L_A$ . La longitud del conductor desde el apoyo  $B$  hasta el punto más bajo es  $L_B$ . La altura desde el punto más bajo hasta el apoyo en  $A$  es  $y_A$ . La altura desde el punto más bajo hasta el apoyo en  $B$  es  $y_B$ . La altura del apoyo  $A$  respecto al nivel de referencia es  $h_A$ .

La flecha  $D$  es la máxima distancia vertical que hay entre la línea imaginaria que une los dos apoyos y la catenaria. La flecha sucede en el punto en el que la tangente de la catenaria es igual a la pendiente de la línea imaginaria que une los dos apoyos. Este punto está a una distancia horizontal  $\bar{x}_f$  del punto más bajo.

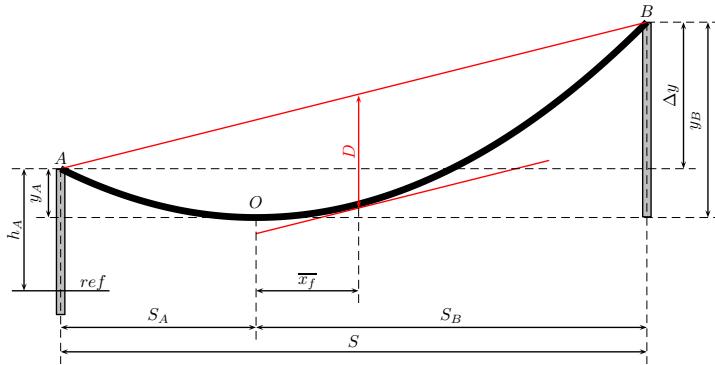


Figura 9.2 Geometría de la catenaria

### 9.1.3.1 Geometría

La elevación  $\frac{y}{x}$  respecto al punto más bajo se calcula así:

$$y(\bar{x}) = \frac{H}{W} \cosh\left(\frac{W\bar{x}}{H}\right) - \frac{H}{W}; \quad (9.23)$$

en donde  $\bar{x}$  es la distancia horizontal al punto más bajo. Sea  $x$  la distancia horizontal al apoyo  $A$ ; en esas condiciones se tiene:

$$\bar{x} = S_A - x \quad (9.24)$$

Y por tanto:

$$y(x) = \frac{H}{W} \cosh\left(\frac{W(S_A - x)}{H}\right) - \frac{H}{W} \quad (9.25)$$

La altura del conductor en  $x$  respecto al nivel de referencia es

$$h(x) = h_A - Y_A + y(x) \quad (9.26)$$

$$h(x) = h_A - \frac{H}{W} \cosh\left(\frac{WS_A}{H}\right) + \frac{H}{W} \cosh\left(\frac{W(S_A - x)}{H}\right) \quad (9.27)$$

### 9.1.3.2 Cálculo de $S_A$

La distancia del punto más bajo de la catenaria al apoyo más bajo es  $S_{PB}$ :

$$S_{PB} = \frac{S_A}{2} - \frac{H}{W} \sinh^{-1} \left\{ \frac{\Delta y/2}{\frac{H}{W} \sinh \left( \frac{W}{H} S_A/2 \right)} \right\} \quad (9.28)$$

$S_{PB}$  es igual a  $S_A$  o a  $S_B$  según el primer apoyo sea o no el más bajo de los dos. Expresado de otra forma:

$$S_A = \begin{cases} S_{PB} & \text{Si } \Delta y \geq 0 \\ A - S_{PB} & \text{Si } \Delta y < 0 \end{cases} \quad (9.29)$$

### 9.1.3.3 Cálculo de la flecha

Para determinar la flecha se calcula primero la pendiente  $m$  de la línea imaginaria que une los dos puntos:

$$m = \frac{\Delta y}{S} \quad (9.30)$$

La tangente de la catenaria se obtiene derivando la ecuación 9.23:

$$\frac{dy}{d\bar{x}} = \sinh \left( \frac{W\bar{x}}{H} \right) \quad (9.31)$$

El punto  $\bar{x}_f$  en el que la tangente de la catenaria iguala a  $m$  es, entonces:

$$\bar{x}_f = \frac{H}{W} \operatorname{asinh} \left( \frac{\Delta y}{S} \right) \quad (9.32)$$

La flecha  $D$  es la diferencia entre la altura de la recta imaginaria que une los dos apoyos  $y_r$  y la altura de la catenaria  $y_c$ , medidas en  $\bar{x}_f$ :

$$\begin{aligned} D &= y_r(\bar{x}_f) - y_f(\bar{x}_f) \\ y_r(\bar{x}_f) &= \frac{H}{W} \cosh \left( \frac{WS_B}{H} \right) - \frac{H}{W} - m(S_b - x_f) \\ y_c(\bar{x}_f) &= \frac{H}{W} \cosh \left( \frac{W\bar{x}_f}{H} \right) - \frac{H}{W} \end{aligned} \quad (9.33)$$

### 9.1.3.4 Cálculo de la tensión horizontal

La longitud total del cable  $L$  se obtiene con  $L = L_A + L_B$ :

$$L_A = \frac{H}{W} \sinh \left( \frac{WS_A}{H} \right) \quad L_B = \frac{H}{W} \sinh \left( \frac{WS_B}{H} \right) \quad (9.34)$$

La tensión longitudinal en el cable es<sup>2</sup>

$$Ten = H \cosh \frac{WS}{2H} \quad (9.35)$$

Consideramos ahora dos estados 0 y 1 para un mismo cable. En cada uno de ellos especificamos las siguientes variables:

- Tensión longitudinal, denotada por  $Ten_0$  y  $Ten_1$  para cada estado.
- Tensión horizontal, denotada por  $H_0$  y  $H_1$  para cada estado.
- Temperatura del conductor, denotada por  $T_0$  y  $T_1$  para cada estado.
- Longitud del conductor, denotada por  $L_0$  y  $L_1$  para cada estado.

Las longitudes del conductor en los dos estados satisfacen:

$$L_1 = L_0 \left[ 1 + a(T_1 - T_0) + \frac{Ten_1 - Ten_0}{EA} \right] \quad (9.36)$$

en donde  $a$  es el coeficiente de dilatación,  $E$  es el módulo de elasticidad y  $A$  es el área.

Si se supone que  $W$  no cambia del estado 0 al 1, el valor de  $H_1$  se obtiene al resolver:

$$\begin{aligned} \frac{H_1}{W} \sinh \left( \frac{WS_A}{H_1} \right) + \frac{H_1}{W} \sinh \left( \frac{W(S - S_A)}{H_1} \right) = \\ \left\{ \frac{H_0}{W} \sinh \left( \frac{WS_A}{H_0} \right) + \frac{H_1}{W} \sinh \left( \frac{W(S - S_A)}{H_0} \right) \right\} \\ \left[ 1 + a(T_1 - T_0) \frac{1}{EA} \left[ H_1 \cosh \frac{WS}{2H_1} - H_0 \cosh \frac{WS}{2H_0} \right] \right] \end{aligned} \quad (9.37)$$

---

<sup>2</sup>A una distancia  $S/2$  del punto más bajo.

## 9.2 PLANTAS DE EXPERIMENTACIÓN Y EXPERIMENTOS SUGERIDOS

### Planta de experimentación 2: Calentamiento de un cable aéreo desnudo.

**Presentación:** en este experimento se explora la manera como se afecta la temperatura del conductor y su flecha al alterar la corriente y las condiciones ambientales a lo largo de un periodo de veinticuatro horas.

Se han modelado dos perfiles: uno para la corriente y otro para la temperatura del aire (véase figura 9.3). Estos perfiles buscan simular la variabilidad de las dos condiciones a lo largo del día. El usuario puede modificar los perfiles ajustando los valores de  $I_{min_1}$ ,  $I_{max_1}$ ,  $I_{min_2}$ ,  $I_{max_2}$ ,  $T_{a,min}$  y  $T_{a,max}$ .

Las condiciones de viento (velocidad y dirección) también pueden ser modificadas por el usuario. En el experimento modelado estos valores se mantienen constantes a lo largo de las 24 horas. El usuario también puede modificar el día del año que se simula, y la posición geográfica del vano.

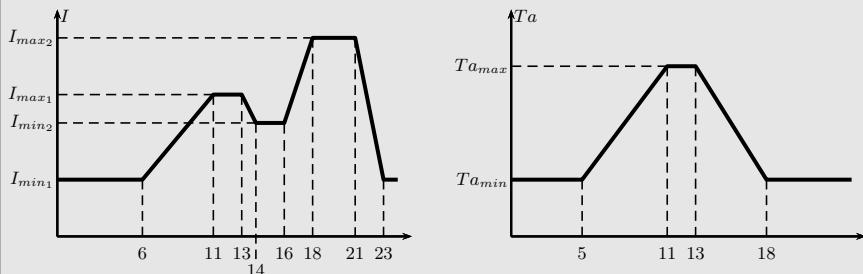


Figura 9.3 Perfiles de corriente y temperatura de aire para el experimento 2

**Instrumentación<sup>3</sup>:** el modelo cuenta con once parámetros ajustables organizados en 3 grupos de controles (véase tabla 9.1). Como resultado del experimento, el programa despliega:

- 8 curvas organizadas en 4 gráficos (véase tabla 9.2).
- Una tabla de datos del comportamiento de 9 variables (véase tabla 9.3).

**Experimentos sugeridos:** el siguiente es el listado de experimentos sugeridos:

***Experimento 2.1: Tiempos de respuesta.***

¿Qué tiempo de respuesta tiene el calentamiento del cable y la flecha, ante variaciones de corriente? Determine cuánto tiempo tarda en estabilizarse la temperatura del conductor y la flecha ante cambios bruscos de corriente. ¿Estos tiempos son iguales a lo largo de las 24 horas del día?

***Experimento 2.2: Fuentes de calentamiento y enfriamiento.***

¿Qué factores inciden en la cantidad de calor que entra y sale del conductor? Explore qué efecto tiene la variación de las condiciones ambientales, eléctricas y geográficas en los flujos de calor.

***Experimento 2.3: Ubicación geográfica.***

¿En qué lugares de Colombia se calentaría más fácilmente un conductor? A partir de las condiciones geográficas y climáticas de varios lugares de Colombia, explore el calentamiento de los conductores en dichos lugares.

***Experimento 2.4: Estaciones.***

¿El fenómeno de calentamiento del conductor es semejante en todos los días del año? ¿Cómo inciden las estaciones en los países en donde estas suceden? Explore la incidencia que tiene la variación de las condiciones climáticas a lo largo del año para: 1) un lugar específico en Colombia, y 2) para un lugar del planeta con estaciones pronunciadas.

### **Experimento 2.5: Margen de cargabilidad.**

¿De qué factores depende en mayor medida el margen de cargabilidad? El margen de cargabilidad es la cantidad de corriente adicional que podría circular por el conductor sin que se alcancen las condiciones límite. Este margen varía a lo largo del día. Explore los factores que afectan más este margen.

Tabla 9.1 Parámetros del experimento 2, “Calentamiento de un cable aéreo desnudo”

<b>Título:</b>	Elongación		
<b>Descripción:</b>	El experimento propuesto permite visualizar cómo inciden en el calentamiento de un cable aéreo dos aspectos: 1) La corriente eléctrica que circula por el conductor, y 2) Las condiciones ambientales.		
<b>Créditos</b>	<b>Implementación</b>	Oscar Germán Duarte Velasco	
	<b>e-mail</b>	ogduartev@unal.edu.co	
<b>Parámetros</b>			
Grupo	nombre Modelica	nombre	descripción
<b>Corriente</b>	<i>Imin</i>	I1	Prueba
	<i>Imax</i>	I2	
	<i>Imin2</i>	I3	
	<i>Imax2</i>	I4	
<b>Condiciones ambientales</b>	<i>windVel.k</i>	Velocidad de viento	Constant output value
	<i>windDir.k</i>	Dirección del viento	Constant output value
	<i>to.Day</i>	Día del año	Day of the year (1-365)
	<i>Tmin</i>	Temperatura mínima de aire	
	<i>Tmax</i>	Temperatura máxima de aire	
<b>Ubicación geográfica</b>	<i>span.He</i>	Altitud	Altitude above sea level in m
	<i>span.L</i>	Latitud	Latitude in deg

<sup>3</sup>La información en las tablas siguientes se muestra tal como aparece en la base de datos. Por esta razón hay datos tanto en español como en inglés.

Tabla 9.2 Figuras del experimento 2, “Calentamiento de un cable aéreo desnudo”

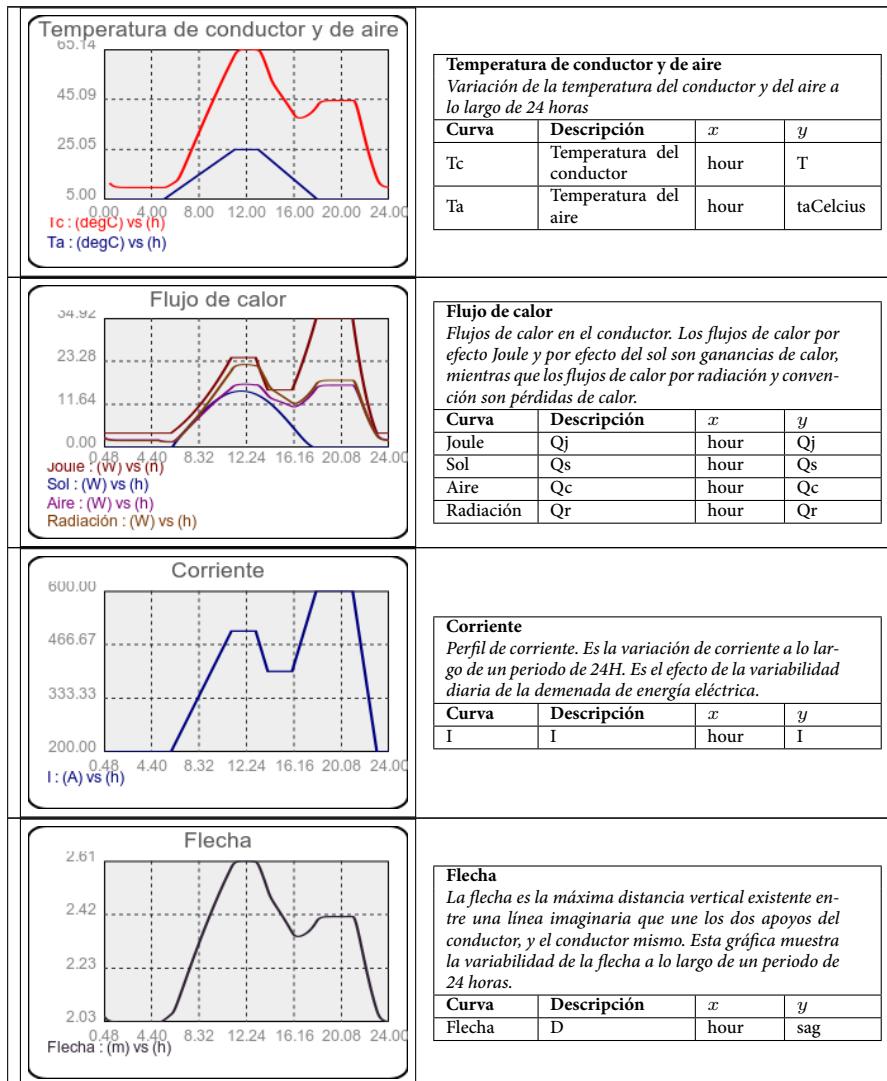


Tabla 9.3 Variables en la tabla de resultados del experimento 2, “Calentamiento de un cable aéreo desnudo”

Variable	Descripción	Unidades
hour		h
T		degC

Variable	Descripción	Unidades
taCelcius		degC
Qj		W
Qs		W
Qc		W
Qr		W
I		A
sag		m

### Planta de experimentación 3: Capacidad de carga.

#### Presentación:

en este experimento se realiza un estudio de cargabilidad de una línea para una condición límite de temperatura en el conductor. Se calcula la máxima corriente permisible en la línea que no viola las condiciones de máxima temperatura de conductor permisible a lo largo de un periodo de 24 horas.

Se simula la variación de la temperatura ambiente mediante dos perfiles diferentes (véase figura 9.4), y la variación de exposición solar según el modelo de la IEEE 738. La velocidad del viento se considera constante a lo largo de las 24 horas.

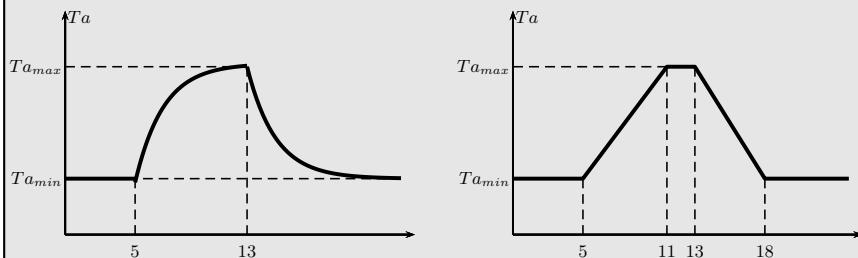


Figura 9.4 Perfiles de temperatura de aire disponibles para el experimento 3

**Instrumentación<sup>4</sup>:** el modelo cuenta con 4 parámetros ajustables organizados en 2 grupos de controles (véase tabla 9.4). Como resultado del experimento, el programa despliega:

- 2 curvas organizadas en 2 gráficos (véase tabla 9.5).
- Una tabla de datos del comportamiento de 3 variables (véase tabla 9.6).

**Experimentos sugeridos:** el siguiente es el listado de experimentos sugeridos:

***Experimento 3.1: Efecto de las condiciones ambientales.***

¿Qué impacto tienen las condiciones ambientales en la variación de la capacidad de carga a lo largo del día? Explore la variabilidad de la capacidad de carga a lo largo del día, para diferentes condiciones ambientales.

***Experimento 3.2: Condición límite.***

¿Cómo afecta la determinación de la máxima temperatura permisible a la capacidad de carga? Explore el impacto de variar el valor de la máxima temperatura permisible sobre los valores máximo y mínimo de la capacidad de carga a lo largo del día.

***Experimento 3.3: Capacidad horaria.***

¿Qué tanto aumentaría la capacidad de carga máxima a lo largo del día, en comparación con la práctica usual de declarar una única capacidad de carga constante para las 24 horas? Determine el aumento máximo de la capacidad de carga de la línea a lo largo del día para diversas condiciones. Reflexione sobre qué condiciones del sistema de potencia deberían cambiarse para poder aprovechar ese margen.

***Experimento 3.4: Capacidad de carga por flecha.***

¿Cómo se calcularía la capacidad de carga en función de una flecha máxima permisible? Adapte el código fuente disponible para desarrollar un modelo que calcule la capacidad de carga de la línea para una flecha máxima permisible.

Tabla 9.4 Parámetros del experimento 3, “Capacidad de carga”

<b>Título:</b>	Capacidad de carga		
<b>Descripción:</b>	En este experimento se realiza un estudio de cargabilidad (capacidad de carga o rating) de una línea para una condición límite de temperatura en el conductor. Se calcula la máxima corriente permisible en la línea que no viola las condiciones de máxima temperatura de conductor permisible a lo largo de un periodo de 24 horas. Se simula la variación de la temperatura ambiente mediante dos perfiles diferentes, y la variación de exposición solar. La velocidad del viento se considera constante a lo largo de las 24 horas.		
<b>Créditos</b>	<b>Implementación</b>	Oscar Germán Duarte Velasco	
<b>Parámetros</b>			
<b>Grupo</b>	<b>Nombre Modelica</b>	<b>Nombre</b>	<b>Descripción</b>
<b>Condiciones ambientales</b>	$T_{Amin}$	Temperatura mínima de aire	
	$T_{Amax}$	Temperatura máxima de aire	
	$V_{vto}$	Velocidad de viento	
<b>Condición límite</b>	$TC_{max}$	Temperatura máxima de conductor	

Tabla 9.5 Figuras del experimento 3, “Capacidad de carga”

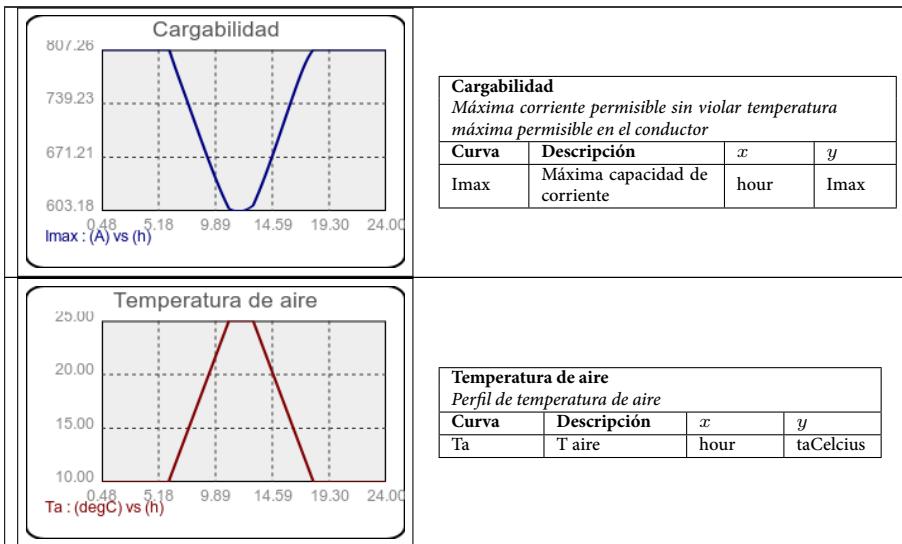


Tabla 9.6 Variables en la tabla de resultados del experimento 3, “Capacidad de carga”

Variable	Descripción	Unidades
hour		h
Imax		A
taCelcius		degC

<sup>4</sup>La información en las tablas siguientes se muestra tal como aparece en la base de datos. Por esta razón hay datos tanto en español como en inglés.

### Planta de experimentación 4: Análisis de flecha.

**Presentación:** en este experimento se muestran los retratos de fase que involucran a la flecha, para evaluar el efecto que tienen la corriente, la temperatura ambiente y la temperatura del conductor sobre la flecha.

Para las simulaciones se han empleado los perfiles de corriente y temperatura de aire que se muestran en la figura 9.5.

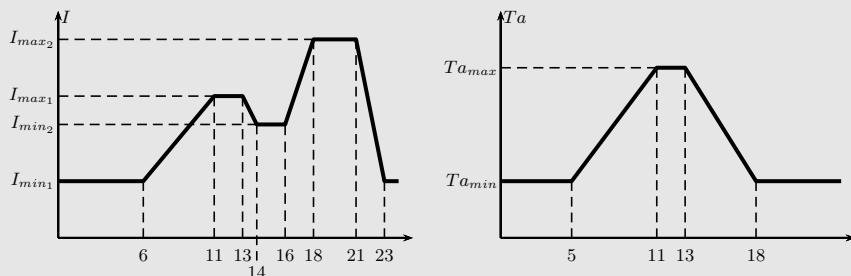


Figura 9.5 Perfiles de corriente y temperatura de aire para el experimento 4

**Instrumentación<sup>5</sup>:** el modelo cuenta con 8 parámetros ajustables organizados en 2 grupos de controles (véase tabla 9.7). Como resultado del experimento, el programa despliega:

- 5 curvas organizadas en 5 gráficos (véase tabla 9.8).
- Una tabla de datos del comportamiento de 5 variables (véase tabla 9.9).

**Experimentos sugeridos:** el siguiente es el listado de experimentos sugeridos:

#### **Experimento 4.1: Flecha vs temperatura de conductor.**

¿Qué influencia tienen las condiciones ambientales y la corriente en la relación entre flecha y la corriente? La relación entre flecha y temperatura de conductor puede aproximarse a una línea recta. Explore cómo se afecta esa recta con diferentes condiciones eléctricas y ambientales.

***Experimento 4.2: Análisis de sensibilidad.***

¿La flecha es más sensible a los cambios de las condiciones eléctricas o a las ambientales? Explore qué tanto cambia la flecha ante modificaciones en las condiciones ambientales y eléctricas.

***Experimento 4.3: Aproximaciones de linea recta.***

¿Es adecuado modelar el comportamiento de la flecha con relaciones afines (líneas rectas)? Efectúe ajustes de línea recta para las relaciones flecha-corriente, flecha-temperatura de aire y flecha-temperatura de conductor, y evalúe la correlación de esos ajustes.

***Experimento 4.4: Efecto del conductor.***

¿Cómo inciden los parámetros del conductor en la relación flecha-temperatura de conductor? Utilice el código fuente del modelo para explorar la sensibilidad del ajuste de línea recta entre flecha y conductor a los parámetros del conductor.

***Experimento 4.5: Efecto del vano.***

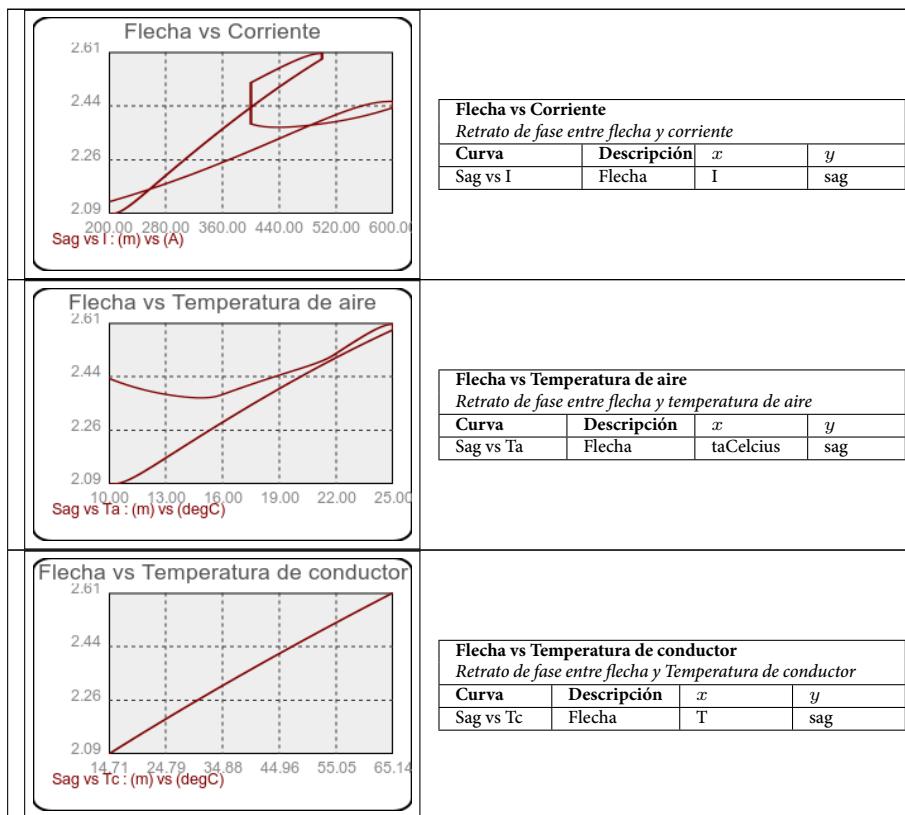
¿Cómo inciden los parámetros del vano en la relación flecha-temperatura de conductor? Utilice el código fuente del modelo para explorar la sensibilidad del ajuste de línea recta entre flecha y conductor a los parámetros del vano.

<sup>5</sup>La información en las tablas siguientes se muestra tal como aparece en la base de datos. Por esta razón hay datos tanto en español como en inglés.

Tabla 9.7 Parámetros del experimento 4, “Análisis de flecha”

Título:	Análisis de flecha		
Descripción:	En este experimento se muestran los retratos de fase que involucran a la flecha, para evaluar el efecto que tienen la corriente, la temperatura ambiente y la temperatura del conductor sobre la flecha		
Créditos	Implementación	Oscar Germán Duarte Velasco	
Parámetros			
Grupo	nombre Modelica	nombre	descripción
Corriente	$I_{min}$	I1	Primer valor
	$I_{max}$	I2	
	$I_{min2}$	I3	
	$I_{max2}$	I4	
Condiciones ambientales	$T_{Amin}$	Temperatura mínima de aire	
	$T_{Amax}$	Temperatura máxima de aire	
	$t_f$	Perfil de temperatura	1.0= trapecio; 2.0=exponencial
	$V_{vto}$	Velocidad de viento	

Tabla 9.8 Figuras del experimento 4, “Análisis de flecha”



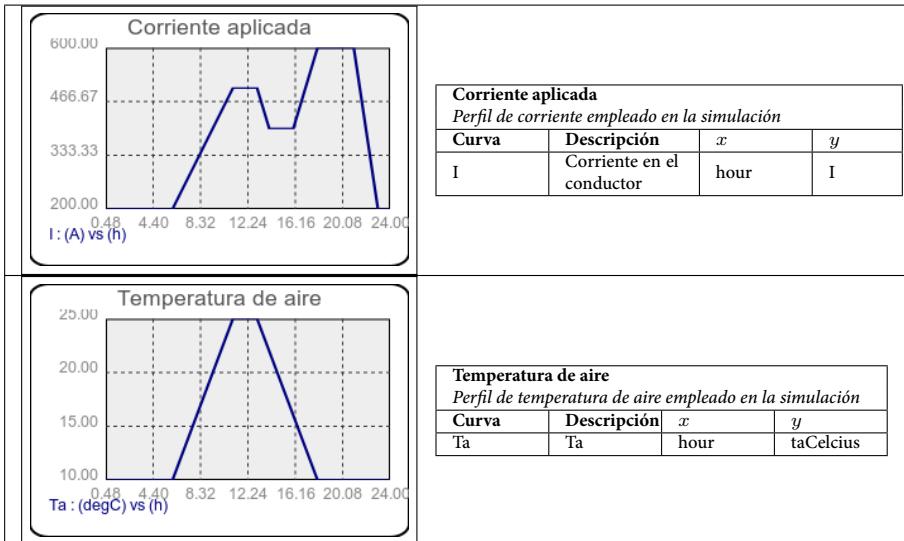


Tabla 9.9 Variables en la tabla de resultados del experimento 4, "Análisis de flecha"

Variable	Descripción	Unidades
I		A
sag		m
taCelcius		degC
T		degC
hour		h

### Planta de experimentación 5: Catenaria.

**Presentación:** la curva descrita por un cable suspendido entre dos soportes es una catenaria. La geometría de esta curva está determinada por el estado de operación (temperatura del conductor y tensiones mecánicas, principalmente). En este experimento se puede analizar el cambio de la geometría de la catenaria ante cambios en las condiciones del vano o del tendido de la línea.

**Instrumentación<sup>6</sup>:** el modelo cuenta con 10 parámetros ajustables organizados en 4 grupos de controles (véase tabla 9.10). Como resultado del experimento, el programa despliega:

- 1 curva organizada en 1 gráfico (véase tabla 9.11).

- Una tabla de datos del comportamiento de 2 variables (véase tabla 9.12).

**Experimentos sugeridos:** el siguiente es el listado de experimentos sugeridos:

***Experimento 5.1: Robustez del algoritmo.***

¿Bajo qué condiciones de simulación el algoritmo arroja valores correctos? Explore diversas combinaciones de parámetros para evaluar la robustez del algoritmo.

***Experimento 5.2: Condiciones de tendido.***

¿Qué efecto tienen las condiciones de tendido en la geometría de la curva? ¿Por qué? Explore los cambios en la geometría debidos a variaciones en las condiciones de tendido.

***Experimento 5.3: Longitudes.***

¿Qué relación debe haber entre la geometría del vano y la longitud de tendido? Explore cómo debe cambiar la longitud de cable requerido, cuando se modifican las separaciones horizontales y verticales de los apoyos.

***Experimento 5.4: Tipos de conductor.***

¿Qué tipo de conductores se elongan más? Compare conductores de capacidad semejante, pero de diferente tipo (por ejemplo ACSR vs AAC).

**Experimento 5.5: Calibre.**

¿Cómo afecta el calibre de un conductor la geometría de la catenaria?  
Compare la geometría de la catenaria correspondiente a conductores de diferente calibre.

Tabla 9.10 Parámetros del experimento 5, “Catenaria”

Título:	Catenaria		
Descripción:	La curva descrita por un cable suspendido entre dos soportes es una catenaria. La geometría de esta curva está determinada por el estado de operación (temperatura del conductor y tensiones mecánicas, principalmente). En este experimento se puede analizar el cambio de la geometría de la catenaria ante cambios en las condiciones del vano o del tendido de la línea		
Créditos	Implementación	Oscar Germán Duarte Velasco	
	e-mail	ogduartev@unal.edu.co	
Parámetros			
Grupo	nombre Modelica	nombre	descripción
Datos del vano	<i>span.Dy</i>	Desnivel	Desnivel de los apoyos en m
	<i>span.S</i>	Vano horizontal	Separación horizontal de los apoyos en m
Condiciones de tendido	<i>span.L_0</i>	Longitud de referencia	Lenght of conductor in state of reference in m
	<i>span.Ten_0</i>	Tensión de referencia	Tension of conductor in state of reference in KgF
	<i>span.T_0</i>	Temperatura de referencia	Temperature of conductor in state of reference in K
Ajustes de simulación	<i>Sag.alpha0</i>	Alpha inicial	
Parámetros del conductor	<i>con.W</i>	Densidad lineal	Linear weight in Kg/m
	<i>con.E</i>	Módulo de elasticidad	Module of elasticity KgF/cm <sup>2</sup>
	<i>con.a</i>	Coeficiente de dilatación	Coefficient of thermal dilatation in 1/K
	<i>con.R_ref</i>	Coeficiente de cambio térmico de resistencia eléctrica	Linear resistance at T_ref in ohm-s/m

<sup>6</sup>La información en las tablas siguientes se muestra tal como aparece en la base de datos. Por esta razón hay datos tanto en español como en inglés.

Tabla 9.11 Figuras del experimento 5, “Catenaria”

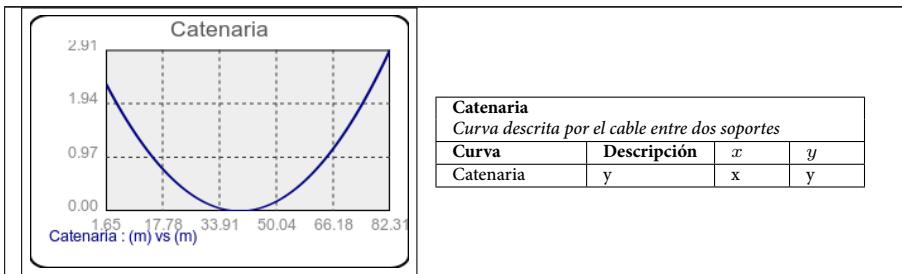


Tabla 9.12 Variables en la tabla de resultados del experimento 5, “Catenaria”

Variable	Descripción	Unidades
x		m
y		m

### 9.3 LA IMPLEMENTACIÓN

Los parámetros del conductor y del vano se almacenan en registros (*record*) separados denominados *ConductorData* y *SpanData*. El día del año y la hora del día se almacenan en un tercer registro cuyo nombre es *TimeData*. Las tablas 9.13 a 9.15 muestran los parámetros de cada registro.

Tabla 9.13 Parámetros en el registro ConductorData

declaración	significado	unidades
Real D	Diámetro externo	mm
Real a	Coeficiente de dilatación	1/K
Real E	Módulo de elasticidad	Kg/cm <sup>2</sup>
Real W	Peso lineal	Kg/m
Real A	Área de sección transversal	cm <sup>2</sup>
Real C	Capacitancia térmica lineal	J/K
Real R_ref	Resistencia eléctrica lineal	Ω/m
Real T_ref	Temperatura de referencia	°C
Real alpha	Pendiente de cambio de resistencia	Ω/K
Real abs =0.5	Absorbida	—
Real emi =1.0	Emisividad	—

Tabla 9.14 Parámetros en el registro SpanData

declaración	significado	unidades
Real He	Altitud sobre el nivel del mar	<i>m</i>
Real L	Latitud	°
Real Zl	Azimut de la línea	°
Real S	Longitud de vano	<i>m</i>
Real Dy	Desnivel de los soportes	<i>m</i>
Real T_0	Temperatura del conductor en el estado de referencia	<i>K</i>
Real Ten_0	Tensión del conductor en el estado de referencia	<i>KgF</i>
Real L_0	Longitud del conductor en el estado de referencia	<i>m</i>

Tabla 9.15 Parámetros en el registro TimeData

declaración	significado
Integer Day	Día del año (1-365)
Real Hour	Hora del día (0-24, 13.5 significa 1:30 pm)

### 9.3.1 Modelo térmico

Se han definido 14 funciones para la implementación del modelo térmico (véase tabla 9.16). También se han diseñado las siguientes clases:

Tabla 9.16 Funciones para el modelo térmico

Función	Calcula:
AirConductivity	Conductividad térmica del aire como función de la temperatura de la película de aire
AirDensity	Densidad del aire como función de la temperatura de la película de aire
AirViscosity	Viscosidad del aire como función de la temperatura de la película de aire
AngleFactor	Factor de corrección para las pérdidas de calor por convección en como función del ángulo entre el viento y el conductor
asinh	$\text{asinh}(x)$
ConvectionFlow	Pérdidas de calor por convección
ForcedConvectionHigh	Pérdidas de calor por convección forzada para velocidades de viento elevadas
ForcedConvectionLow	Pérdidas de calor por convección forzada para velocidades de viento bajas
ForcedConvection	Pérdidas de calor por convección forzada para cualquier velocidades de viento
NaturalConvection	Pérdidas de calor por convección natural
FilmTemperature	Temperatura de película de aire como función de las temperaturas de aire y de conductor
SolarAltitude	Altitud solar como función de la latitud geográfica, el día del año y la hora del día
SolarAzimuth	Altitud solar como función de la latitud geográfica, el día del año y la hora del día
SolarFlux	Ganancia de calor solar como función de la altitud solar, la altitud geográfica, azimut solar, azimut del vano y tipo de atmósfera.

- **ConvectionHeatFlow**: un modelo similar al modelo **HeatTransfer**. Convection de la librería estándar, pero cuyos parámetros dependen de los parámetros del conductor, del vano y del tiempo.
- **SolarHeatFlow**: un modelo similar al modelo **HeatTransfer**. PrescribedHeatFlow de la librería estándar, pero cuyos parámetros dependen de la posición del vano y del sol.
- **StandAloneHeatingResistor**: un modelo similar al modelo **Electrical.Analog.Basic.HeatingResistor** de la librería estándar, que también calcula la ganancia de calor debida al efecto Joule del propio conductor.
- **Conductor**: es una capacitancia térmica con un puerto que da una señal de temperatura.

### 9.3.2 Modelos mecánico y geométrico

Los modelos mecánico y geométrico se implementan mediante cuatro funciones (tabla 9.17) y cuatro clases principales:

Tabla 9.17 Funciones para los modelos mecánico y geométrico

Función	Calcula:
CatenaryLength	Ecuación 9.34
CatenarySag	Ecuación 9.33
CatenarySa	Ecuación 9.29
CatenaryXbar	Ecuación 9.23

- **CatenaryStateChange**: esta clase es la implementación de las ecuaciones de cambio de estado 9.36 y 9.37. (véase archivo **MyStandAlone-Line.mo**). Es quizás la clase más importante del modelo mecánico. En esta clase se establece que el modelo debe satisfacer la nueva condición de longitud dada en 9.34. Para acelerar la solución de las ecuaciones de cambio de estado, puede establecerse un punto de inicio para  $\alpha = H/W$  que ayude al algoritmo. Se sugiere usar  $\alpha = 300$ <sup>7</sup>.
- **Catenary**: es una clase virtual (**partial**) que une los modelos térmico, mecánico y geométrico (véase figura 9.3). Implementa la ecuación

---

<sup>7</sup>A manera de ejemplo, las simulaciones numéricas por defecto del modelo encuentran un valor de  $\alpha = 394,7$  para  $t = 0$ .

9.2 como un *Conductor* (es decir, como una capacitancia térmica), cuyo puerto de calor tiene acoplados los modelos de flujos de calor (efecto Joule  $q_J$ , radiación solar  $q_s$ , convección  $q_c$  y radiación  $q_r$ ). La temperatura del aire  $T_a$  se incluye como una temperatura prescrita.

De otra parte, tiene también un componente de la clase *CatenaryState Change*; la temperatura del Conductor se usa como entrada para el análisis del Cambio de Estado, cuyo principal resultado es el cálculo de la flecha  $D$ . Para usar esta clase, debe diseñarse una clase derivada de tal manera que se establezca el valor de la corriente eléctrica  $I$ . Véase el archivo 9.5.

- *ElectricalCatenary*: es una clase derivada de la clase *Catenary*, en la que una señal de corriente eléctrica se acopla al conductor.
- *StandAloneCatenary*: es una clase derivada de la clase *Catenary*, en la que el valor de la corriente eléctrica es directamente un Real. En el archivo 9.24 se muestra cómo usar esta clase para unas condiciones específicas de simulación.

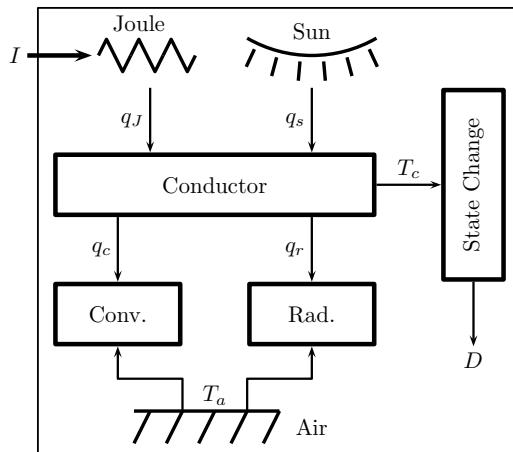


Figura 9.3 Modelo *Catenary*

Los parámetros empleados por defecto corresponden a un caso real de un vano ubicado en una red de transmisión de 230KV en Bogotá, Colombia. Dichos datos se muestran en la tabla 9.18.

Tabla 9.18 Parámetros para la implementación por defecto

<b>Parámetros de vano</b>	
Conductor	Peacock
Altitud	2600 <i>msnm</i>
Latitud	4,779423 <i>Norte</i>
Azimut	76,14
Separación horizontal entre soportes	82,31 <i>m</i>
Diferencia de nivel entre soportes	0,4; <i>m</i>
Tensión longitudinal nominal	2970 <i>Kgf</i>
<b>Parámetros de conductor</b>	
Diámetro externo	24,2 <i>mm</i>
Resistencia lineal a 25°C	$9,7 * 10^{-5}$ <i>ohm/m</i>
Resistencia lineal a 75°C	0,000116 <i>ohm/m</i>
Masa lineal de aluminio	0,79716 <i>Kg/m</i>
Masa lineal de acero	0,31227 <i>Kg/m</i>
Peso lineal	1,16 <i>Kg/m</i>
Módulo de elasticidad nominal	$0,7530 * 10^6$ <i>KG/cm<sup>2</sup></i>
Coeficiente de dilatación nominal	$19,73 * 10^{-6}$ <i>1/C</i>
Área de sección transversal	3,4638 <i>cm<sup>2</sup></i>

### 9.3.3 Listado de archivos

La tabla 9.19 muestra el listado de los archivos fuente de la implementación del modelo.

Tabla 9.19 Archivos del modelo

Número	Archivo
9.1	AirConductivity.mo
9.2	AirDensity.mo
9.3	AirViscosity.mo
9.4	AngleFactor.mo
9.5	Catenary.mo
9.6	CatenaryLenght.mo
9.7	CatenarySa.mo
9.8	CatenarySag.mo
9.9	CatenaryStateChange.mo
9.10	CatenaryX.mo
9.11	CatenaryXbar.mo
9.12	Conductor.mo
9.13	ConductorData.mo
9.14	ConvectionFlow.mo
9.15	ConvectionHeatFlow.mo
9.16	Curve.mo
9.17	DynamicCurve.mo
9.18	ElectricalCatenary.mo
9.19	FilmTemperature.mo
9.20	ForcedConvection.mo
9.21	ForcedConvectionHigh.mo

9.22	ForcedConvectionLow.mo
9.23	MyElectricalLine.mo
9.24	MyStandAloneLine.mo
9.25	NaturalConvection.mo
9.26	Rating.mo
9.27	SagAnalysis.mo
9.28	SolarAltitude.mo
9.29	SolarAzimuth.mo
9.30	SolarFlux.mo
9.31	SolarHeatFlow.mo
9.32	SpanData.mo
9.33	StandAloneCatenary.mo
9.34	StandAloneHeatingResistor.mo
9.35	TimeData.mo
9.36	asinh.mo
9.37	package.mo

Archivo 9.1 AirConductivity.mo

```
within Catenary;

function AirConductivity "IEEE-738 eq 14"
  input Modelica.SIunits.Temp_C Tf "Film Temperature in C";
  output Modelica.SIunits.ThermalConductivity Kf "Thermal conductivity";
algorithm
  Kf:=2.424e-2 + 7.477e-5*Tf -4.407e-9*Tf^2;
end AirConductivity;
```

Archivo 9.2 AirDensity.mo

```
within Catenary;

function AirDensity "IEEE-738 eq 13"
  input Modelica.SIunits.Temp_C Tf "Film temperature in C";
  input Modelica.SIunits.Length He "Altitude in m";
  output Modelica.SIunits.Density rho "Air density in Kg/m^3";
algorithm
  rho:= (1.293 - 1.525e-4*He + 6.379e-9*He^2)/(1+ 0.00367*Tf);
end AirDensity;
```

Archivo 9.3 AirViscosity.mo

```
within Catenary;

function AirViscosity "IEEE-738 eq 12"
  input Modelica.SIunits.Temp_C Tf "Film Temperature in C";
  output Modelica.SIunits.DynamicViscosity mu "viscosity in Pa.s";
algorithm
  mu:=1.4578e-6*(Tf+273)^1.5/(Tf+383.4);
end AirViscosity;
```

## Archivo 9.4 AngleFactor.mo

```

within Catenary;

function AngleFactor "IEEE-738 eq 4a-4b"
  input Modelica.SIunits.Angle phi "angle";
  input Boolean axis "true if angle measured from conductor axis";
  output Real Ka;
protected
  Real Ka1,Ka2;
  Real phi_r;
  constant Real PI=Modelica.Constants.pi;
algorithm
  phi_r:=phi*PI/180;
  Ka1:=1.194 - cos(phi_r) + 0.194*cos(2*phi_r) + 0.368*sin(2*phi_r);
  Ka2:=1.194 - sin(phi_r) - 0.194*cos(2*phi_r) + 0.368*sin(2*phi_r);
  Ka:=if axis then Ka1 else Ka2;
end AngleFactor;

```

## Archivo 9.5 Catenary.mo

```

within Catenary;

partial class Catenary
  parameter ConductorData
    con(D=24.2,a=19.7e-6,E=0.753e6,A=3.4638,W=1.16,C=909.9,R_ref=0.000097,T_ref=273.15+25,alpha=3.8e-7,abs=0.5,emi=1.0);
  parameter SpanData
    span(He=2600,L=4.779420,Zl=76.14,S=82.31,Dy=0.4,T_0=273.15+20,Ten_0=2970,L_0=82.54);
  parameter TimeData
    to(Hour=0,Day=57);
  parameter Real InitialTemp=20;

  // Weather
  Modelica.SIunits.Temp_K ta;
  Modelica.SIunits.Temp_C taCelcius;
  Modelica.SIunits.Velocity windVelocity;
  Modelica.SIunits.Angle windDirection;
  Boolean atm;

  // Thermal
  Conductor
    Wire(C=con.C);
  Modelica.Thermal.HeatTransfer.Sources.PrescribedTemperature Env;
  Modelica.Thermal.HeatTransfer.Components.BodyRadiation Rad(Gr=con.emi*con.D*0.0178/5.6704);
  SolarHeatFlow
    Sun(He=span.He,L=span.L,absorvity=con.abs,Zl=span.Zl,area=con.D/1000,
      Hour=to.Hour,Day=to.Day);
  ConvectionHeatFlow
    Conv(axis=true,D=con.D,He=span.He);

  // Mechanical & geometric

```

```

CatenaryStateChange                                     Sag(a=con.a,E=con.E
    ,A=con.A,W=con.W,S=span.S,Dy=span.Dy,T_0=span.T_0,Ten_0=span.
    Ten_0,L_0=span.L_0);

Modelica.SIunits.Temp_C      T(start=InitialTemp, fixed=false);
Modelica.SIunits.Power      Qj,Qs,Qr,Qc;
Modelica.SIunits.Conversions.NonSIunits.Time_hour hour;
Modelica.SIunits.Length     sag;

equation
hour=time/(60*60);
T=Wire.T-273.15;
Qj=HR.heatPort.Q_flow;
Qc=Conv.Q_flow;
Qs=Sun.Q_flow;
Qr=Rad.Q_flow;
sag=Sag.Sag;

// weather signals to components
Conv.Vw=windVelocity;
Conv.phi=windDirection;
Sun.Atm=atm;
Env.T=ta;
taCelcius=ta-273.15;

// Thermal
connect(HR.heatPort,Wire.port);
connect(Wire.port,Sun.port);
connect(Wire.port,Rad.port_a);
connect(Rad.port_b,Env.port);
connect(Wire.port,Conv.solid);
connect(Conv.fluid,Env.port);

// Mechanical and geometric
// connect(Wire.port_T,Sag.port_T);
Wire.T=Sag.T;
end Catenary;

```

Archivo 9.6 CatenaryLength.mo

```

within Catenary;

function CatenaryLength
  input Modelica.SIunits.Length S;
  input Real alpha;
  input Modelica.SIunits.Length Dy;
  output Modelica.SIunits.Length L;
protected
  Modelica.SIunits.Length Sa,Sb,La,Lb;
algorithm
  Sa:=CatenarySa(S,alpha,Dy);
  Sb:=S-Sa;
  La:=alpha*sinh(Sa/alpha);
  Lb:=alpha*sinh(Sb/alpha);

```

```
L:=La+Lb;
end CatenaryLength;
```

Archivo 9.7 CatenarySa.mo

```
within Catenary;

function CatenarySa
  input Modelica.SIunits.Length S;
  input Real alpha;
  input Modelica.SIunits.Length Dy;
  output Modelica.SIunits.Length Sa;
algorithm
  Sa:= S/2 - alpha*asinh((Dy/2)/(alpha*sinh(S/2/alpha)));
end CatenarySa;
```

Archivo 9.8 CatenarySag.mo

```
within Catenary;

function CatenarySag
  input Modelica.SIunits.Length S;
  input Real alpha;
  input Modelica.SIunits.Length Dy;
  output Modelica.SIunits.Length Sag;
protected
  Modelica.SIunits.Length Xf,Yf,YB,YAB,Sa,Sb;
algorithm
  Sa:=CatenarySa(S,alpha,Dy);
  Sb:=S-Sa;
  Xf:=alpha*asinh(Dy/S);
  Yf:=alpha*cosh(Xf/alpha);
  YB:=alpha*cosh(Sb/alpha);
  YAB:=YB - Dy/S*(Sb-Xf);
  Sag:=YAB-Yf;
end CatenarySag;
```

Archivo 9.9 CatenaryStateChange.mo

```
within Catenary;

model CatenaryStateChange
  parameter Modelica.SIunits.LinearDensity W "Linear Density";
  parameter Modelica.SIunits.Length S "Span";
  parameter Modelica.SIunits.Length Dy "Difference of level";
  parameter Modelica.SIunits.Temp_K T_0;
  parameter Real Ten_0(unit="kgf");
  parameter Modelica.SIunits.Length L_0;
  parameter Modelica.SIunits.LinearExpansionCoefficient a;
  parameter Real E(quantity="ModulusOfElasticity",unit="kg/cm2");
  parameter Modelica.SIunits.Conversions.NonSIunits.Area_cm A;
  parameter Real alpha0=500;
```

```

Modelica.SIunits.Temp_K T;
Modelica.SIunits.Length L, Sag;
Real Ten(unit="kgf");
Real H(unit="kgf");
Real alpha(start=alpha0);
equation
  alpha=abs(H/W);
  Ten=H*cosh(S/2/alpha);
  L=L_0*(1 + a*(T-T_0) + (Ten-Ten_0)/(E*A));
  L=CatenaryLength(S,alpha,Dy);
  Sag=CatenarySag(S,alpha,Dy);
end CatenaryStateChange;

```

Archivo 9.10 CatenaryX.mo

```

within Catenary;

function CatenaryX
  input Modelica.SIunits.Length x_bar;
  input Modelica.SIunits.Length Sa;
  input Real alpha;
  output Modelica.SIunits.Length y;
algorithm
  y:=alpha*cosh((Sa-x_bar)/alpha) - alpha;
end CatenaryX;

```

Archivo 9.11 CatenaryXbar.mo

```

within Catenary;

function CatenaryXbar
  input Modelica.SIunits.Length x_bar;
  input Real alpha;
  output Modelica.SIunits.Length y;
algorithm
  y:=alpha*cosh(x_bar/alpha) - alpha;
end CatenaryXbar;

```

Archivo 9.12 Conductor.mo

```

within Catenary;

class Conductor = Modelica.Thermal.HeatTransfer.Components.
  HeatCapacitor;

```

Archivo 9.13 ConductorData.mo

```

within Catenary;

record ConductorData
public

```

```

parameter Real D(quantity="Length", unit="mm") "External
diameter in mm";
parameter Modelica.SIunits.LinearExpansionCoefficient a " "
Coefficient of thermal dilatation in 1/K;
parameter Real E(quantity="ModulusOfElasticity", unit="kg/cm2")
"Module of elasticity Kgf/cm2";
parameter Modelica.SIunits.LinearDensity W "Linear weight in
Kg/m";
parameter Modelica.SIunits.Conversions.NonSIunits.Area_cm A " "
Cross section area in cm2";
parameter Real C(final quantity="LinearHeatCapacity", final unit="J/
Km") "Linear Heat capacity in J/(mK)";
parameter Modelica.SIunits.Resistance R_ref "Linear resistance at
T_ref in ohms/m";
parameter Modelica.SIunits.Temp_K T_ref "Temperature of reference
for change in resistance in K";
parameter Modelica.SIunits.LinearTemperatureCoefficient alpha " "
Change of resistance per K";
parameter Modelica.SIunits.SpectralAbsorptionFactor abs=0.5 " "
Absorvity of the wire";
parameter Modelica.SIunits.Emissivity emi=1.0 "Emissivity of the
wire";
end ConductorData;

```

Archivo 9.14 ConvectionFlow.mo

```

within Catenary;

function ConvectionFlow
  input Real D(quantity="Length", unit="mm") "External diameter in mm";
  input Modelica.SIunits.Length He "Altitude in m";
  input Modelica.SIunits.Velocity Vw "Wind velocity in m/s";
  input Modelica.SIunits.Temp_C Tc "Temperature of conductor";
  input Modelica.SIunits.Temp_C Ta "Temperature of air";
  input Modelica.SIunits.Angle phi "angle";
  input Boolean axis "true if angle measured from conductor axis";
  output Modelica.SIunits.Power Qc "flow of heat";
protected
  Modelica.SIunits.Power Qcf "Forced convection";
  Modelica.SIunits.Power Qcn "Natural convection";
algorithm
  Qcf:=ForcedConvection(D,He,Vw,Tc,Ta,phi,axis);
  Qcn:=NaturalConvection(D,He,Tc,Ta);
  Qc:= if Qcf>Qcn then Qcf else Qcn;
end ConvectionFlow;

```

Archivo 9.15 ConvectionHeatFlow.mo

```

within Catenary;

class ConvectionHeatFlow
  input Real D(quantity="Length", unit="mm") "External diameter in mm";
  input Modelica.SIunits.Length He "Altitude in m";

```

```

parameter Boolean axis "true if angle measured from conductor axis";
input Modelica.SIunits.Velocity Vw "Wind velocity in m/s";
input Modelica.SIunits.Angle phi "angle";
input Modelica.SIunits.Temp_C Tc "Temperature of conductor";
input Modelica.SIunits.Temp_C Ta "Temperature of air";
Modelica.SIunits.Power Q_flow "Heat flow in W";
Modelica.Thermal.HeatTransfer.Interfaces.HeatPort_a solid;
Modelica.Thermal.HeatTransfer.Interfaces.HeatPort_b fluid;
equation
Tc=solid.T;
Ta=fluid.T;
Q_flow=ConvectionFlow(D,He,Vw,Tc,Ta,phi, axis);
solid.Q_flow=Q_flow;
fluid.Q_flow=-Q_flow;
end ConvectionHeatFlow;

```

Archivo 9.16 Curve.mo

```

within Catenary;

model Curve
extends StandAloneCatenary;
parameter Modelica.SIunits.Temp_C TCmax=75;
Modelica.Blocks.Sources.Constant temperature_source(k=273.15+
TCmax);
Modelica.Blocks.Sources.Constant windVel(k=0.61);
Modelica.Blocks.Sources.Constant windDir(k=0);
Modelica.Blocks.Sources.Constant airTemp(k=273.15+20);
Modelica.Blocks.Sources.BooleanConstant atmos(k=true);
Modelica.SIunits.Length x(start=0);
Modelica.SIunits.Length y;
Real alpha;
Modelica.SIunits.Length Sa;
equation
der(x)=span.S;
alpha=Sag.H/con.W;
Sa=CatenarySa(span.S, alpha, span.Dy);
y=CatenaryX(x, Sa, alpha);
Wire.T=temperature_source.y;
windVelocity=windVel.y;
windDirection=windDir.y;
atm=atmos.y;
ta=airTemp.y;
end Curve;

```

Archivo 9.17 DynamicCurve.mo

```

within Catenary;

model DynamicCurve
parameter Integer Xsteps=20;
parameter Modelica.SIunits.Current Imax=500;
parameter Modelica.SIunits.Current Imin=200;

```

```

parameter Modelica.SIunits.Current Imin2=400;
parameter Modelica.SIunits.Current Imax2=600;
parameter Integer IFlag=2;
parameter Modelica.SIunits.Current mitable [:,2]={
  {0,Imin},
  {60*60*6,Imin},
  {60*60*11,Imax},
  {60*60*13,Imax},
  {60*60*14,Imin2},
  {60*60*16,Imin2},
  {60*60*18,Imax2},
  {60*60*21,Imax2},
  {60*60*23,Imin},
  {60*60*24,Imin}};
extends StandAloneCatenary;
Modelica.Blocks.Sources.Constant      current_source1(k=Imax);
Modelica.Blocks.Sources.TimeTable     current_source2(table=mitable);
Modelica.Blocks.Sources.Constant      windVel(k=0.61);
Modelica.Blocks.Sources.Constant      windDir(k=0);
Modelica.Blocks.Sources.Constant      airTemp(k=273.15+20);
Modelica.Blocks.Sources.BooleanConstant atmos(k=true);
Modelica.SIunits.Length x[Xsteps+1]=0:(span.S/Xsteps):span.S;
Modelica.SIunits.Length y[Xsteps+1];
// falta calcular la longitud, posición y rotación de las líneas rectas
// para dibujar la catenaria
Modelica.SIunits.Length length[Xsteps];
Modelica.SIunits.Length traslationY[Xsteps];
Modelica.SIunits.Length rotation[Xsteps];
Real alpha;
Modelica.SIunits.Length Sa;
equation
  alpha=Sag.H/con.W;
  Sa=CatenarySa(span.S,alpha,span.Dy);
  y=CatenaryX(x,Sa,alpha);
  if IFlag==1 then
    I=current_source1.y;
  elseif IFlag==2 then
    I=current_source2.y;
  end if;
  windVelocity=windVel.y;
  windDirection=windDir.y;
  atm=atmos.y;
  ta=airTemp.y;
algorithm
  length:=Length(x,y,Xsteps);
  rotation:=Rotation(x,y,Xsteps);
  traslationY:=TraslationY(y,Xsteps);
end DynamicCurve;

```

## Archivo 9.18 ElectricalCatenary.mo

```

within Catenary;

class ElectricalCatenary
  extends Catenary;
  Modelica.Electrical.Analog.Basic.HeatingResistor HR(R_ref=con.R_ref,
    T_ref=con.T_ref, alpha=con.alpha);
  Modelica.Electrical.Analog.Sources.SignalCurrent Current;
  Modelica.Electrical.Analog.Basic.Ground G;
equation
  connect(Current.p,HR.p);
  connect(Current.n,HR.n);
  connect(Current.n,G.p);
end ElectricalCatenary;

```

## Archivo 9.19 FilmTemperature.mo

```

within Catenary;

function FilmTemperature "IEEE-738 eq 6"
  input Modelica.SIunits.Temp_C Tc "Conductor Temperature in C";
  input Modelica.SIunits.Temp_C Ta "Air Temperature in C";
  output Modelica.SIunits.Temp_C Tf "Film Temperature in C";
algorithm
  Tf:=0.5*(Tc+Ta);
end FilmTemperature;

```

## Archivo 9.20 ForcedConvection.mo

```

within Catenary;

function ForcedConvection
  input Real D(quantity="Length", unit="mm") "External diameter in mm";
  input Modelica.SIunits.Length He "Altitude in m";
  input Modelica.SIunits.Velocity Vw "Wind velocity in m/s";
  input Modelica.SIunits.Temp_C Tc "Temperature of conductor";
  input Modelica.SIunits.Temp_C Ta "Temperature of air";
  input Modelica.SIunits.Angle phi "angle";
  input Boolean axis "true if angle measured from conductor axis";
  output Modelica.SIunits.Power Qc "flow of forced heat";
protected
  Real Qc1 "Forced convection for low winds";
  Real Qc2 "Force convection for high winds";
algorithm
  Qc1:=ForcedConvectionLow(D,He,Vw,Tc,Ta,phi,axis);
  Qc2:=ForcedConvectionHigh(D,He,Vw,Tc,Ta,phi,axis);
  Qc:= if Qc1>Qc2 then Qc1 else Qc2;
end ForcedConvection;

```

## Archivo 9.21 ForcedConvectionHigh.mo

```

within Catenary;

function ForcedConvectionHigh
  input Real D(quantity="Length", unit="mm") "External diameter in mm";
  input Modelica.SIunits.Length He "Altitude in m";
  input Modelica.SIunits.Velocity Vw "Wind velocity in m/s";
  input Modelica.SIunits.Temp_C Tc "Temperature of conductor";
  input Modelica.SIunits.Temp_C Ta "Temperature of air";
  input Modelica.SIunits.Angle phi "angle";
  input Boolean axis "true if angle measured from conductor axis";
  output Modelica.SIunits.Power Qc2 "flow of forced heat";
protected
  Real Tf=0 "Temperature of film";
  Real rho_f;
  Real mu_f;
  Real kf;
  Real Kang;
algorithm
  Tf:=FilmTemperature(Tc,Ta);
  rho_f:=AirDensity(Tf,He);
  mu_f:=AirViscosity(Tf);
  kf:=AirConductivity(Tf);
  Kang:=AngleFactor(phi, axis);
  Qc2:=(0.0119*(D*rho_f*Vw/mu_f)^0.6)*kf*Kang*(Tc-Ta);
end ForcedConvectionHigh;

```

## Archivo 9.22 ForcedConvectionLow.mo

```

within Catenary;

function ForcedConvectionLow
  input Real D(quantity="Length", unit="mm") "External diameter in mm";
  input Modelica.SIunits.Length He "Altitude in m";
  input Modelica.SIunits.Velocity Vw "Wind velocity in m/s";
  input Modelica.SIunits.Temp_C Tc "Temperature of conductor";
  input Modelica.SIunits.Temp_C Ta "Temperature of air";
  input Modelica.SIunits.Angle phi "angle";
  input Boolean axis "true if angle measured from conductor axis";
  output Modelica.SIunits.Power Qc1 "flow of forced heat";
protected
  Modelica.SIunits.Temp_C Tf=0 "Temperature of film";
  Modelica.SIunits.Density rho_f=0 "density of air";
  Modelica.SIunits.DynamicViscosity mu_f=0 "viscosity of air";
  Modelica.SIunits.ThermalConductivity kf=0 "thermal conductivity of
    air";
  Real Kang=0 "angle factor";
algorithm
  Tf:=FilmTemperature(Tc,Ta);
  rho_f:=AirDensity(Tf,He);
  mu_f:=AirViscosity(Tf);
  kf:=AirConductivity(Tf);
  Kang:=AngleFactor(phi, axis);

```

```

Qc1:=(1.01 + 0.0372*(D* rho_f*Vw/ mu_f)^0.52)*kf*Kang*(Tc-Ta);
end ForcedConvectionLow;

```

Archivo 9.23 MyElectricalLine.mo

```

within Catenary;

model MyElectricalLine
  extends ElectricalCatenary;

  Modelica.Blocks.Sources.Sine      current(offset=400,amplitude
    =100,freqHz=1/(4*60*60));
  Modelica.Blocks.Sources.Constant   windVel(k=0.61);
  Modelica.Blocks.Sources.Constant   windDir(k=0);
  Modelica.Blocks.Sources.Constant   airTemp(k=273.15+20);
  Modelica.Blocks.Sources.BooleanConstant atmos(k=true);

  equation
    connect( Current.i , current.y );
    windVelocity=windVel.y;
    windDirection=windDir.y;
    atm=atmos.y;
    ta=airTemp.y;
  end MyElectricalLine;

```

Archivo 9.24 MyStandAloneLine.mo

```

within Catenary;

model MyStandAloneLine
  parameter Modelica.SIunits.Current Imax=500;
  parameter Modelica.SIunits.Current Imin=200;
  parameter Modelica.SIunits.Current Imin2=400;
  parameter Modelica.SIunits.Current Imax2=600;
  parameter Modelica.SIunits.Current currentTable [:,2]={
    {0,Imin},
    {60*60*6,Imin},
    {60*60*11,Imax},
    {60*60*13,Imax},
    {60*60*14,Imin2},
    {60*60*16,Imin2},
    {60*60*18,Imax2},
    {60*60*21,Imax2},
    {60*60*23,Imin},
    {60*60*24,Imin}};
  parameter Modelica.SIunits.Temp_C Tmax=25;
  parameter Modelica.SIunits.Temp_C Tmin=5;
  parameter Modelica.SIunits.Temp_C airtempTable [:,2]={
    {0,273.15+Tmin},
    {60*60*5,273.15+Tmin},
    {60*60*11,273.15+Tmax},
    {60*60*13,273.15+Tmax},
    {60*60*18,273.15+Tmin},
    
```

```

{60*60*24,273.15+Tmin}};
extends StandAloneCatenary(I(start=Imin));
Modelica.Blocks.Sources.TimeTable      current_source(table=
currentTable);
Modelica.Blocks.Sources.Constant      windVel(k=0.61);
Modelica.Blocks.Sources.Constant      windDir(k=0);
// Modelica.Blocks.Sources.Constant    airTemp(k=273.15+20);
Modelica.Blocks.Sources.TimeTable      airTemp(table=airtempTable);
Modelica.Blocks.Sources.BooleanConstant atmos(k=true);
equation
I=current_source.y;
windVelocity=windVel.y;
windDirection=windDir.y;
atm=atmos.y;
ta=airTemp.y;
end MyStandAloneLine;

```

Archivo 9.25 NaturalConvection.mo

```

within Catenary;

function NaturalConvection
  input Real D(quantity="Length", unit="mm") "External diameter in mm";
  input Modelica.SIunits.Length He "Altitude in m";
  input Modelica.SIunits.Temp_C Tc "Temperature of conductor";
  input Modelica.SIunits.Temp_C Ta "Temperature of air";
  output Modelica.SIunits.Power Qcn=0 "Natural convection flow";
protected
  Modelica.SIunits.Temp_C Tf=0 "Temperature of film";
  Modelica.SIunits.Temp_C rho_f=0 "density of air";
  Modelica.SIunits.ThermalConductivity kf=0 "thermal conductivity of
air";
algorithm
  Tf:=FilmTemperature(Tc,Ta);
  rho_f:=AirDensity(Tf,He);
  kf:=AirConductivity(Tf);
  Qcn:=0.0205*rho_f^0.5*D^0.75*(Tc-Ta)^1.25;
end NaturalConvection;

```

Archivo 9.26 Rating.mo

```

within Catenary;

model Rating
  parameter Modelica.SIunits.Temp_C TAmin=10;
  parameter Modelica.SIunits.Temp_C TAmax=25;
  parameter Modelica.SIunits.Temp_C TCmax=75;
  parameter Modelica.SIunits.Velocity Vvto=0.61;
  Real TAFlag(start=1.0);
  extends StandAloneCatenary;
  Modelica.Blocks.Sources.Constant      temperature_source(k=273.15+
TCmax);                                // Ex. of Temperature of
conductor known

```

```

Modelica.Blocks.Sources.Constant      windVel(k=Vvto);
Modelica.Blocks.Sources.Constant      windDir(k=0);
Modelica.Blocks.Sources.Trapezoid     airTemp1(offset=273.15+Tamin
    , amplitude=Tmax-Tamin, startTime=60*60*6, rising=60*60*5, width
    =60*60*2, falling=60*60*5, period=60*60*24);
Modelica.Blocks.Sources.Exponentials airTemp3(offset=273.15+Tamin
    , outMax=Tmax-Tamin, startTime=60*60*6, riseTime=60*60*7,
    riseTimeConst=60*60*2, fallTimeConst=60*60*3);
Modelica.Blocks.Sources.BooleanConstant atmos(k=true);
Modelica.SIunits.Current Imax;
equation
  der(TAFlag)=0;
  Wire.T=temperature_source.y;
  windVelocity=windVel.y;
  windDirection=windDir.y;
  atm=atmos.y;
  if TAFlag<=1.5 then
    ta = airTemp1.y;
  else
    ta = airTemp3.y;
  end if;
algorithm
  Imax:=sqrt(Qj/HR.R);
end Rating;

```

Archivo 9.27 SagAnalysis.mo

```

within Catenary;

model SagAnalysis
  parameter Modelica.SIunits.Temp_C Tamin=10;
  parameter Modelica.SIunits.Temp_C Tmax=25;
  parameter Modelica.SIunits.Current Imax=500;
  parameter Modelica.SIunits.Current Imin=200;
  parameter Modelica.SIunits.Current Imin2=400;
  parameter Modelica.SIunits.Current Imax2=600;
  parameter Modelica.SIunits.Velocity Vvto=0.61;
  parameter Real tf=1.0;
  Real TAFlag(start=tf);
  parameter Modelica.SIunits.Current mitable [:,2]={
    {0,Imin},
    {60*60*6,Imin},
    {60*60*11,Imax},
    {60*60*13,Imax},
    {60*60*14,Imin2},
    {60*60*16,Imin2},
    {60*60*18,Imax2},
    {60*60*21,Imax2},
    {60*60*23,Imin},
    {60*60*24,Imin}};
  extends StandAloneCatenary(T(start=60));
  Modelica.Blocks.Sources.TimeTable current_source(table=mitable);
  Modelica.Blocks.Sources.Constant windVel(k=Vvto);
  Modelica.Blocks.Sources.Constant windDir(k=0);

```

```

Modelica.Blocks.Sources.Trapezoid      airTemp1(offset=273.15+Tamin
    , amplitude=TAmax-TAmin, startTime=60*60*6, rising=60*60*5, width
    =60*60*2, falling=60*60*5, period=60*60*24);
Modelica.Blocks.Sources.Exponentials   airTemp3(offset=273.15+Tamin
    , outMax=TAmax-TAmin, startTime=60*60*6, riseTime=60*60*7,
    riseTimeConst=60*60*2, fallTimeConst=60*60*3);
Modelica.Blocks.Sources.BooleanConstant atmos(k=true);

equation
  der(TAFlag)=0;
  I=current_source.y;
  windVelocity=windVel.y;
  windDirection=windDir.y;
  atm=atmos.y;
  if TAFlag<=1.5 then
    ta = airTemp1.y;
  else
    ta = airTemp3.y;
  end if;
end SagAnalysis;

```

Archivo 9.28 SolarAltitude.mo

```

within Catenary;

function SolarAltitude "IEEE-738 eq 15a 15b"
  input Modelica.SIunits.Conversions.NonSIunits.Angle_deg L "Latitude";
  input Modelica.SIunits.Conversions.NonSIunits.Time_day N "Day of
    the year";
  input Modelica.SIunits.Conversions.NonSIunits.Time_hour Hour "Hour
    of the day";
  output Modelica.SIunits.Conversions.NonSIunits.Angle_deg Hc "
    Altitude of the sun";
protected
  Modelica.SIunits.Angle dec "declination in rad";
  Modelica.SIunits.Angle lat;
  Modelica.SIunits.Angle w;
  constant Real PI=Modelica.Constants.pi;
algorithm
  dec:=23.4583* sin((284+N)*2*PI/365)*PI/180;
  lat:=L*PI/180.0;
  w:=(Hour-12.0)*15*PI/180.0;
  Hc:=asin( cos(lat)*cos(dec)*cos(w) + sin(lat)*sin(dec))*180/PI;
end SolarAltitude;

```

Archivo 9.29 SolarAzimuth.mo

```

within Catenary;

function SolarAzimuth "IEEE-738 eq 16a 16b"
  input Modelica.SIunits.Conversions.NonSIunits.Angle_deg L "Latitude";
  input Modelica.SIunits.Conversions.NonSIunits.Time_day N "Day of
    the year";

```

```



```

Archivo 9.30 SolarFlux.mo

```

within Catenary;

function SolarFlux


```

## Archivo 9.31 SolarHeatFlow.mo

```

within Catenary;

class SolarHeatFlow
  parameter Modelica.SIunits.Length He "Altitude above sea level in m";
  parameter Modelica.SIunits.Conversions.NonSIunits.Angle_deg L "
    Latitude";
  parameter Modelica.SIunits.Conversions.NonSIunits.Time_day Day "
    Initial day of the year(1-365)";
  parameter Modelica.SIunits.Conversions.NonSIunits.Time_hour Hour "
    Initial hour (0-24 13.5 means 1:30pm)";
  parameter Modelica.SIunits.SpectralAbsorptionFactor absorvity=0.5 "
    Absorvity of the wire";
  parameter Modelica.SIunits.Conversions.NonSIunits.Angle_deg Zl "
    Azimuth of the line";
  parameter Modelica.SIunits.Area area           "Proyected area of wire
  ";
  Boolean Atm          "Atmosphere: true for clear, false for industrial
  ";
  Real day(start=Day)      "Initial day of the year (1-365)";
  Real t(start=Hour)       "Hour of the day (0-24 13.5 means 1:30pm)
  ";
  Modelica.SIunits.Power Q_flow "Heat flow in W";
  Modelica.Thermal.HeatTransfer.Interfaces.HeatPort_a port;
equation
  day=pre(day);
  der(t)=1/3600;
  Q_flow=SolarFlux(SolarAltitude(L,Day,t),Atm,He,SolarAzimuth(L,Day,t),
    Zl)*absorvity*area;
  when t>24 then
    reinit(t,0);
    reinit(day,pre(day)+1);
  end when;
  port.Q_flow=-Q_flow;
end SolarHeatFlow;

```

## Archivo 9.32 SpanData.mo

```

within Catenary;

record SpanData
public
  parameter Modelica.SIunits.Length He "Altitude above sea level in m";
  parameter Modelica.SIunits.Conversions.NonSIunits.Angle_deg L "
    Latitude in deg";
  parameter Modelica.SIunits.Conversions.NonSIunits.Angle_deg Zl "
    Azimuth of the line in deg";
  parameter Modelica.SIunits.Length S "Longitud del vano en m";
  parameter Modelica.SIunits.Length Dy "Desnivel de los apoyos en m";
  parameter Modelica.SIunits.Temp_K T_0        "Temperature of conductor
  in state of reference in K";
  parameter Real Ten_0(unit="kgf")      "Tension of conductor in state of
  reference in KgF";

```

```

parameter Modelica.SIunits.Length L_0          "Length of conductor in
      state of reference in m";
end SpanData;

```

Archivo 9.33 StandAloneCatenary.mo

```

within Catenary;

partial class StandAloneCatenary
  extends Catenary;
  // Electrical
  StandAloneHeatingResistor HR(R_ref=con.R_ref,T_ref=con.T_ref,alpha=
    con.alpha);
  Modelica.SIunits.Current I;
equation
  // Electrical
  HR.i=I;
end StandAloneCatenary;

```

Archivo 9.34 StandAloneHeatingResistor.mo

```

within Catenary;

class StandAloneHeatingResistor
  parameter Modelica.SIunits.Resistance R_ref=1 "Resistance at
    temperature T_ref";
  parameter Modelica.SIunits.Temp_C T_ref=300 "Reference temperature";
  parameter Modelica.SIunits.LinearTemperatureCoefficient alpha=0 "
    Temperature coefficient of resistance";
  Modelica.SIunits.Resistance R "Resistance = R_ref*(1 + alpha*(heatPort.T - T_ref));";
  Modelica.SIunits.Current i "Current flowing into the pin";
  Modelica.Thermal.HeatTransfer.Interfaces.HeatPort_a heatPort ;
equation
  R = R_ref*(1+alpha*(heatPort.T-T_ref));
  heatPort.Q_flow = -i*i*R;
end StandAloneHeatingResistor;

```

Archivo 9.35 TimeData.mo

```

within Catenary;

record TimeData
public
  parameter Modelica.SIunits.Conversions.NonSIunits.Time_day Day   "Day
    of the year (1-365)";
  parameter Modelica.SIunits.Conversions.NonSIunits.Time_hour Hour
    "Hour of the day (0-24, 13.5 means 1:30pm)";
end TimeData;

```

Archivo 9.36 asinh.mo

```
within Catenary;

function asinh
  input Real x;
  output Real y;
  external "C" y=asinh(x);
end asinh;
```

Archivo 9.37 package.mo

```
package Catenary
  import Modelica.SIunits.Conversions.*;
  import SI = Modelica.SIunits;
  import NonSI = Modelica.SIunits.Conversions.NonSIunits;

end Catenary;
```

# 10

## EVOLUCIÓN DE ENFERMEDADES

*El modelo representa la evolución de enfermedades infecciosas. Las poblaciones de las especies involucradas se representan por compartimentos. Se modela el tamaño de la población de cada compartimento y las interacciones entre estos.*

La evolución de las enfermedades infecciosas es uno de los temas de investigación de la epidemiología. Los modelos matemáticos de estos fenómenos permiten realizar experimentos computacionales que ayudan a analizar la propagación de las enfermedades y, eventualmente, realizar predicciones.

Los modelos deben representar los mecanismos de infección y recuperación de la enfermedad, así como el nacimiento y muerte de individuos. En ocasiones, en estos procesos intervienen poblaciones de diferentes especies, o subpoblaciones de una misma especie.



Figura 10.1 *Aedes aegypti*. Insecto transmisor del Dengue<sup>1</sup>

<sup>1</sup>Foto tomada de <http://www.saludputumayo.gov.co>.

Se presenta aquí una implementación que permite ensamblar modelos relativamente sofisticados sin la necesidad de escribir explícitamente el conjunto completo de ecuaciones diferenciales y algebraicas. Los primeros modelos presentados tienen pocos compartimentos (2 o 3) y, posteriormente, se desarrollan casos más complejos (hasta 19 compartimentos).

## 10.1 EL MODELO

Los modelos aquí explicados han sido compilados e implementados en [Cam12]. La estrategia básica consiste en identificar las especies involucradas en el proceso de propagación de la enfermedad y realizar una *partición*<sup>2</sup>. de su población. Cada uno de los subconjuntos de la partición es un *compartimento*. Las poblaciones de los compartimentos evolucionan con el tiempo debido a fenómenos tales como nacimientos, muertes, infecciones y recuperaciones de la enfermedad.

Para cada compartimento se plantea una ecuación dinámica que describe la rata de cambio del tamaño de la población  $x$ . La forma general de la ecuación es:

$$\dot{x} = \sum \text{entradas} - \sum \text{salidas} \quad (10.1)$$

en donde  $\sum \text{entradas}$  representa el ingreso de individuos al compartimento y  $\sum \text{salidas}$ , el egreso. Es usual que cada uno de los términos que aparecen en las sumatorias dependa del tamaño de la población de uno o más de los compartimentos del modelo.

En los modelos que se presentan a continuación hay dos especies involucradas:

- La especie que sufre la enfermedad, que será identificada como la especie de humanos.
- La especie que sirve como transmisora de la enfermedad (como *vector* de la enfermedad) que será identificada como la especie de mosquitos.

### 10.1.1 Modelo SIR

En este modelo solo se considera la especie de humanos. Su población se subdivide en tres compartimentos, que dan origen al nombre del modelo:

---

<sup>2</sup>La partición de un conjunto es la definición de subconjuntos de dicho conjunto con dos propiedades: 1) los subconjuntos son disyuntos, es decir no tienen elementos en común o, lo que es igual, su intersección es vacía, y 2) la unión de los subconjuntos es igual al conjunto original

- **S** -Susceptibles: individuos que no han contraído la enfermedad pero que pueden llegar a contraerla.
- **I** -Infectados: individuos que han contraído la enfermedad y aún no la han superado.
- **R** -Recuperados: individuos que se han superado la enfermedad.

Los tamaños de las poblaciones de estos compartimentos se representan por  $S$ ,  $I$ , y  $R$ , respectivamente. En el modelo *SIR*, las dinámicas de las poblaciones se rigen por las siguientes reglas:

- I. Un individuo puede pasar del compartimento **S** al **I** a través del *contagio*.
- II. Un individuo puede pasar del compartimento **I** al **R** a través de la *recuperación*.
- III. Los individuos recuperados adquieren inmunidad y, por tanto, no vuelven a enfermarse.
- IV. La población total de la especie es constante.
- V. La dinámica de la enfermedad es mucho más rápida que la dinámica natural de la especie y, por tanto, no se modelan los nacimientos ni las muertes.

El contagio se da por el contacto entre un individuo susceptible y uno infectado. Por esta razón, el número de contagios es directamente proporcional al producto  $SI$  (proporcional a la probabilidad de encuentro entre un individuo infectado y uno susceptible), con constante de proporcionalidad  $\beta$ . El número de recuperaciones es proporcional al número de infectados  $I$ ; la tasa de recuperación  $\gamma$  es el inverso del tiempo de recuperación  $t_r = 1/\gamma$ . Esta dinámica se representa gráficamente en la figura 10.2. La ecuación 10.2 muestra las relaciones matemáticas del modelo:

Ecuaciones del modelo SIR

$$\begin{aligned}\dot{S} &= -\beta SI \\ \dot{I} &= \beta SI - \gamma I \\ \dot{R} &= \gamma I\end{aligned}$$

(10.2)

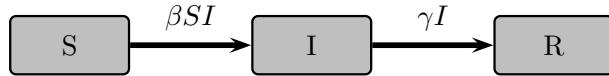


Figura 10.2 Diagrama del modelo SIR

### 10.1.2 Modelo SIR con nacimientos y muertes

El modelo *SIR* puede modificarse para incluir el fenómeno de muerte natural en cada compartimento. La muerte natural se modela como proporcional a la población de cada compartimento, con una constante de proporcionalidad  $\mu$  igual para todos los compartimentos. Para compensar la disminución de la población, se consideran los nacimientos, que igualan en cantidad a las muertes.

La figura 10.3 muestra el diagrama correspondiente al modelo. Las ecuaciones correspondientes son las siguientes:

#### Ecuaciones del modelo SIR con nacimientos y muertes

$$\begin{aligned}\dot{S} &= \mu N - \beta SI - \mu S \\ \dot{I} &= \beta SI - \gamma I - \mu I \\ \dot{R} &= \gamma I - \mu I \\ N &= S + I + R\end{aligned}\tag{10.3}$$

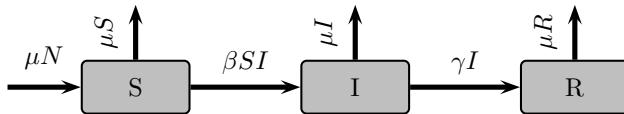


Figura 10.3 Diagrama del modelo SIR con nacimientos y muertes

### 10.1.3 Modelo SIS

En algunas enfermedades la recuperación no asegura inmunidad. En otras palabras, es posible adquirir la enfermedad más de una vez. Como se muestra en la figura 10.4, el modelo *SIS* (sin nacimientos ni muertes) solo tiene dos compartimentos: *S* e *I*; los individuos que se recuperan pueden enfermarse de nuevo, y por tanto pasan al compartimento de los susceptibles. Las ecuaciones del modelo son las siguientes:

Ecuaciones del modelo SIS

$$\begin{aligned}\dot{S} &= -\beta SI + \gamma I \\ \dot{I} &= \beta SI - \gamma I\end{aligned}\tag{10.4}$$

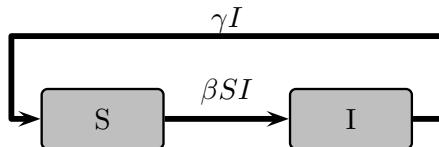


Figura 10.4 Diagrama del modelo SI

#### 10.1.4 Modelo SIR-SI

En este modelo se considera también la dinámica de la especie transmisora de la enfermedad (la especie *vector*). Para distinguir las dos especies se utilizarán los subíndices  $h$  y  $v$  para referirnos a las poblaciones de humanos y vectores, respectivamente. La figura 10.5 ilustra el modelo, cuyas ecuaciones se presentan en 10.5 y se explican a continuación:

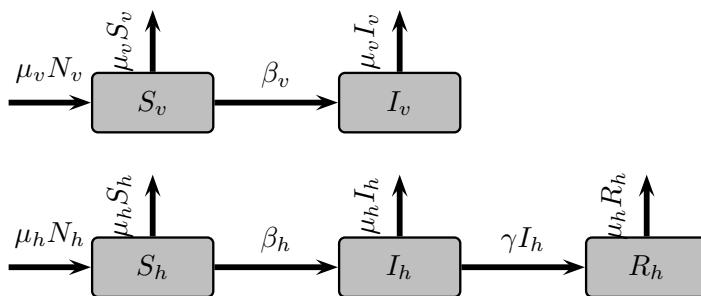


Figura 10.5 Diagrama del modelo SIR-SI

##### 10.1.4.1 Modelo del vector

El vector se representa por dos compartimentos, uno **S** y uno **I**. Los mosquitos infectados son los que servirán de transmisores de la enfermedad. La probabilidad de que un mosquito se infecte es proporcional a la probabilidad de encuentro entre un mosquito no infectado y un humano infectado, es decir,

es proporcional al producto  $S_v I_h / N_h$ . Hay dos elementos más que intervienen: la tasa de picadura (representada por  $b$ ) y la probabilidad de que el mosquito resulte infectado (representada por  $p_v$ ).

#### 10.1.4.2 Modelo del humano

La población humana se representa por tres compartimentos, uno **S**, uno **I** y uno **R**. La probabilidad de que un humano se infecte es proporcional a la probabilidad de encuentro entre un mosquito infectado y un humano susceptible, es decir, es proporcional al producto  $S_h I_v / N_v$ . También intervienen: la tasa de picadura (representada por  $b$ ) y la probabilidad de que el humano resulte infectado (representada por  $p_h$ ).

Ecuaciones del modelo SIR-SI	
Vector	Humano
$\dot{S}_v = \mu N_v - \beta_v - \mu_v S_v$	$\dot{S}_h = \mu N_h - \beta_h - \mu_h S_h$
$\dot{I}_v = \beta_v - \mu_v I_v$	$\dot{I}_h = \beta_h - \gamma I_h - \mu_h I_h$
$N_v = S_v + I_v$	$\dot{R}_h = \gamma I_h - \mu_h R_h$
$\beta_v = p_v b S_v I_h / N_h$	$N_h = S_h + I_h + R_h$
	$\beta_h = p_h b S_v I_v / N_v$

#### 10.1.5 Modelo SIR-SI con dos serotipos

Este modelo contempla dos variantes de la enfermedad, caracterizada por dos serotipos. Los serotipos se identifican por los subíndices 1 y 2. El número de compartimentos se incrementa ahora así:

- La población de vectores infectados ( $I_v$ ) se subdivide en dos compartimentos:
  - $I_{v_1}$ : los vectores infectados por el serotipo 1.
  - $I_{v_2}$ : los vectores infectados por el serotipo 2.
- La población de humanos infectados ( $I_h$ ) se subdivide en cuatro compartimentos:
  - $I_{h_1}$ : los humanos infectados por el serotipo 1.
  - $I_{h_2}$ : los humanos infectados por el serotipo 2.
  - $I_{h_{21}}$ : los humanos infectados por el serotipo 1 que previamente se habían infectado (y recuperado) con el serotipo 2.

- $I_{h_{12}}$ : los humanos infectados por el serotipo 2 que previamente se habían infectado (y recuperado) con el serotipo 1.
- La población de humanos recuperados ( $R_h$ ) se subdivide en tres compartimentos:
  - $R_{h_1}$ : los humanos recuperados después de haberse infectado (únicamente) con el serotipo 1.
  - $R_{h_2}$ : los humanos recuperados después de haberse infectado (únicamente) con el serotipo 2.
  - $R_h$ : los humanos recuperados después de haberse infectado con ambos serotipos.

El modelo se representa gráficamente en la figura 10.6. La ecuación 10.6 consigna el conjunto de expresiones matemáticas del modelo:

Ecuaciones del modelo SIR-SI con dos serotipos	
<b>Vector</b>	
$\dot{S}_v$	$= \mu N_v - \beta_{v_1} - \beta_{v_2} - \mu_v S_v$
$\dot{I}_{v_1}$	$= \beta_{v_1} - \mu_v I_{v_1}$
$\dot{I}_{v_2}$	$= \beta_{v_2} - \mu_v I_{v_2}$
$N_v$	$= S_v + I_v$
$\beta_{v_1}$	$= p_v b S_v (I_{h_1} + I_{h_{21}}) / N_h$
$\beta_{v_2}$	$= p_v b S_v (I_{h_2} + I_{h_{12}}) / N_h$
<b>Humano</b>	
$\dot{S}_h$	$= \mu N_h - \beta_{h_1} - \beta_{h_2} - \mu_h S_h$
$\dot{R}_h$	$= \gamma I_{h_{12}} + \gamma I_{h_{21}} - \mu_h R_h$
$N_h$	$= S_h + I_h + R_h$
<b>Serotipo 1</b>	
$\dot{I}_{h_1}$	$= \beta_{h_1} - \gamma I_{h_1} - \mu_h I_{h_1}$
$\dot{R}_{h_1}$	$= \gamma I_{h_1} - \mu_h R_{h_1} - \beta_{h_2} R_{h_1}$
$\dot{I}_{h_{12}}$	$= \beta_{h_2} R_{h_1} - \gamma I_{h_{12}} - \mu_h I_{h_{12}}$
$\beta_{h_1}$	$= p_h b S_v I_{v_1} / N_v$
<b>Serotipo 2</b>	
$\dot{I}_{h_2}$	$= \beta_{h_2} - \gamma I_{h_2} - \mu_h I_{h_2}$
$\dot{R}_{h_2}$	$= \gamma I_{h_2} - \mu_h R_{h_2} - \beta_{h_1} R_{h_2}$
$\dot{I}_{h_{21}}$	$= \beta_{h_1} R_{h_2} - \gamma I_{h_{21}} - \mu_h I_{h_{21}}$
$\beta_{h_2}$	$= p_h b S_v I_{v_2} / N_v$

(10.6)

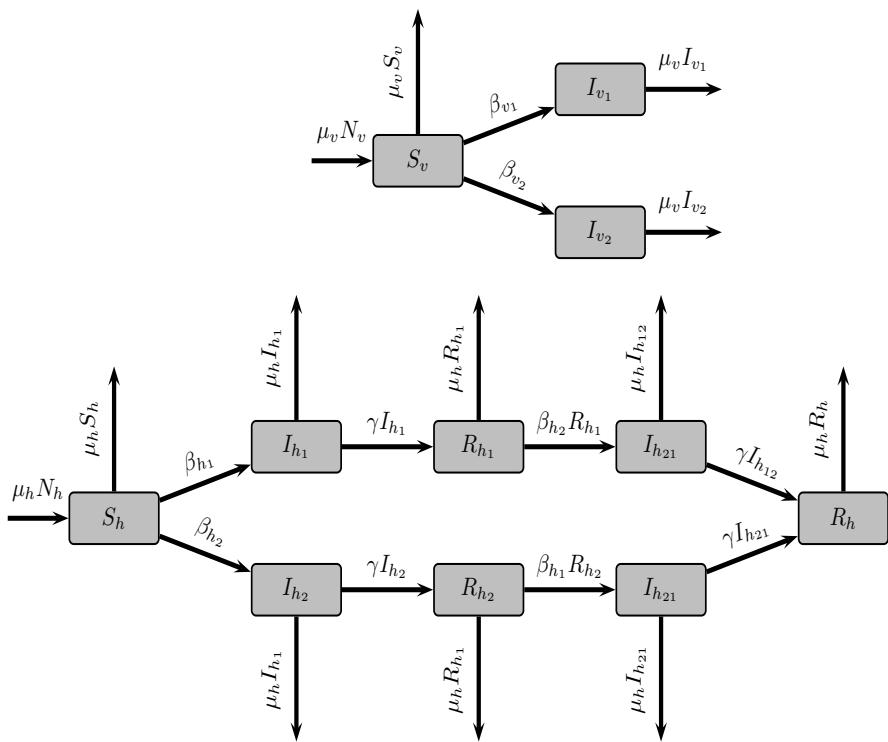


Figura 10.6 Diagrama del modelo SIR-SI con dos serotipos de virus

### 10.1.6 Modelo del dengue

En [Cam12] se propone un modelo para la propagación del dengue, y se identifican los parámetros correspondientes para el caso colombiano. El modelo separa la población de humanos en dos poblaciones diferentes:

- La población de jóvenes, identificada con el subíndice  $j$ .
- La población de adultos, identificada con el subíndice  $a$ .

Para cada una de estas subpoblaciones se formula un modelo SIR-SI con dos serotipos, como el presentado en la sección 10.1.5. Sobre ese modelo, se adiciona la migración de cada compartimiento de la población joven a su homólogo de la población adulta, a una tasa  $\alpha$  que representa la rapidez del paso de joven a adulto.

En esas condiciones, el diagrama resultante es el que se muestra en la figura 10.8. La ecuación 10.7 consigna el conjunto de relaciones matemáticas correspondientes.

Ecuaciones del modelo para el dengue	
Vector	
$\dot{S}_v = \mu N_v - \beta_{v1} - \beta_{v2} - \mu_v S_v$	
$\dot{I}_{v1} = \beta_{v1} - \mu_v I_{v1}$	
$\dot{I}_{v2} = \beta_{v2} - \mu_v I_{v2}$	
$N_v = S_v + I_v$	
$\beta_{v1} = p_v b S_v (I_{j1} + I_{j21} + I_{a1} + I_{a21}) / N_h$	
$\beta_{v2} = p_v b S_v (I_{j2} + I_{j12} + I_{a2} + I_{a12}) / N_h$	
Humano	
Joven	Adulto
$\dot{S}_j = \mu N_j - \beta_{j1} - \beta_{j2} - \mu_j S_j - \alpha S_j$	$\dot{S}_a = \mu N_a - \beta_{a1} - \beta_{a2} - \mu_a S_a + \alpha S_j$
$\dot{R}_j = \gamma I_{j12} + \gamma I_{j21} - \mu_j R_j - \alpha R_j$	$\dot{R}_a = \gamma I_{a12} + \gamma I_{a21} - \mu_a R_a + \alpha R_j$
$N_j = S_j + I_j + R_j$	$N_a = S_a + I_a + R_a$
Joven-Serotipo 1	Joven-Serotipo 2
$\dot{I}_{j1} = \beta_{j1} - \gamma I_{j1} - \mu_j I_{j1} - \alpha I_{j1}$	$\dot{I}_{j2} = \beta_{j2} - \gamma I_{j2} - \mu_j I_{j2} - \alpha I_{j2}$
$\dot{R}_{j1} = \gamma I_{j1} - \mu_j R_{j1} - \beta_{j2} R_{j1} - \alpha R_{j1}$	$\dot{R}_{j2} = \gamma I_{j2} - \mu_j R_{j2} - \beta_{j1} R_{j2} - \alpha R_{j2}$
$\dot{I}_{j12} = \beta_{j2} R_{j1} - \gamma I_{j12} - \mu_j I_{j12} - \alpha I_{j12}$	$\dot{I}_{j21} = \beta_{j1} R_{j2} - \gamma I_{j21} - \mu_j I_{j21} - \alpha I_{j21}$
$\beta_{j1} = p_j b S_v I_{v1} / N_v$	$\beta_{j2} = p_j b S_v I_{v2} / N_v$
Adulto-Serotipo 1	Adulto-Serotipo 2
$\dot{I}_{a1} = \beta_{a1} - \gamma I_{a1} - \mu_a I_{a1} + \alpha I_{j1}$	$\dot{I}_{a2} = \beta_{a2} - \gamma I_{a2} - \mu_a I_{a2} + \alpha I_{j2}$
$\dot{R}_{a1} = \gamma I_{a1} - \mu_a R_{a1} - \beta_{a2} R_{a1} + \alpha R_{j1}$	$\dot{R}_{a2} = \gamma I_{a2} - \mu_a R_{a2} - \beta_{a1} R_{a2} + \alpha R_{j2}$
$\dot{I}_{a12} = \beta_{a2} R_{a1} - \gamma I_{a12} - \mu_a I_{a12} + \alpha I_{j12}$	$\dot{I}_{a21} = \beta_{a1} R_{a2} - \gamma I_{a21} - \mu_a I_{a21} + \alpha I_{j21}$
$\beta_{a1} = p_a b S_v I_{v1} / N_v$	$\beta_{a2} = p_a b S_v I_{v2} / N_v$

(10.7)

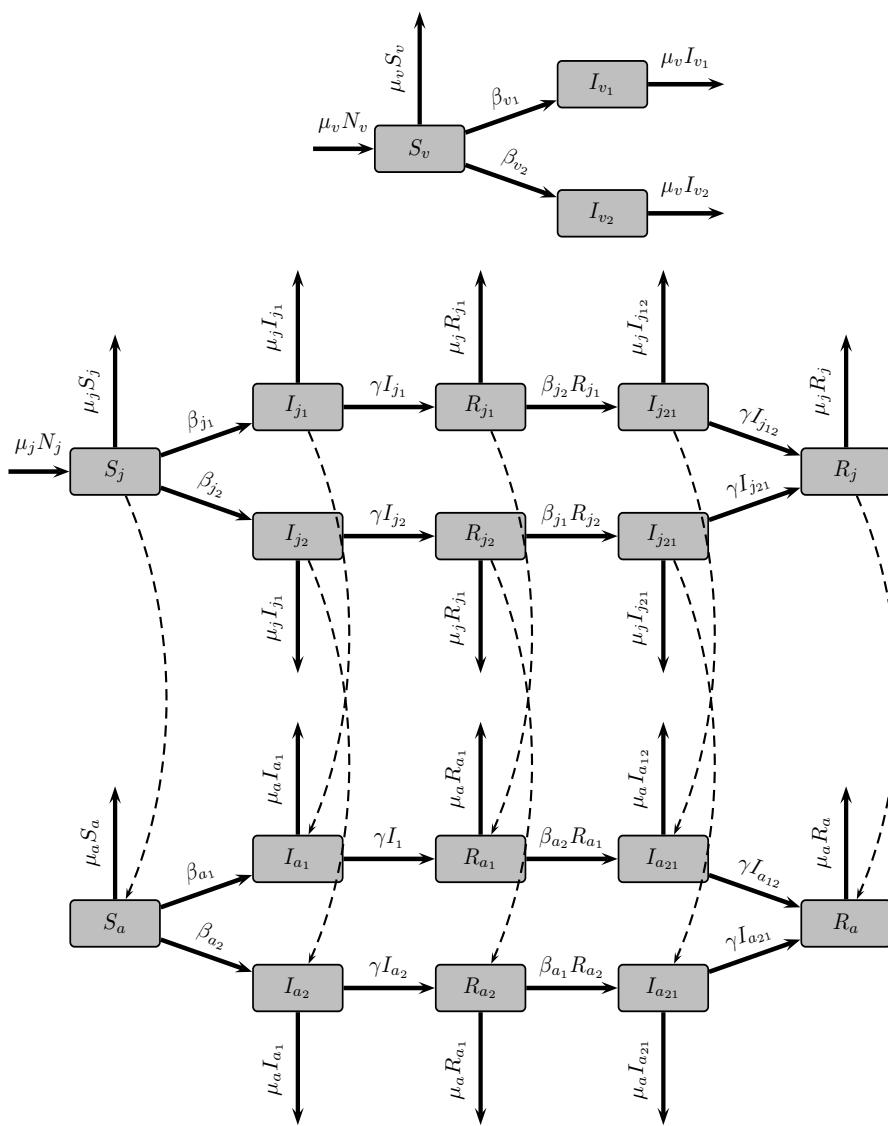


Figura 10.8 Diagrama del dengue modelo SIR-SI con dos serotipos de virus y población de humanos separad por edades. Las líneas punteadas representan la migración de las poblaciones jóvenes a adultas,  $\alpha X_j$ , donde  $X_j$  es la población del compartimento joven

## 10.2 PLANTAS DE EXPERIMENTACIÓN Y EXPERIMENTOS SUGERIDOS

### Planta de experimentación 6: Modelo Susceptible-Infectado-Recuperado.

**Presentación:** modelo de propagación de una enfermedad con tres compartimentos de individuos: susceptibles, infectados y recuperados. El modelo se presenta en la sección 10.1.1.

**Instrumentación<sup>3</sup>:** el modelo cuenta con 4 parámetros ajustables organizados en 2 grupos de controles (véase tabla 10.1). Como resultado del experimento, el programa despliega:

- 6 curvas organizadas en 4 gráficos (véase tabla 10.2).
- Una tabla de datos del comportamiento de 4 variables (véase tabla 10.3).

**Experimentos sugeridos:** el siguiente es el listado de experimentos sugeridos:

#### *Experimento 6.1: Estado estacionario.*

¿Qué factores inciden en los tamaños de las poblaciones cuando se estabiliza el comportamiento? Explore cómo afectan los parámetros del modelo a los valores finales de las poblaciones susceptibles, infectadas y recuperadas.

#### *Experimento 6.2: Tiempos de respuesta.*

¿Qué factores inciden en la rapidez con que se estabiliza el comportamiento? Explore cómo afectan los parámetros del modelo a los tiempos en que se alcanzan los valores finales de las poblaciones susceptibles, infectadas y recuperadas.

***Experimento 6.3: Infección máxima.***

¿Qué combinación de factores causa la máxima infección? Explore cómo afectan los parámetros del modelo al valor máximo (valor pico) de la población de infectados.

***Experimento 6.4: Tiempo de infección máxima.***

¿De qué depende que el pico de infección suceda antes o después? Explore cómo afectan los parámetros del modelo al tiempo en que sucede el pico de la población de infectados.

Tabla 10.1 Parámetros del experimento 6, “Modelo Susceptible-Infectado-Recuperado”

<b>Título:</b>	Modelo SIR		
<b>Descripción:</b>	Modelo de propagación de una enfermedad con tres compartimentos de individuos: Susceptibles, Infectados y Recuperados		
<b>Créditos</b>	<b>Implementación</b>	Oscar Germán Duarte Velasco	
	<b>e-mail</b>	ogduartev@unal.edu.co	
<b>Parámetros</b>			
<b>Grupo</b>	<b>nombre Modelica</b>	<b>nombre</b>	<b>descripción</b>
<b>Enfermedad</b>	<i>beta</i>	Tasa de contagio	Probabilidad de contagio cuando hay un contacto
	<i>trec</i>	Tiempo de recuperación	Número de días que tarda en recuperarse un infectado
<b>Población</b>	<i>NI</i>	Infectados iniciales	Número de individuos infectados al inicio de la simulación
	<i>NR</i>	Recuperados iniciales	Número de individuos recuperados al inicio de la simulación

<sup>3</sup>La información en las tablas siguientes se muestra tal como aparece en la base de datos. Por esta razón hay datos tanto en español como en inglés.

Tabla 10.2 Figuras del experimento 6, "Modelo Susceptible-Infectado-Recuperado"

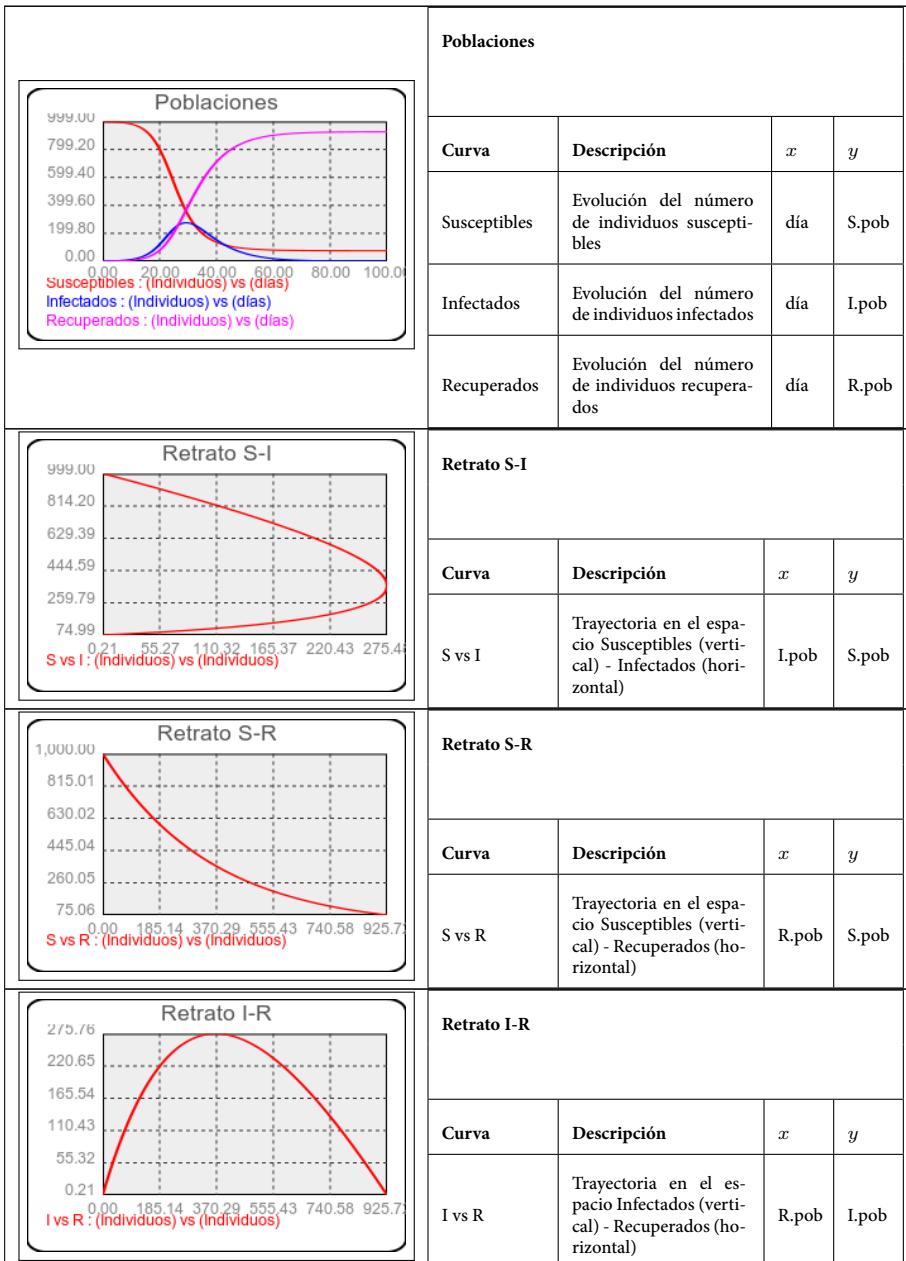


Tabla 10.3 Variables en la tabla de resultados del experimento 6, “Modelo Susceptible-Infectado-Recuperado”

Variable	Descripción	Unidades
Día		Días
S.pob		Individuos
I.pob		Individuos
R.pob		Individuos

### Planta de experimentación 7: Modelo Susceptible-Infectado-Recuperado con nacimientos.

**Presentación:** modelo de propagación de una enfermedad con tres compartimentos de individuos: susceptibles, infectados y recuperados. Incluye muertes y nacimientos. El modelo se presenta en la sección 10.1.2.

**Instrumentación<sup>4</sup>:** el modelo cuenta con 5 parámetros ajustables organizados en 2 grupos de controles (véase tabla 10.4). Como resultado del experimento, el programa despliega:

- 7 curvas organizadas en 5 gráficos (véase tabla 10.5).
- Una tabla de datos del comportamiento de 4 variables (véase tabla 10.6).

**Experimentos sugeridos:** el siguiente es el listado de experimentos sugeridos:

#### *Experimento 7.1: Frecuencia.*

¿De qué factores depende la frecuencia de aparición de la infección? Explore cómo afectan los parámetros del modelo el tiempo entre picos de infección.

#### *Experimento 7.2: Linealización S-R.*

Obtenga un ajuste de línea recta para la relación entre el número de susceptibles y el número de recuperados. Explore cómo se afectan los parámetros de la recta ajustada con los diferentes parámetros del modelo.

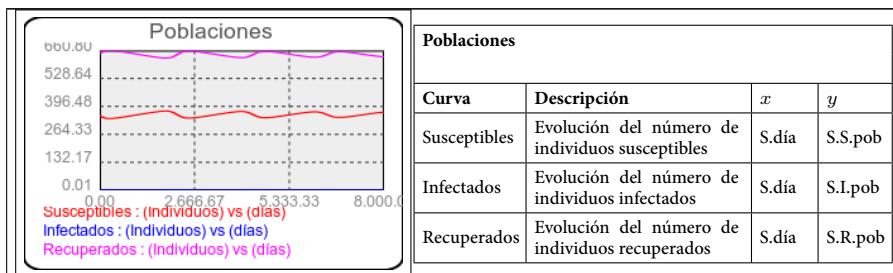
### Experimento 7.3: Infección mínima.

¿Qué factores inciden en los valores mínimos de infección? Explore cómo afectan los parámetros del modelo el tamaño de la población de infectados más bajo entre dos picos sucesivos.

Tabla 10.4 Parámetros del experimento 7, “Modelo Susceptible-Infectado-Recuperado con nacimientos”

Título:	Modelo SIR con nacimientos		
Descripción:	Modelo de propagación de una enfermedad con tres compartimentos de individuos: Susceptibles, Infectados y Recuperados. Incluye muertes y nacimientos.		
Créditos	Implementación		
	e-mail		
<b>Parámetros</b>			
Grupo	nombre Modelica	nombre	descripción
Enfermedad	<i>S.beta</i>	Tasa de contagio	Probabilidad de contagio cuando hay un contacto
	<i>S.trec</i>	Tiempo de recuperación	Número de días que tarda en recuperarse un infectado
	<i>mu</i>	Tasa de nacimientos (*e-6)	Tasa de natalidad y muerte natural
Poblaciones	<i>S.NI</i>	Infectados iniciales	Número de individuos infectados al inicio de la simulación
	<i>S.NR</i>	Recuperados iniciales	Número de individuos recuperados al inicio de la simulación

Tabla 10.5 Figuras del experimento 7, “Modelo Susceptible-Infectado-Recuperado con nacimientos”



<sup>4</sup>La información en las tablas siguientes se muestra tal como aparece en la base de datos. Por esta razón hay datos tanto en español como en inglés.

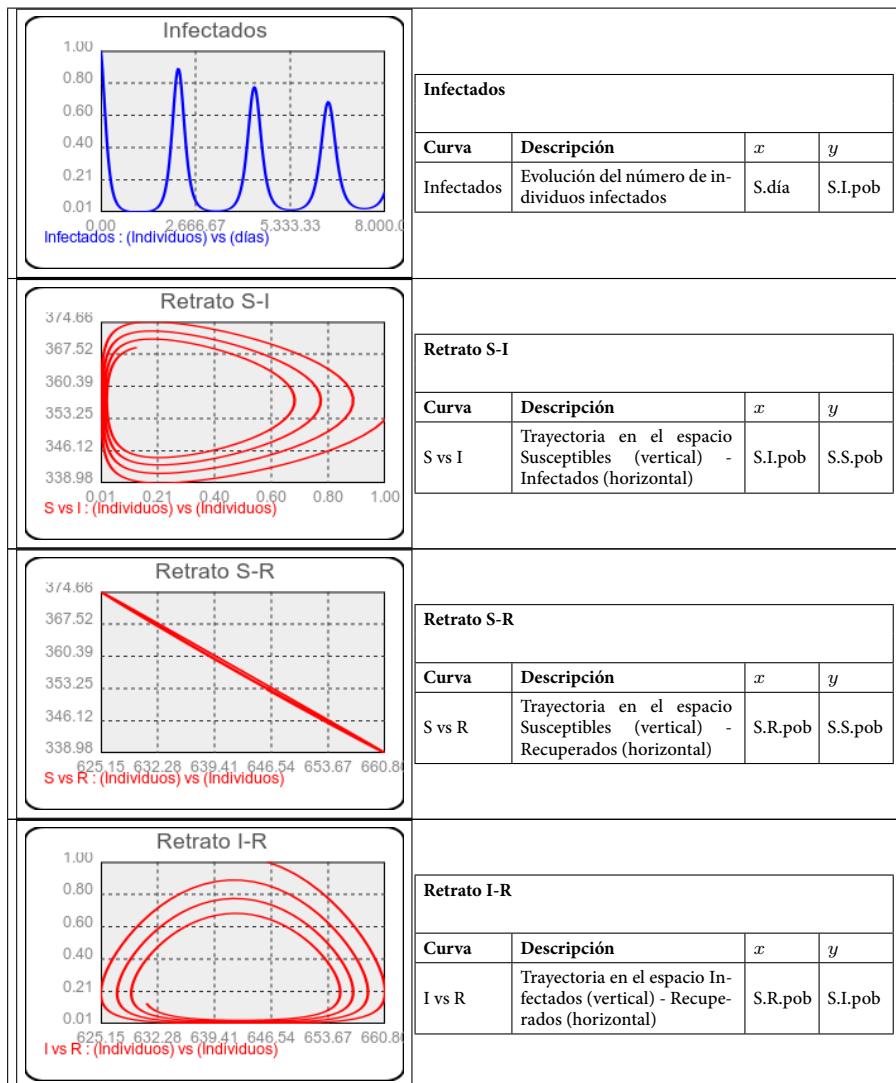


Tabla 10.6 Variables en la tabla de resultados del experimento 7, “Modelo Susceptible-Infectado-Recuperado con nacimientos”

Variable	Descripción	Unidades
S.día		Días
S.S.pob		Individuos

S.I.pob		Individuos
S.R.pob		Individuos

### Planta de experimentación 8: Modelo Susceptible-Infectado-Susceptible.

**Presentación:** modelo de propagación de una enfermedad con dos compartimentos de individuos: susceptibles e infectados. Los individuos que se recuperan pueden volver a infectarse, y por tanto se convierten en susceptibles. El modelo se presenta en la sección 10.1.3.

**Instrumentación<sup>5</sup>:** el modelo cuenta con 3 parámetros ajustables organizados en 2 grupos de controles (véase tabla 10.7). Como resultado del experimento, el programa despliega:

- 2 curvas organizadas en 1 gráfico (véase tabla 10.8).
- Una tabla de datos del comportamiento de 3 variables (véase tabla 10.9).

**Experimentos sugeridos:** el siguiente es el listado de experimentos sugeridos:

#### ***Experimento 8.1: Estado estacionario.***

¿Qué factores inciden en los tamaños de las poblaciones cuando se estabiliza el comportamiento? Explore cómo afectan los parámetros del modelo a los valores finales de las poblaciones susceptibles, infectadas y recuperadas.

#### ***Experimento 8.2: Tiempos de respuesta.***

¿Qué factores inciden en la rapidez con que se estabiliza el comportamiento? Explore cómo afectan los parámetros del modelo a los tiempos en que se alcanzan los valores finales de las poblaciones susceptibles, infectadas y recuperadas.

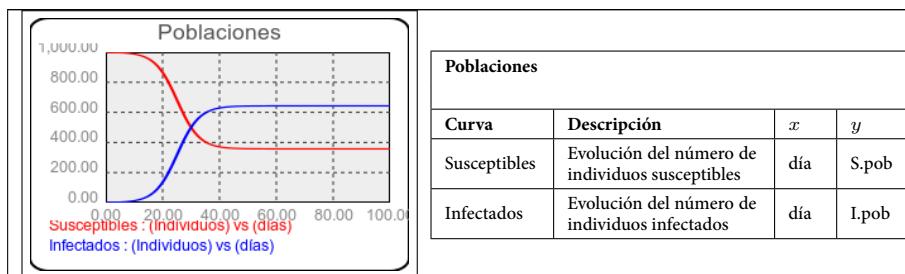
### **Experimento 8.3: Nacimientos y muertes.**

¿Cómo se comportaría el modelo si se incluyen nacimientos y muertes? Modifique el modelo para incorporar los fenómenos de nacimiento y muerte. Formule las ecuaciones que describen el nuevo modelo y compárelas con las del modelo original.

Tabla 10.7 Parámetros del experimento 8, “Modelo Susceptible-Infectado-Susceptible”

Título:	Modelo SIS		
Descripción:	Modelo de propagación de una enfermedad con dos compartimentos de individuos: Susceptibles, Infectados. Los individuos que se recuperan pueden volver a infectarse, y por tanto se convierten en susceptibles.		
Créditos	Implementación		
Parámetros			
Grupo	Nombre Modelica	Nombre	Descripción
Enfermedad	<i>beta</i>	Tasa de contagio	Probabilidad de contagio cuando hay un contacto
	<i>trec</i>	Tiempo de recuperación	Número de días que tarda en recuperarse un infectado
Poblaciones	<i>NI</i>	Infectados iniciales	Número de individuos infectados al inicio de la simulación

Tabla 10.8 Figuras del experimento 8, “Modelo Susceptible-Infectado-Susceptible”



<sup>5</sup>La información en las tablas siguientes se muestra tal como aparece en la base de datos. Por esta razón hay datos tanto en español como en inglés.

Tabla 10.9 Variables en la tabla de resultados del experimento 8, “Modelo Susceptible-Infectado-Susceptible”

Variable	Descripción	Unidades
Día		Días
S.pob		Individuos
I.pob		Individuos

### Planta de experimentación 9: Modelo SIR para humanos y SI con dos cepas para mosquito.

**Presentación:** modelo de propagación de una enfermedad en el que hay dos cepas del virus. Los humanos recuperados de la infección de una cepa pueden reinfecarse con la otra. El modelo se presenta en la sección 10.1.5.

**Instrumentación:** el modelo cuenta con 11 parámetros ajustables organizados en 2 grupos de controles (véase tabla 10.10). Como resultado del experimento, el programa despliega:

- 11 curvas organizadas en 4 gráficos (véase tabla 10.11).
- Una tabla de datos del comportamiento de 12 variables (véase tabla 10.12).

**Experimentos sugeridos:** el siguiente es el listado de experimentos sugeridos:

#### *Experimento 9.1: Tiempos de vida.*

¿Qué incidencia tienen los tiempos de vida media de humanos y mosquitos en el comportamiento de la epidemia? Explore cómo afecta la modificación de los tiempos de vida de humanos y mosquitos en los diferentes aspectos del comportamiento de la epidemia.

***Experimento 9.2: Poblaciones iniciales.***

¿Qué efecto tiene el tamaño de las poblaciones iniciales infectadas de humanos y mosquitos en la evolución de la epidemia? Explore cómo afecta el tamaño de las poblaciones iniciales infectadas de humanos y mosquitos los diferentes aspectos del comportamiento de la epidemia.

***Experimento 9.3: Infección máxima.***

¿Qué condiciones favorecen la propagación de la infección? Explore los factores del modelo que hacen que el número de infectados sea mayor, para cada uno de los dos serotipos.

***Experimento 9.4: Entrada del serotipo 2.***

¿Qué efecto tiene la entrada del serotipo 2 sobre la infección del serotipo 1? Explore cómo incide la aparición del segundo serotipo en la propagación de la infección del primero. Para ello, puede modificar el día de entrada del segundo serotipo

***Experimento 9.5: Modelo del dengue.***

Construya un modelo del dengue El archivo dengue.mo que se encuentra con el código fuente de la planta de experimentación contiene un modelo del dengue (Camargo G. “Modelamiento de la dinámica del dengue en Colombia” Tesis de Maestría. Universidad Nacional de Colombia, 2012). Utilice ese modelo para explorar la sensibilidad del modelo a los diferentes parámetros

**Experimento 9.6: Una única cepa.**

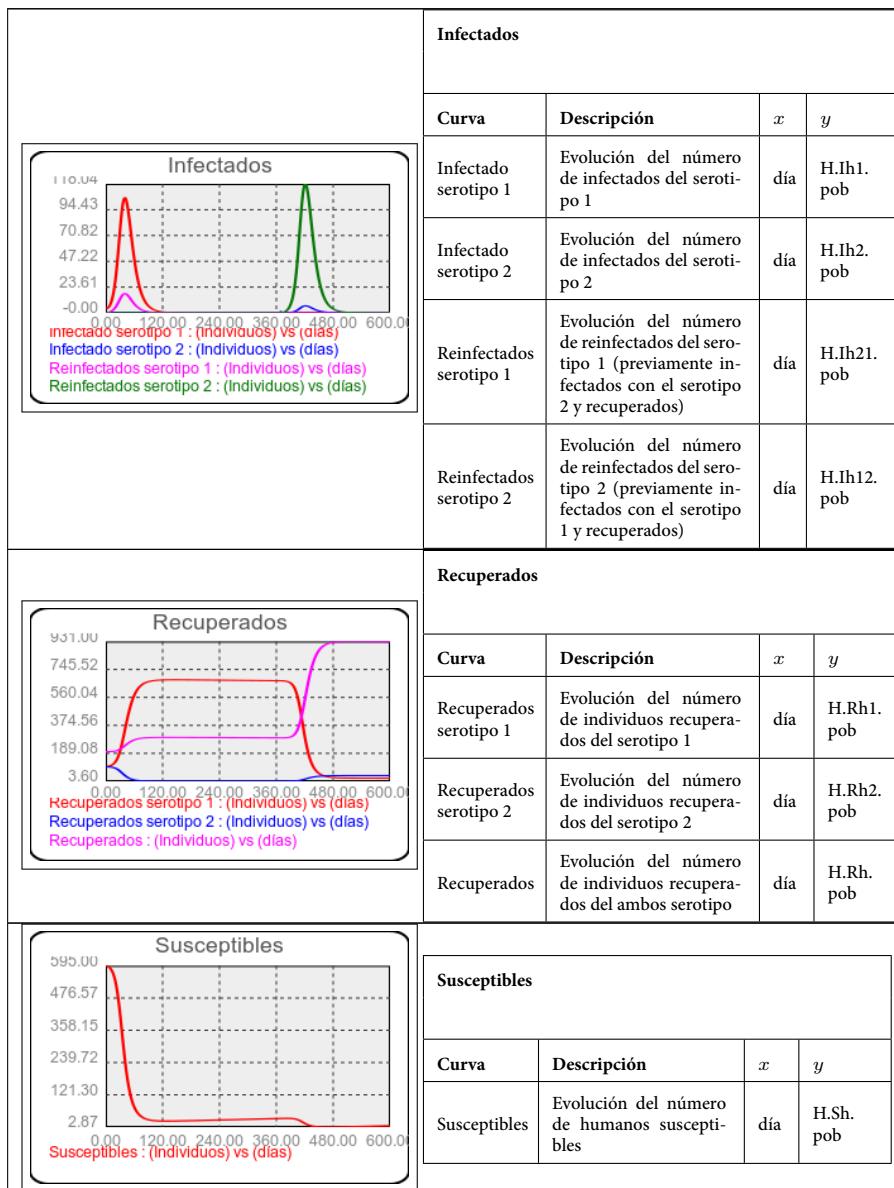
¿En qué difiere la propagación de la enfermedad entre los casos de una y dos cepas? Implemente el modelo de una única cepa y compare el comportamiento de los dos casos.

Tabla 10.10 Parámetros del experimento 9, “Modelo SIR para humanos y SI con dos cepas para mosquito”

Título:	Modelo SIR-SI con dos cepas		
Descripción:	Modelo de propagación de una enfermedad en el que hay dos cepas del virus. Los humanos recuperados de la infección de una cepa pueden reinfectarse con la otra.		
Créditos	Implementación		
	e-mail		
<b>Parámetros</b>			
Grupo	Nombre Modelica	Nombre	Descripción
Enfermedad	<i>b</i>	Tasa de picadura	Tasa de picadura ante contacto entre humano y mosquito
	<i>H.ph</i>	Probabilidad de contagio del humano	Probabilidad de contagio ante picadura
	<i>V.pv</i>	Probabilidad de contagio del vector	Probabilidad de contagio ante picadura
	<i>H.trec</i>	Tiempo de recuperación	Número de días de recuperación de la enfermedad
	<i>V.tvida</i>	Días de vida del mosquito	Número de días de vida del mosquito
	<i>H.tvida</i>	Años de vida del humano	Número de años de vida de los humanos
Poblaciones	<i>to2</i>	Día de entrada del serotipo 2	Día en que aparece el primer mosquito del serotipo 2
	<i>V.NIv1</i>	Número inicial de mosquitos infectados	Número de mosquitos infectados con el serotipo 1 al inicio de la simulación
	<i>V.NIv2</i>	Número inicial de mosquitos con serotipo 2	Número de mosquitos infectados con el serotipo 2 al inicio de la simulación
	<i>H.Nih1</i>	Número inicial de humanos infectados	Número de humanos infectados con el serotipo 1 al inicio de la simulación
	<i>H.Nih2</i>	Número inicial de humanos infectados con serotipo 2	Número de humanos infectados con el serotipo 2 al inicio de la simulación

<sup>5</sup>La información en las tablas siguientes se muestra tal como aparece en la base de datos. Por esta razón hay datos tanto en español como en inglés.

Tabla 10.11 Figuras del experimento 9, "Modelo SIR para humanos y SI con dos cepas para mosquito"



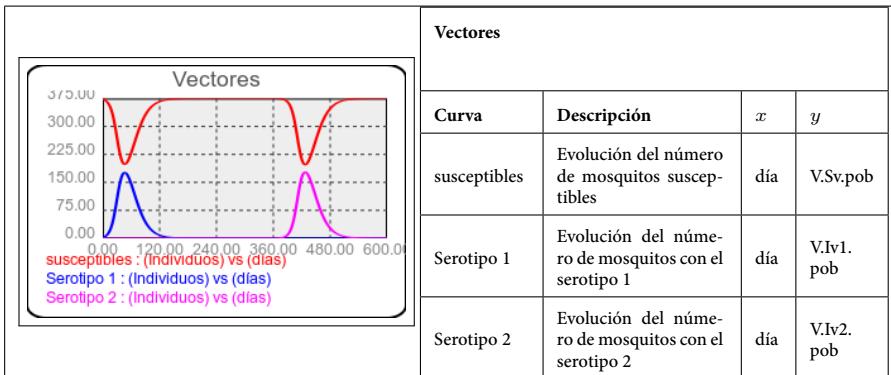


Tabla 10.12 Variables en la tabla de resultados del experimento 9, “Modelo SIR para humanos y SI con dos cepas para mosquito”

Variable	Descripción	Unidades
Día		Días
H.Ih1.pob		Individuos
H.Ih2.pob		Individuos
H.Ih21.pob		Individuos
H.Ih12.pob		Individuos
H.Rh1.pob		Individuos
H.Rh2.pob		Individuos
H.Rh.pob		Individuos
H.Sh.pob		Individuos
V.Sv.pob		Individuos
V.Iv1.pob		Individuos
V.Iv2.pob		Individuos

### 10.3 LA IMPLEMENTACIÓN

La implementación del modelo utiliza la librería de bloques Modelica Standard Library. Se definen dos clases principales: *compartimento* e *interaccion*.

- class *compartimento*: esta clase implementa la ecuación 10.1, haciendo explícitos los fenómenos de nacimiento y muerte. Para ello se utilizan dos vectores de objetos de la clase *interaccion*, que modelan el conjunto de entradas y salidas de individuos al compartimento, respectivamente.

Las variables pertenecientes a la clase *compartimento* son:

- `pob`: representa el número de individuos en el compartimento, es decir, su población.
- `nace`: representa el número de nacimientos en el compartimento.
- `muere`: representa el número de muertes de individuos del compartimento.
- `entra`: vector de objetos de la clase `interaccion`, que representa los fenómenos por los que aumenta la población del compartimento.
- `sale`: vector de objetos de la clase `interaccion`, que representa los fenómenos por los que disminuye la población del compartimento.
- `N`: variable de la población total de la especie del compartimento. Es una variable de tipo `outer` porque la población total de la especie es externa al compartimento.

Los parámetros pertenecientes a la clase `compartimento` son:

- `natalidad`: tasa de natalidad.
- `mortalidad`: tasa de mortalidad.
- `nentra`: número de interacciones en el vector `entra`.
- `nsale`: número de interacciones en el vector `sale`.

Las ecuaciones básicas implementadas son las siguientes:

$$\begin{aligned} \frac{dpob}{dt} &= \sum_{nentra} \text{entra} - \sum_{nsale} \text{sale} + nace - muere \\ nace &= \text{natalidad} \times N \\ muere &= \text{mortalidad} \times pob \end{aligned} \quad (10.8)$$

- `class interaccion`: esta clase modela el contacto entre dos especies. Es una extensión de un bloque general MISO con entradas  $x_1, x_2, \dots, x_{nin}$  y salida  $y$ . La salida  $y$  se calcula como

$$y = \text{coeficiente} * \prod_{i=1}^{nin} x_i \quad (10.9)$$

En general,  $x_i$  representa la población de algún compartimento. Por ejemplo, el número de contagios de una especie a otra es proporcional al número de veces que se encuentran dos individuos de esas especies; en esta situación, existirán dos entradas ( $nin = 2$ ), cada una de las cuales será la población de cada especie, y el coeficiente representará la probabilidad de contagio.

### 10.3.1 Ejemplo: implementación del modelo SIR

El modelo SIR (sección 10.1.1) emplea tres compartimientos denominados  $S$ ,  $I$ , y  $R$  (susceptibles, infectados y recuperados, respectivamente) y dos interacciones denominadas  $I1$  e  $I2$ .

La interacción  $I1$  representa la infección de individuos susceptibles, y corresponde al término  $\beta SI$  de la ecuación 10.2. Esta interacción tiene entonces dos entradas, correspondientes a las poblaciones de los compartimientos  $S$  e  $I$ , y su coeficiente corresponde a la tasa de contagio  $\beta$ .

La interacción  $I2$  representa la recuperación de individuos infectados, y corresponde al término  $\gamma I$  de la ecuación 10.2. Esta interacción tiene entonces una entrada, correspondiente a la población del  $I$ , y su coeficiente corresponde a la tasa de recuperación  $\gamma$ .

Las poblaciones iniciales de los tres compartimientos se han modelado con los parámetros  $NS$ ,  $NI$ , y  $NR$ . La implementación total del modelo se encuentra en el archivo SIR.mo.

### 10.3.2 Listado de archivos

La tabla 10.13 muestra el listado de los archivos fuente de la implementación del modelo.

Tabla 10.13 Archivos del modelo

Número	Archivo
10.1	SIR.mo
10.2	SIRn.mo
10.3	SIS.mo
10.4	compartimento.mo
10.5	dengue.mo
10.6	dosserotipos.mo
10.7	humano.mo
10.8	interaccion.mo
10.9	package.mo
10.10	vector.mo

## Archivo 10.1 SIR.mo

```

within Epidemia;

model SIR "Modelo Susceptible – Infectado – Recuperado"
  compartimento S(pob(start=NS),nentra=0,nsale=1);
  compartimento I(pob(start=NI),nentra=1,nsale=1);
  compartimento R(pob(start=NR),nentra=1,nsale=0);
  interaccion I1(nin=2,coeficiente=beta/(NS+NI+NR)) "Contagio";
  interaccion I2(nin=1,coeficiente=gamma) "Recuperacion";
  inner Real N;
  parameter Real beta=0.4 "Tasa de contagio";
  parameter Real gamma=1/trec "Tasa de recuperacion";
  parameter Real trec(unit="dias")=7 "Tiempo de recuperación";
  parameter Real NS=1000-NI-NR "Poblacion inicial de susceptibles";
  parameter Real NI=1 "Poblacion inicial de infectados";
  parameter Real NR=0 "Poblacion inicial de recuperados";
  Real dia(unit="días");
equation
  dia=time;
  N=(S.pob+I.pob+R.pob);
  connect(I1.u[1],S.pob);
  connect(I1.u[2],I.pob);
  connect(I2.u[1],I.pob);
  connect(I1.y,S.sale[1]);
  connect(I1.y,I.entrada[1]);
  connect(I2.y,I.sale[1]);
  connect(I2.y,R.entrada[1]);
end SIR;

```

## Archivo 10.2 SIRn.mo

```

within Epidemia;

model SIRn
  S(S.natalidad=mul,S.mortalidad=mul,I.mortalidad=mul,R.mortalidad=
    mul,NR=646);
  parameter Real mul=mu*1e-6;
  parameter Real mu=44.189;
end SIRn;

```

## Archivo 10.3 SIS.mo

```

within Epidemia;

model SIS
  interaccion I1(nin=2,coeficiente=beta/(NS+NI)) "Contagio";
  interaccion I2(nin=1,coeficiente=gamma) "Recuperación";
  compartimento S(nentra=1,nsale=1,pob(start=NS)) "Susceptibles";
  compartimento I(nentra=1,nsale=1,pob(start=NI)) "Infectados";
  inner Real N;
  parameter Real beta=0.4;
  parameter Real gamma=1/trec "Tasa de recuperacion";

```

```

parameter Real trec(unit="días")=7 "Tiempo de recuperación";
parameter Real NS=1000-NI;
parameter Real NI=1;
Real dia(unit="días");
equation
dia=time;
N=(S.pob+I.pob);
connect(I1.u[1],S.pob);
connect(I1.u[2],I.pob);
connect(I2.u[1],I.pob);
connect(S.entrada[1],I2.y);
connect(S.sale[1],I1.y);
connect(I.entrada[1],I1.y);
connect(I.sale[1],I2.y);
end SIS;

```

Archivo 10.4 compartimento.mo

```

within Epidemia;

model compartimento "compartimento de un grupo poblacional de una
especie"
outer Real N;
Real pob(unit="Individuos"),nace(unit="Individuos"),muere(unit="
Individuos");
parameter Real mortalidad=0,natalidad=0;
Modelica.Blocks.Interfaces.RealVectorInput entra[nentra];
Modelica.Blocks.Interfaces.RealVectorInput sale[nsale];
parameter Integer nentra=1,nsale=1;
equation
nace=N*natalidad;
muere=pob*mortalidad;
der(pob)=sum(entra)-sum(sale)+nace-muere;
end compartimento;

```

Archivo 10.5 dengue.mo

```

within Epidemia;

model dengue
vector V(muv=muv,pv=pv,b=b,NSv=Nv,NIv1=0,NIv2=0,Nh=Nh);
humano_joven(muh=muj,gamma=gamma,Nsh=600000,Nih1=100,Nih2=0,Nrh1=0,
Nrh2=500,Nih12=0,Nih21=0,Nrh=0,Nh=Nh,ph=pj,b=b,Sh.natalidad=
lambda/Nh,
Sh.nsale=3,Ih1.nsale=2,Ih2.nsale=2,Rh1.nsale=2,Rh2.nsale=2,
Ih12.nsale=2,Ih21.nsale=2,Rh.nsale=1);
humano_adulto(muh=muv,gamma=gamma,Nsh=199300,Nih1=100,Nih2=0,Nrh1
=100000,Nrh2=100000,Nih12=0,Nih21=0,Nrh=0,Nh=Nh,ph=pv,b=b,Sh.
natalidad=0,
Sh.nentra=1,Ih1.nentra=2,Ih2.nentra=2,Rh1.nentra=2,Rh2.nentra=2,
Ih12.nentra=2,Ih21.nentra=2,Rh.nentra=3);
interaccion ASh(nin=1,coeficiente=alfa) "paso a la adultez";
interaccion AIh1(nin=1,coeficiente=alfa) "paso a la adultez";

```

```

interaccion AIh2(nin=1,coeficiente=alfa)    "paso a la adultez";
interaccion ARh1(nin=1,coeficiente=alfa)    "paso a la adultez";
interaccion ARh2(nin=1,coeficiente=alfa)    "paso a la adultez";
interaccion AIh12(nin=1,coeficiente=alfa)   "paso a la adultez";
interaccion AIh21(nin=1,coeficiente=alfa)   "paso a la adultez";
interaccion ARh(nin=1,coeficiente=alfa)    "paso a la adultez";
parameter Real Nh = 1e6;
parameter Real Nv = 4e5;
parameter Real lambda = 54.8;
parameter Real muj = 1/(62*365);
parameter Real mua = 1/(47*365);
parameter Real muv = 1/11;
parameter Real kappa = 1;
parameter Real tr = 0;
parameter Real tir = 100;
parameter Real ts = 300;
parameter Real b = 0.8;
parameter Real pj = 0.1;
parameter Real pa = 0.2;
parameter Real pv = 0.5;
parameter Real alfa = 1/5500;
parameter Real gamma = 1/10;           // OJO: esto no está bien
manejado !!!!;
parameter Real fi = 1;
parameter Real q = 0.8;
Real Year;
equation
Year=time/365;
connect(ASh.u[1],joven.Sh.pob);
connect(ASh.y,    joven.Sh.sale[3]);
connect(ASh.y,    adulto.Sh.entrada[1]);
connect(AIh1.u[1],joven.Ih1.pob);
connect(AIh1.y,   joven.Ih1.sale[2]);
connect(AIh1.y,   adulto.Ih1.entrada[2]);
connect(AIh2.u[1],joven.Ih2.pob);
connect(AIh2.y,   joven.Ih2.sale[2]);
connect(AIh2.y,   adulto.Ih2.entrada[2]);
connect(ARh1.u[1],joven.Rh1.pob);
connect(ARh1.y,   joven.Rh1.sale[2]);
connect(ARh1.y,   adulto.Rh1.entrada[2]);
connect(ARh2.u[1],joven.Rh2.pob);
connect(ARh2.y,   joven.Rh2.sale[2]);
connect(ARh2.y,   adulto.Rh2.entrada[2]);
connect(AIh12.u[1],joven.Ih12.pob);
connect(AIh12.y,  joven.Ih12.sale[2]);
connect(AIh12.y,  adulto.Ih12.entrada[2]);
connect(AIh21.u[1],joven.Ih21.pob);
connect(AIh21.y,  joven.Ih21.sale[2]);
connect(AIh21.y,  adulto.Ih21.entrada[2]);
connect(ARh.u[1],joven.Rh.pob);
connect(ARh.y,   joven.Rh.sale[1]);
connect(ARh.y,   adulto.Rh.entrada[3]);
V.Iv1.pob = joven.Iv1;
V.Iv2.pob = joven.Iv2;

```

```

V.Iv1.pob = adulto.Iv1;
V.Iv2.pob = adulto.Iv2;
joven.Ih1.pob + adulto.Ih1.pob =V.Ih1;
joven.Ih12.pob + adulto.Ih12.pob =V.Ih12;
joven.Ih2.pob + adulto.Ih2.pob =V.Ih2;
joven.Ih21.pob + adulto.Ih21.pob =V.Ih21;
when time>ts then
    reinit(V.Iv2.pob,1);
end when;
end dengue;

```

Archivo 10.6 dosserotipos.mo

```

within Epidemia;

model dosserotipos
    vector V(b=b);
    humano H(b=b);
    parameter Real to2=370;
    parameter Real Iot2=1;
    Real dia(unit="días");
    Real anno(unit="años");
    parameter Real b=0.9;
equation
    dia=time;
    anno=dia/365;
    V.Iv1.pob =H.Iv1;
    V.Iv2.pob =H.Iv2;
    H.Ih1.pob =V.Ih1;
    H.Ih12.pob=V.Ih12;
    H.Ih2.pob =V.Ih2;
    H.Ih21.pob=V.Ih21;
    when time>to2 then
        reinit(H.Ih2.pob,Iot2);
    end when;
end dosserotipos;

```

Archivo 10.7 humano.mo

```

within Epidemia;

model humano
    compartimento Sh (pob(start=Nsh),nentra=0,nsale=2,mortalidad=muh,
        natalidad=muh);
    compartimento Ih1 (pob(start=Nih1),nentra=1,nsale=1,mortalidad=muh);
    compartimento Ih2 (pob(start=Nih2),nentra=1,nsale=1,mortalidad=muh);
    compartimento Rh1 (pob(start=Nrh1),nentra=1,nsale=1,mortalidad=muh);
    compartimento Rh2 (pob(start=Nrh2),nentra=1,nsale=1,mortalidad=muh);
    compartimento Ih12(pob(start=Nih12),nentra=1,nsale=1,mortalidad=muh);
    compartimento Ih21(pob(start=Nih21),nentra=1,nsale=1,mortalidad=muh);
    compartimento Rh (pob(start=Nrh),nentra=2,nsale=0,mortalidad=muh);
    inner Real N;
    interaccion C1(nin=2,coeficiente=ph*b/Nh) "Sh-Ih1";

```

```

interaccion C2(nin=1,coeficiente=gamma) "Ih1-Rh1";
interaccion C3(nin=2,coeficiente=ph*b/Nh) "Rh1-Ih12";
interaccion C4(nin=1,coeficiente=gamma) "Ih12-Rh";
interaccion C5(nin=2,coeficiente=ph*b/Nh) "Sh-Ih2";
interaccion C6(nin=1,coeficiente=gamma) "Ih2-Rh2";
interaccion C7(nin=2,coeficiente=ph*b/Nh) "Rh2-Ih21";
interaccion C8(nin=1,coeficiente=gamma) "Ih21-Rh";
parameter Real muh=1/(365*tvida);
parameter Real tvida(unit="años")=62;
parameter Real gamma=1/trec "Tasa de recuperacion";
parameter Real trec(unit="dias")=7 "Tiempo de recuperación";
parameter Real Nh=1000,Nih1=5,Nih2=0,Nrh1=100,Nrh2=100,Nih12=0,Nih21
=0,Nrh=200;
parameter Real Nsh=Nh-(Nih1+Nih2+Nrh1+Nrh2+Nih12+Nih21+Nrh);
parameter Real ph=0.4;
parameter Real b=0.9;
Real Iv1,Iv2;
equation
N=Sh.pob+Ih1.pob+Ih2.pob+Rh1.pob+Rh2.pob+Ih12.pob+Ih21.pob+Rh.pob;
connect(C1.u[1],Sh.pob);
connect(C1.u[2],Iv1);
connect(C2.u[1],Ih1.pob);
connect(C3.u[1],Rh1.pob);
connect(C3.u[2],Iv2);
connect(C4.u[1],Ih12.pob);
connect(C5.u[1],Sh.pob);
connect(C5.u[2],Iv2);
connect(C6.u[1],Ih2.pob);
connect(C7.u[1],Rh2.pob);
connect(C7.u[2],Iv1);
connect(C8.u[1],Ih21.pob);
connect(Sh.sale[1],C1.y);
connect(Sh.sale[2],C5.y);
connect(Ih1.entrada[1],C1.y);
connect(Ih1.sale[1],C2.y);
connect(Rh1.entrada[1],C2.y);
connect(Rh1.sale[1],C3.y);
connect(Ih12.entrada[1],C3.y);
connect(Ih12.sale[1],C4.y);
connect(Ih2.entrada[1],C5.y);
connect(Ih2.sale[1],C6.y);
connect(Rh2.entrada[1],C6.y);
connect(Rh2.sale[1],C7.y);
connect(Ih21.entrada[1],C7.y);
connect(Ih21.sale[1],C8.y);
connect(Rh.entrada[1],C4.y);
connect(Rh.entrada[2],C8.y);
end humano;

```

## Archivo 10.8 interaccion.mo

```

within Epidemia;

block interaccion "contactos entre especies. La probabilidad de
    ocurrencia se modela por el producto"
    extends Modelica.Blocks.Interfaces.MISO;
    parameter Real coeficiente=1.0 "parametro de la interacción";
equation
    y = coeficiente*product(u);
end interaccion;

```

## Archivo 10.9 package.mo

```

package Epidemia
end Epidemia;

```

## Archivo 10.10 vector.mo

```

within Epidemia;

model vector
    compartimento Sv(pob(start=NSv),nentra=0,nsale=4,mortalidad=muv,
        natalidad=muv);
    compartimento Iv1(pob(start=NIv1),nentra=2,nsale=0,mortalidad=muv);
    compartimento Iv2(pob(start=NIv2),nentra=2,nsale=0,mortalidad=muv);
    inner Real N;
    interaccion C11(nin=2,coeficiente=pv*b/Nh) ;
    interaccion C12(nin=2,coeficiente=pv*b/Nh) ;
    interaccion C21(nin=2,coeficiente=pv*b/Nh) ;
    interaccion C22(nin=2,coeficiente=pv*b/Nh) ;
    parameter Real muv=1/tvida;
    parameter Real tvida(unit="días")=11;
    parameter Real pv=0.8;
    parameter Real b=0.9;
    parameter Real NSv=375-(NIv1+NIv2);
    parameter Real NIv1=0;
    parameter Real NIv2=0;
    parameter Real Nh=1000;
    Real Ih1,Ih2,Ih12,Ih21;
equation
    N=Sv.pob+Iv1.pob+Iv2.pob;
    connect(C11.u[1],Sv.pob);
    connect(C11.u[2],Ih1);
    connect(C12.u[1],Sv.pob);
    connect(C12.u[2],Ih21);
    connect(C21.u[1],Sv.pob);
    connect(C21.u[2],Ih2);
    connect(C22.u[1],Sv.pob);
    connect(C22.u[2],Ih12);
    connect(C11.y,Iv1.entrada[1]);
    connect(C12.y,Iv1.entrada[2]);
    connect(C21.y,Iv2.entrada[1]);

```

```
connect(C22.y,Iv2.entrada[2]);
connect(C11.y,Sv.sale[1]);
connect(C12.y,Sv.sale[2]);
connect(C21.y,Sv.sale[3]);
connect(C22.y,Sv.sale[4]);
end vector;
```

# 11

## TRÁNSITO DE ESTUDIANTES A TRAVÉS DE UN PLAN DE ESTUDIOS

*El modelo representa el flujo de estudiantes a través de un plan de estudios de un programa universitario. El modelo permite estimar el número de estudiantes en cada periodo académico, el número total de estudiantes, el número de graduados y el tiempo medio de graduación.*

El número de estudiantes inscritos en un plan de estudios universitario depende de factores tales como el flujo de inscripción, las tasas de aprobación de asignaturas, la estructura curricular del plan, etc. Para la institución universitaria es necesario estimar la forma como evoluciona el número de estudiantes, para así poder prever los recursos mínimos que se deben proveer para atender la demanda académica de esa población estudiantil.



Figura 11.1 Ceremonia de graduación de ingenieros en la Universidad Nacional de Colombia. Año 1964<sup>1</sup>

<sup>1</sup>Tomada de [link:14].

El modelo que se presenta parte de una simplificación de los planes de estudio reales. A partir de allí, se formula una colección de ecuaciones de diferencia que derivan en un modelo dinámico lineal. El análisis matemático del modelo permite inferir el efecto que tienen algunos fenómenos sobre el número de estudiantes y el tiempo medio de graduación. El modelo se presenta de forma extensa y aplicada al caso particular de la Facultad de Ingeniería de la Universidad Nacional de Colombia en [Dua13]. Para comodidad del lector, se reproducen en la sección 11.1 los principales elementos matemáticos del modelo.

### 11.1 EL MODELO

Un plan de estudios típico de un programa universitario cuenta con una colección de asignaturas, entre obligatorias y electivas, que se distribuyen a lo largo de varios períodos académicos. A manera de ejemplo, los planes de estudio de la Facultad de Ingeniería en la Universidad Nacional constan de cerca de 60 asignaturas, distribuidas a lo largo de 10 semestres académicos.

La figura 11.2 muestra una versión simplificada de un plan de estudios. Se trata de un plan de estudios de  $n$  períodos académicos, con una única asignatura en cada semestre, que es requisito de la asignatura del siguiente semestre. Con este modelo simplificado se pueden representar varios fenómenos asociados al tránsito de los estudiantes a través del plan de estudios.

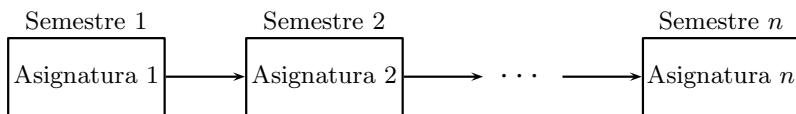
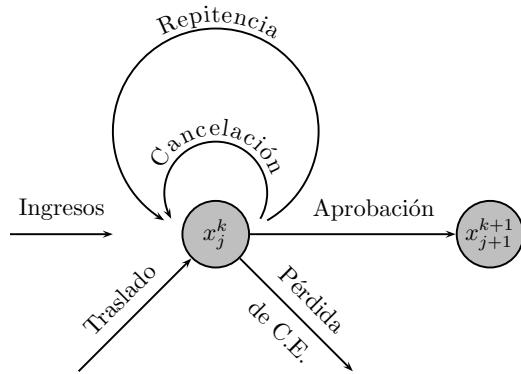


Figura 11.2 Plan de estudios del modelo mínimo

Para ello, se formula el siguiente comportamiento dinámico (D), que se ilustra en la figura 11.3:

- D 1. Se utiliza el subíndice  $j$  para diferenciar cada una de las  $n$  asignaturas del plan de estudios. Debido a que en cada semestre hay una única asignatura, el subíndice  $j$  también sirve para identificar el semestre.
- D 2. Se utiliza la variable  $k$  para referirse al periodo académico.  $k$  es una variable discreta que puede asimilarse a la secuencia de períodos (p. ej., ‘2009-01’, ‘2009-03’, ‘2010-01’, etc.) o a los enteros ( $k = 1, 2, \dots$ ).
- D 3. El número de estudiantes que inscriben la asignatura  $j$  en el periodo académico  $k$  se denota por  $x_j(k)$ .

Figura 11.3 Diagrama de flujo para el estado  $x_j(k)$ 

D 4. Al pasar de un periodo académico al siguiente, algunos estudiantes pasan a la asignatura del semestre posterior; algunos no lo logran, bien porque cancelan o porque no aprueban (véase figura 11.3):

$$x_{j+1}(k+1) = x_{ap_j}(k) + x_{can_{j+1}}(k) + x_{nap_{j+1}}(k) + t_j(k) \quad (11.1)$$

en donde:

- $x_{ap_j}(k)$ : número de estudiantes que aprobaron la asignatura  $j$  en el periodo  $k$ .
- $x_{can_{j+1}}(k)$ : número de estudiantes que cancelaron la asignatura  $j+1$  en el periodo  $k$ .
- $x_{nap_{j+1}}(k)$ : número de estudiantes que inscribieron y que no cancelaron ni aprobaron la asignatura  $j+1$  en el periodo  $k$  y que tampoco perdieron la calidad de estudiante.
- $t_j(k)$ : número de estudiantes que ingresan al programa por traslado o por autorización de doble titulación. Estos estudiantes inician estudios en el periodo  $(k+1)$ , inscribiendo la asignatura  $j$ .

D 5. Para el semestre 1 la dinámica se representa por:

$$x_1(k+1) = x_{can_1}(k) + x_{nap_1}(k) + u(k) + t_j(k) \quad (11.2)$$

En donde

- $u(k)$ : Número de admitidos en el periodo  $k$  que inician estudios en el periodo  $(k+1)$ .

D 6. Se definen los siguientes parámetros:

- $\alpha_j(k)$ : tasa de aprobación para la asignatura  $j$  en el periodo  $k$ .
- $\beta_j(k)$ : tasa de cancelación para la asignatura  $j$  en el periodo  $k$ .
- $\sigma_j(k)$ : tasa de pérdida de calidad de estudiante para el semestre  $j$  en el periodo  $k$ .

D 7. Los términos de la derecha en la ecuación 11.1 pueden entonces calcularse así:

$$\begin{aligned} x_{ap_j}(k) &= x_j(k)(1 - \beta_j(k))\alpha_j(k)(1 - \sigma_{j+1}(k)) \\ x_{can_{j+1}}(k) &= x_{j+1}(k)\beta_{j+1}(k)(1 - \sigma_{j+1}(k)) \\ x_{nap_{j+1}}(k) &= x_{j+1}(k)(1 - \beta_{j+1}(k))(1 - \alpha_{j+1}(k))(1 - \sigma_{j+1}(k)) \end{aligned} \quad (11.3)$$

D 8. Se denota por  $y_j(k)$  al número de estudiantes que inscriben una asignatura de un semestre menor o igual a  $j$  en el periodo  $k$ .

D 9. Se denota por  $y_T(k)$  al número total de estudiantes del programa en el periodo  $k$ , es decir  $y_T(k) = y_n(k)$ .

D 10. Se denota por  $g(k)$  al número de estudiantes graduados al terminar el periodo académico  $k$ .

Con esas condiciones, se puede derivar un modelo dinámico discreto lineal variante en el tiempo. A partir de las ecuaciones 11.1 y 11.3 se obtiene un conjunto de ecuaciones válidas para  $j = 1, 2, \dots, n - 1$ :

$$x_{j+1}(k + 1) = x_{ap_j}(k) + x_{can_{j+1}}(k) + x_{nap_{j+1}}(k) + t_j(k)$$

Para el primer semestre se emplea la ecuación 11.2:

$$x_1(k + 1) = x_{can_1}(k) + x_{nap_1}(k) + u(k) + t_j(k)$$

El conjunto extendido de ecuaciones resulta ser

$$\left. \begin{aligned}
 x_1(k+1) &= x_1(k)\beta_1(k)(1 - \sigma_1(k)) + \\
 &\quad x_1(k)(1 - \beta_1(k))(1 - \alpha_1(k))(1 - \sigma_1(k)) + \\
 &\quad u(k) + t_1(k) \\
 x_2(k+1) &= x_1(k)(1 - \beta_1(k))\alpha_1(k)(1 - \sigma_2(k)) + \\
 &\quad x_2(k)\beta_2(k)(1 - \sigma_2(k)) + \\
 &\quad x_2(k)(1 - \beta_2(k))(1 - \alpha_2(k))(1 - \sigma_2(k)) + \\
 &\quad t_2(k) \\
 x_3(k+1) &= x_2(k)(1 - \beta_2(k))\alpha_2(k)(1 - \sigma_3(k)) + \\
 &\quad x_3(k)\beta_3(k)(1 - \sigma_3(k)) + \\
 &\quad x_3(k)(1 - \beta_3(k))(1 - \alpha_3(k))(1 - \sigma_3(k)) + \\
 &\quad t_3(k) \\
 &\vdots \\
 x_n(k+1) &= x_{n-1}(k)(1 - \beta_{n-1}(k))\alpha_{n-1}(k)(1 - \sigma_n(k)) + \\
 &\quad x_n(k)\beta_n(k)(1 - \sigma_n(k)) + \\
 &\quad x_n(k)(1 - \beta_n(k))(1 - \alpha_n(k))(1 - \sigma_n(k)) + \\
 &\quad t_n(k) \\
 y_1(k) &= x_1(k) \\
 y_2(k) &= x_1(k) + x_2(k) \\
 &\vdots \\
 y_n(k) &= x_1(k) + x_2(k) + x_3(k) + \cdots + x_n(k) \\
 g(k) &= (1 - \beta_n(k))\alpha_n(k)(1 - \sigma_n(k))x_n(k)
 \end{aligned} \right\} \tag{11.4}$$

Las ecuaciones en 11.4 pueden escribirse de forma más compacta. Para ello definimos:

$$\left. \begin{aligned} \gamma_j(k) &= (1 - \beta_j(k))\alpha_j(k)(1 - \sigma_j(k)) \\ \delta_j(k) &= \beta_j(k)(1 - \sigma_j(k)) + (1 - \beta_j(k))(1 - \alpha_j(k))(1 - \sigma_j(k)) \end{aligned} \right\} \quad (11.5)$$

en cuyo caso

$$\left. \begin{aligned} x_1(k+1) &= x_1(k)\delta_1(k) + u(k) + t_1(k) \\ x_2(k+1) &= x_1(k)\gamma_1(k) + x_2(k)\delta_2(k) + t_2(k) \\ x_3(k+1) &= x_2(k)\gamma_2(k) + x_3(k)\delta_3(k) + t_3(k) \\ &\vdots \\ x_n(k+1) &= x_{n-1}(k)\gamma_{n-1}(k) + x_n(k)\delta_n(k) + t_n(k) \\ y_1(k) &= x_1(k) \\ y_2(k) &= x_1(k) + x_2(k) \\ &\vdots \\ y_n(k) &= x_1(k) + x_2(k) + x_3(k) + \cdots + x_n(k) \\ g(k) &= \gamma_n x_n(k) \end{aligned} \right\} \quad (11.6)$$

La ecuación 11.7 presenta el modelo en forma matricial:

$$\left. \begin{array}{l}
 X(k+1) = A(k)X(k) + B(k)U(k) \\
 Y(k) = C(k)X(k) + D(k)U(k) \\
 A(k) = \begin{bmatrix} \delta_1(k) & 0 & 0 & \cdots & 0 & 0 \\ \gamma_1(k) & \delta_2(k) & 0 & \cdots & 0 & 0 \\ 0 & \gamma_2(k) & \delta_3(k) & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \vdots & \gamma_{n-1}(k) & \delta_n(k) \end{bmatrix}_{n \times n} \\
 [3mm]B(k) = \begin{bmatrix} 1 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 \end{bmatrix}_{n \times (n+1)} \\
 C(k) = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 0 & 1 & 1 & \cdots & 1 \\ 0 & 0 & 1 & \cdots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & \gamma_n(k) \end{bmatrix}_{n \times n} \quad D(k) = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}_{n \times (n+1)} \\
 X(k) = \begin{bmatrix} x_1(k) \\ x_2(k) \\ \vdots \\ x_n(k) \end{bmatrix}_{n \times 1} \quad Y(k) = \begin{bmatrix} y_1(k) \\ y_2(k) \\ \vdots \\ y_n(k) \\ g(k) \end{bmatrix}_{(n+1) \times 1} \quad U(k) = \begin{bmatrix} u(k) \\ t_1(k) \\ t_2(k) \\ \vdots \\ t_n(k) \end{bmatrix}_{(n+1) \times 1} \\
 \gamma_j(k) = (1 - \beta_j(k))\alpha_j(k)(1 - \sigma_j(k)) \\
 \delta_j(k) = \beta_j(k)(1 - \sigma_j(k)) + (1 - \beta_j(k))(1 - \alpha_j(k))(1 - \sigma_j(k))
 \end{array} \right\} (11.7)$$

### 11.1.1 Modelo reducido

Con el propósito de facilitar el análisis del modelo, se introducen dos simplificaciones:

- Invarianza en el tiempo: todos los parámetros del modelo permanecen constantes a lo largo del tiempo.
- Homogeneidad: los parámetros del modelo son iguales para las  $n$  asignaturas del plan de estudios.

Estas simplificaciones implican:

$$\begin{aligned}\alpha_j(k) &= \alpha \quad \forall j, k \\ \beta_j(k) &= \beta \quad \forall j, k \\ \sigma_j(k) &= \sigma \quad \forall j, k\end{aligned}\tag{11.8}$$

El modelo reducido resulta ser un modelo lineal invariante en el tiempo:

$$\begin{aligned}X(k+1) &= AX(k) + BU(k) \\ Y(k) &= CX(k) + DU(k)\end{aligned}\tag{11.9}$$

Para un programa de 10 semestres, desarrollado durante 8 períodos, esta simplificación reduce el número de parámetros de  $10 * 8 * 3 = 240$  a solo 3.

### 11.1.2 Análisis del modelo reducido

Utilizando el modelo reducido, se formulan las siguientes preguntas de análisis (P):

- P 1. ¿Cómo evoluciona el número de estudiantes de un programa si el ingreso es constante?
- P 2. ¿Cuál es el número esperado de estudiantes de un programa si el ingreso es constante?
- P 3. ¿Qué efecto tienen los parámetros del modelo sobre el número esperado de estudiantes?
- P 4. ¿Cuántos estudiantes se graduarán y en qué tiempo lo harán?
- P 5. ¿Qué efecto tienen los parámetros del modelo sobre el número de graduados y los tiempos de graduación?

Estas preguntas se abordan en las secciones siguientes.

### 11.1.3 Evolución del número de estudiantes

Para analizar la evolución del número de estudiantes se considera el caso en que no hay traslados ( $t_j(k) = 0$ ) y los ingresos son constantes ( $u(k) = u$ ). En otras palabras, se realiza un análisis de respuesta a una entrada escalón en los ingresos.

Si se considera como salida del sistema a  $y_T(k)$  (el número total de estudiantes del sistema), entonces la función de transferencia  $F_T(z)$  puede definirse como:

$$F_T(z) = \frac{Y_T(z)}{U(z)}$$

La expresión de  $F_T(z)$  corresponde a (véase [Dual3])

$$F_T(z) = \frac{1}{\gamma} \sum_{j=1}^n \frac{\gamma^j}{(z - \delta)^j}$$

El sistema tiene un único polo  $\delta$  que aparece  $n$  veces en la función de transferencia. La estabilidad del sistema depende entonces del valor de  $\delta$ . Tanto  $\alpha$  como  $\beta$  son valores en el intervalo  $[0, 1]$ , por tanto

$$\begin{aligned} 0 \leq \gamma &= (1 - \beta)\alpha \leq 1 \\ 0 \leq \delta &= 1 - \gamma \leq 1 \end{aligned}$$

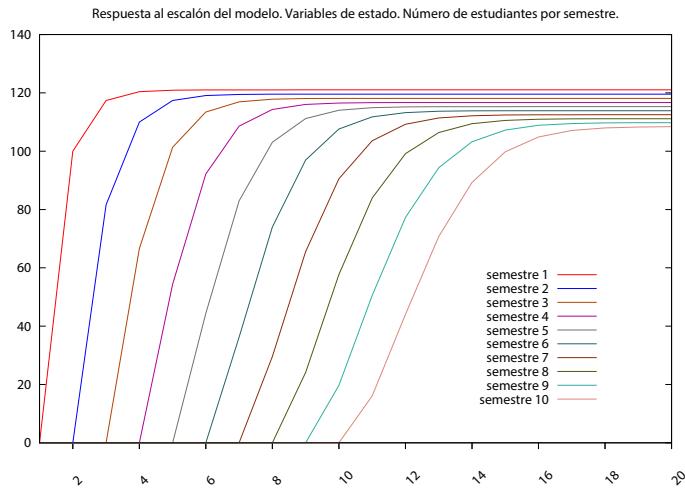
Como  $\delta$  está en el intervalo  $[0, 1]$ , el sistema es estable. En el caso frontera en que  $\delta = 1$ , el sistema será marginalmente estable. Para que esto suceda, se necesita que  $\gamma = 0$ , es decir, se requiere que se satisfaga una de las siguientes condiciones:

- $\beta = 1$ , lo que significa que todos los estudiantes cancelan todas las asignaturas.
- $\alpha = 0$ , lo que significa que ningún estudiante aprueba asignaturas.

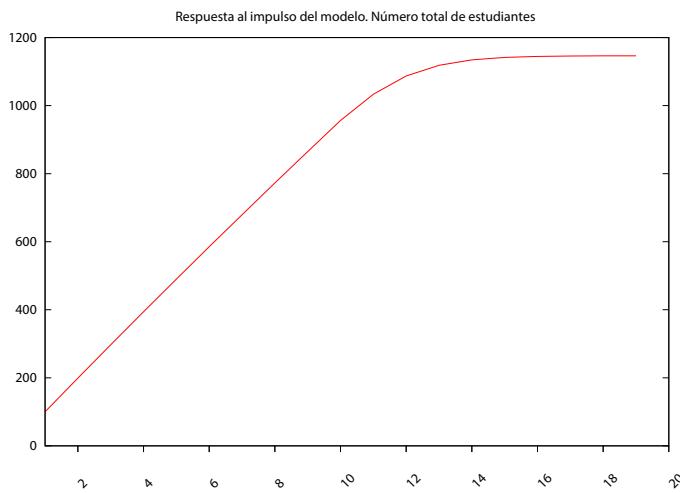
#### 11.1.3.1 Ejemplo

Para ilustrar el comportamiento del número de estudiantes, se ha corrido el modelo para 10 semestres ( $n = 10$ ), sin traslados ( $t_j(k) = 0$ ), con un ingreso constante de 100 estudiantes por periodo ( $u(k) = u = 100$ ) y condiciones iniciales nulas ( $x_j(0) = 0$ ). Los parámetros del modelo se han fijado en  $\alpha = 0,85$ ,  $\beta = 0,03$  y  $\sigma = 0,01$ . Esta simulación representa la evolución esperada de un programa nuevo.

La figura 11.4 muestra los resultados de la simulación. En la figura 11.4(a) se muestra cómo evoluciona el número de estudiantes inscritos en cada semestre, y la figura 11.4(b) muestra la evolución del número total de estudiantes. Al analizar esas figuras se encuentra lo siguiente:



(a) Número de estudiantes por semestre



(b) Número total de estudiantes

Figura 11.4 Respuesta al escalón del modelo.  $\alpha = 0,85$   $\beta = 0,03$   $\sigma = 0,01$

- El sistema se estabiliza después de aproximadamente 16 semestres.
- El valor de estado estacionario de cada semestre es diferente, siendo mayor el de los semestres inferiores. Estas diferencias se deben a los estudiantes que no alcanzan los semestres superiores debido a la pérdida de la calidad de estudiante.
- Los tiempos de estabilización de cada semestre son diferentes: las curvas inician en tiempos diferentes y presentan inclinaciones diferentes, siendo menor la pendiente de los semestres superiores.

#### 11.1.4 Número esperado de estudiantes

Para analizar el número esperado de estudiantes, se calcula el número total de estudiantes cuando han transcurrido muchos semestres, sin traslados ( $t_j(k) = 0$ ) y con ingresos constantes ( $u_j(k) = u$ ). En otras palabras, se evalúa el valor en estado estacionario de la variable  $y_T(k)$  (véase [Dua13])

$$y_{ee} = \lim_{k \rightarrow \infty} y_T(k)$$

$$y_{ee} = \frac{u}{\sigma} \left( 1 - \left( \frac{(1-\beta)\alpha(1-\sigma)}{\sigma + (1-\beta)\alpha(1-\sigma)} \right)^n \right)$$

##### 11.1.4.1 Caso sin pérdida de la calidad de estudiante

Si la única causa de pérdida de la calidad de estudiante se debe a la graduación ( $\sigma = 0$ ), el valor esperado del número de estudiantes debe calcularse (véase [Dua13]) como:

$$y_{ee} = \frac{nu}{\gamma} = \frac{nu}{(1-\beta)\alpha}$$

##### 11.1.4.2 Caso de referencia

Un caso especial se considera: si no hay cancelaciones, ni pérdida de la calidad de estudiante, y todos los estudiantes aprueban todas las asignaturas, el número de estudiantes se reduce a

$$y_{ref} = y_{ee} = nu$$

#### 11.1.5 Efecto de los parámetros sobre el número esperado de estudiantes

Para evaluar el efecto de los parámetros del modelo sobre el número esperado de estudiantes, se ha considerado un programa de 10 semestres ( $n = 10$ ), al

que ingresan 100 estudiantes cada periodo  $u = 100$ . Con esas consideraciones, el valor de referencia es de 1000 estudiantes ( $y_{ref} = nu = 1000$ ).

Sobre ese programa base se han realizado los siguientes experimentos:

- Caso 1: se han variado los parámetros del modelo  $\alpha$ ,  $\beta$  y  $\sigma$ , uno a la vez. La variación se ha hecho desde 0 hasta 1 y se ha calculado el número esperado de estudiantes. En cada caso, cuando un parámetro permanece constante, toma los siguientes valores:

$$\alpha = 0 \quad \beta = 0 \quad \sigma = 0$$

- Caso 2: se han variado los parámetros del modelo  $\alpha$ ,  $\beta$  y  $\sigma$ , uno a la vez. La variación se ha hecho desde 0 hasta 1 y se ha calculado el número esperado de estudiantes. En cada caso, cuando un parámetro permanece constante, toma los siguientes valores:

$$\alpha = 0,83 \quad \beta = 0,01 \quad \sigma = 0,03$$

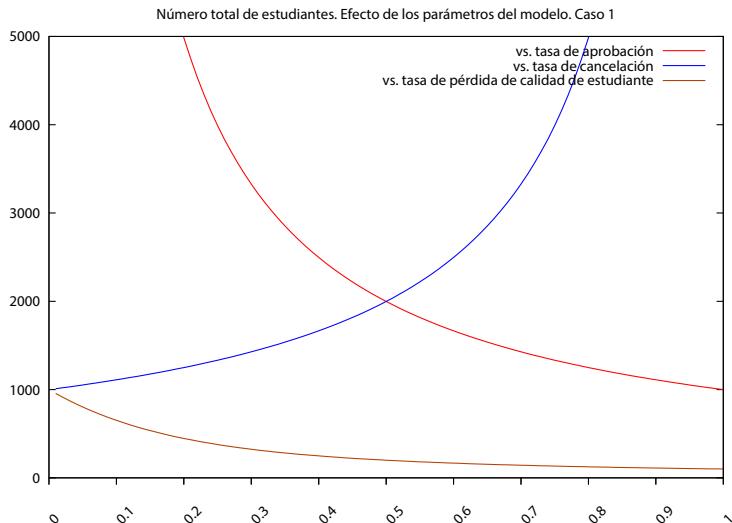
Los resultados de los experimentos se muestran en la figura 11.5. Al analizar esas figuras se hace evidente lo siguiente:

- El número total de estudiantes es muy sensible a las tasas de cancelación y de aprobación.
- La sensibilidad del número total de estudiantes a las tasas de cancelación y de aprobación aumenta en el caso 1, es decir, aumenta al disminuir la tasa de pérdida de la calidad de estudiante.
- La tasa de cancelación actual está cercana a 0.2. Para estos valores, la cancelación puede estar causando un incremento hasta del 20% en el número total de estudiantes.

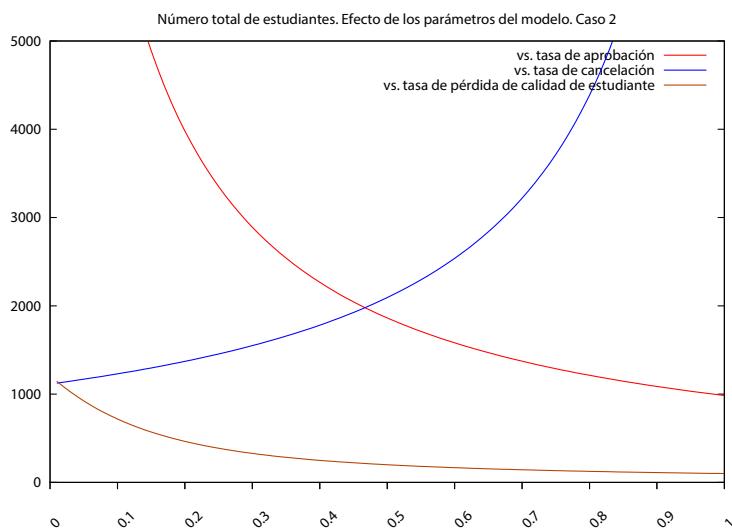
#### 11.1.5.1 Valor esperado y valor de referencia

La figura 11.5 muestra que el efecto sobre el número esperado de estudiantes de algunos parámetros es creciente y el de otros es decreciente. Este hecho genera una pregunta de análisis: ¿qué combinación de parámetros hace que el número esperado de estudiantes se igual el número de referencia? o lo que es igual, bajo qué condiciones se satisface:

$$\frac{u}{\sigma} \left( 1 - \left( \frac{(1-\beta)\alpha(1-\sigma)}{\sigma + (1-\beta)\alpha(1-\sigma)} \right)^n \right) = nu$$



(a) Caso 1.



(b) Caso 2.

Figura 11.5 Efecto de la variación de los parámetros del modelo sobre el número esperado de estudiantes

Para el caso en que  $n = 10$ , la combinación de parámetros resulta ser (véase [Dua13])

$$\alpha = \frac{\sigma/(1-\beta)}{(1-10\sigma)^{-0,1} - 1} \quad (11.10)$$

La figura 11.6 muestra la superficie tridimensional que se genera a partir de la ecuación 11.10:

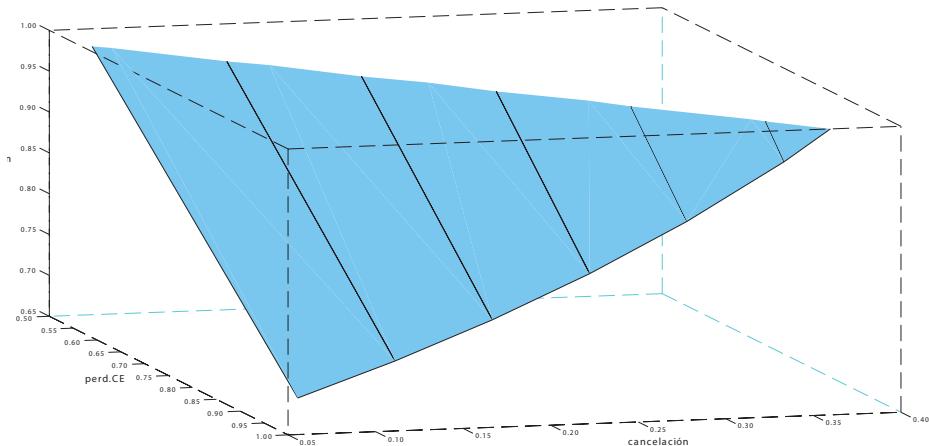


Figura 11.6 Superficie de población de referencia

### 11.1.6 Tiempos de graduación

No todos los estudiantes que ingresan se gradúan; los estudiantes que se gradúan lo hacen en un número de semestres diferentes. Para explorar los tiempos de graduación de los estudiantes admitidos en el periodo  $m$  (que ingresan al programa en el periodo  $m + 1$ ), se define la serie de graduación  $G_m(k)$  como

$$G_m(k) = g(k)|_{impulso} \quad (11.11)$$

El subíndice *impulso* denota las condiciones de respuesta al impulso:

- Condiciones iniciales nulas:  $x_j(0) = 0$ .
- Traslados nulos :  $t_j(k) = 0$ .
- La serie de ingreso representa un impulso discreto en el periodo  $m$ :

$$u(k) = \begin{cases} u & \text{si } k = m \\ 0 & \text{si } k \neq m \end{cases} \quad (11.12)$$

#### 11.1.6.1 Indicadores de los tiempos de graduación

Utilizando la serie  $G_m(k)$  pueden definirse varios indicadores sobre el proceso de graduación:

- Número total de graduados  $GT_m$ : corresponde a la suma de todos los términos de la serie:

$$GT_m = \sum_{k=0}^{\infty} G_m(k)$$

- Tiempo medio de graduación  $\overline{G_m}$ : el tiempo transcurrido entre el ingreso al programa (que sucede en  $m + 1$ ) y el tiempo  $k$  es  $(k - (m + 1)) = (k - m - 1)$ ; por tanto, el tiempo medio de graduación se calcula:

$$\overline{G_m} = \frac{\sum_{k=0}^{\infty} (k - m - 1) G_m(k)}{GT_m}$$

- Varianza en el tiempo de graduación: calculada como

$$GV_m = \text{Var}[G_m(k)] = \frac{\sum_{k=0}^{\infty} G_m(k) \left( \frac{G_m(k) - \overline{G_m}}{G_m} \right)^2}{GT_m}$$

Si se considera el tiempo de graduación para una única cohorte que ingresa en el periodo inicial ( $m = 0$ ), entonces las expresiones para la serie de graduación y el tiempo medio de graduación resultan ser (véase [Dua13]):

$$G_0(k) = \begin{cases} \frac{\delta^{k-n} \gamma^n (k-1)!}{(n-1)!(k-n)!} & \text{si } k \geq n \\ 0 & \text{si } k < n \end{cases}$$

$$\overline{G_0} = \sum_{k=n}^{\infty} \frac{\delta^{k-n} \gamma^n k!}{(n-1)!(k-n)!} \quad (11.13)$$

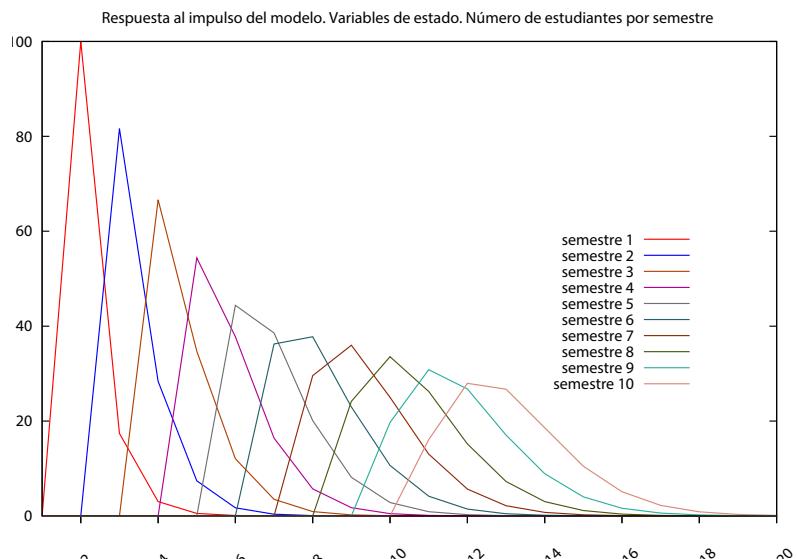
$$\gamma = (1 - \beta)\alpha(1 - \sigma)$$

$$\delta = \beta(1 - \sigma) + (1 - \beta)(1 - \alpha)(1 - \sigma)$$

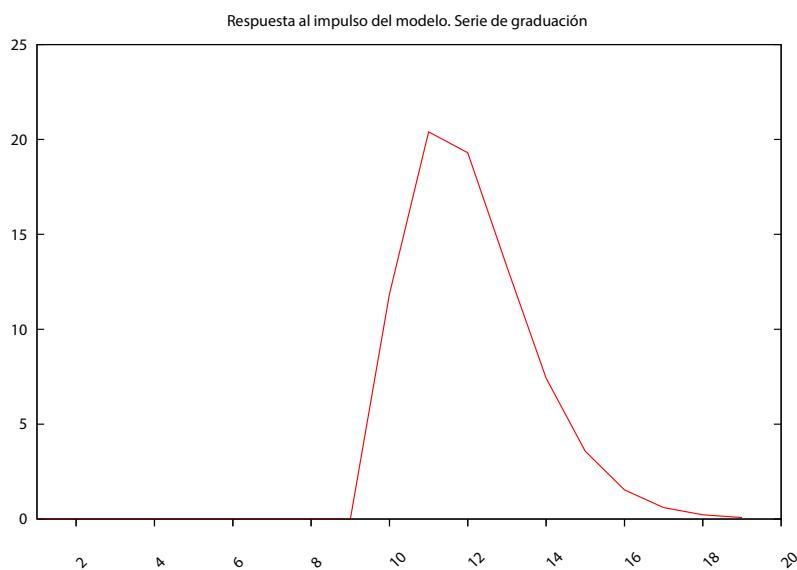
### 11.1.6.2 Ejemplo

Para ilustrar el comportamiento del número de estudiantes se ha corrido el modelo para 10 semestres ( $n = 10$ ), sin traslados ( $t_j(k) = 0$ ), con un ingreso constante de 100 estudiantes en el periodo inicial ( $u(0) = u = 100$ ) y condiciones iniciales nulas ( $x_j(0) = 0$ ). Los parámetros del modelo se han fijado en  $\alpha = 0,85$ ,  $\beta = 0,03$  y  $\sigma = 0,01$ . Esta simulación representa la evolución de una única cohorte.

La figura 11.7 muestra los resultados de la simulación. En la figura 11.7(a) se muestra cómo evoluciona el número de estudiantes inscritos en cada semestre, y la figura 11.7(b) muestra la evolución del número total de estudiantes.



(a) Número de estudiantes por semestre



(b) Número total de estudiantes

Figura 11.7 Respuesta al impulso del modelo.  $\alpha = 0,85$   $\beta = 0,03$   $\sigma = 0,01$

Para este ejemplo, los indicadores de los tiempos de graduación toman los siguientes valores:

$$GT_m = 88,5$$

$$\overline{G_m} = 12,1$$

$$GV_m = 2,5$$

### 11.1.7 Efecto de los parámetros sobre los tiempos de graduación

Para evaluar el efecto de los parámetros del modelo sobre el número esperado de estudiantes, se ha considerado un programa de 10 semestres ( $n = 10$ ) al que ingresan 100 estudiantes cada periodo  $u = 100$ . Con esas consideraciones, el valor de referencia es de 1000 estudiantes ( $y_{ref} = nu = 1000$ ).

Sobre ese programa base se han realizado los siguientes experimentos:

- Caso 1: se han variado los parámetros del modelo  $\alpha$ ,  $\beta$  y  $\sigma$ , uno a la vez. La variación se ha hecho desde 0 hasta 1 y se ha calculado el tiempo medio de graduación. En cada caso, cuando un parámetro permanece constante, toma los siguientes valores:

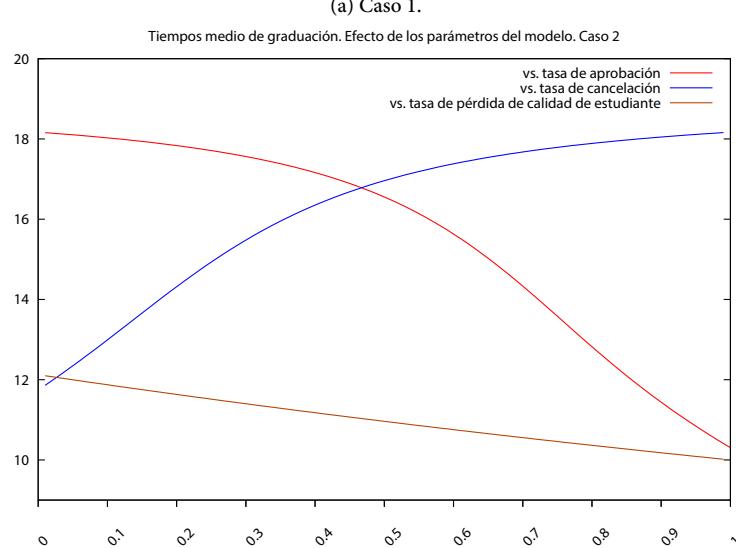
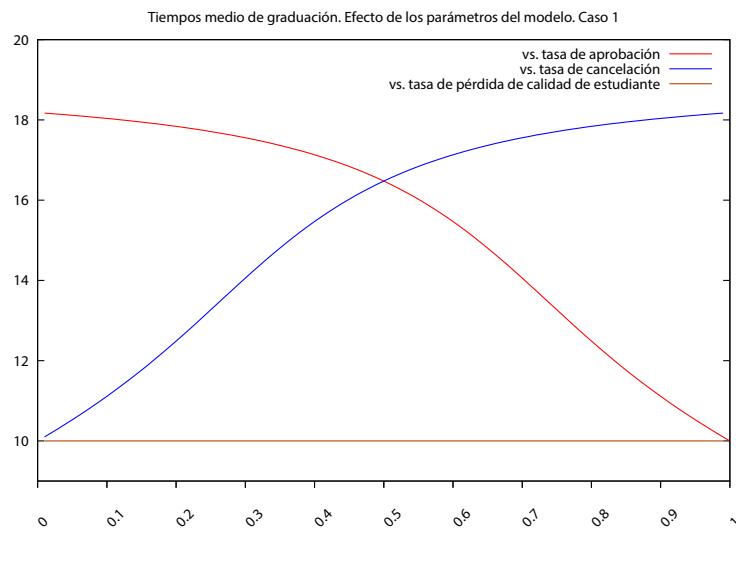
$$\alpha = 0 \quad \beta = 0 \quad \sigma = 0$$

- Caso 2: se han variado los parámetros del modelo  $\alpha$ ,  $\beta$  y  $\sigma$ , uno a la vez. La variación se ha hecho desde 0 hasta 1 y se ha calculado el tiempo medio de graduación. En cada caso, cuando un parámetro permanece constante, toma los siguientes valores:

$$\alpha = 0,83 \quad \beta = 0,01 \quad \sigma = 0,03$$

Los resultados de los experimentos se muestran en la figura 11.8. Al analizar esas figuras se encuentra lo siguiente:

- El tiempo medio de graduación es muy sensible a las tasa de cancelación y de aprobación.
- El tiempo medio de graduación no es muy sensible a la tasa de pérdida de calidad de estudiante.
- La sensibilidad del número total de estudiantes a la tasa de cancelación y de aprobación es semejante en los dos casos.
- La tasa de cancelación actual está cercana a 0.2. Para estos valores, la cancelación puede estar causando un incremento de hasta dos semestres en los tiempos medios de graduación.



(b) Caso 2.

Figura 11.8 Efecto de la variación de los parámetros del modelo sobre el tiempo medio de graduación

## 11.2 PLANTAS DE EXPERIMENTACIÓN Y EXPERIMENTOS SUGERIDOS

### Planta de experimentación 10: Respuesta al escalón.

**Presentación:** esta planta implementa el modelo reducido que se explica en la sección 11.1.1. La planta considera un plan de estudios nuevos, es decir, con condiciones iniciales nulas. El ingreso de estudiantes y los traslados son constantes. En otras palabras, la planta permite realizar simulaciones de la respuesta al escalón del modelo reducido. El número de estudiantes que ingresan al plan por concepto de traslados se distribuye uniformemente en todos los semestres.

**Instrumentación<sup>2</sup>:** el modelo cuenta con 6 parámetros ajustables organizados en 3 grupos de controles (véase tabla 11.1). Como resultado del experimento, el programa despliega:

- 11 curvas organizadas en 2 gráficos (véase tabla 11.2).
- Una tabla de datos del comportamiento de 12 variables (véase tabla 11.3).

**Experimentos sugeridos:** el siguiente es el listado de experimentos sugeridos:

#### *Experimento 10.1: Número estable de estudiantes.*

¿Cómo afectan los parámetros de desempeño académico el número estacionario de estudiantes? Explore el efecto que tiene la acción de modificar los parámetros de desempeño académico en el número de estudiantes cuando han transcurrido suficientes semestres para que se stabilice ese valor.

#### *Experimento 10.2: Tiempo de estabilización.*

¿Cómo afectan los parámetros de desempeño académico el tiempo (número de semestres) en que se estabiliza el número de estudiantes? Explore el efecto que tiene la acción de modificar los parámetros de desempeño académico el número semestres necesarios para que se stabilice el sistema.

***Experimento 10.3: Condiciones equivalentes de referencia.***

¿Qué desempeños académicos producen una población final igual a la de referencia? Explore qué combinaciones de los parámetros de referencia hacen que el número de estudiantes sea igual al de la condición de referencia (ingreso x número de semestres del plan de estudio).

***Experimento 10.4: Ingresos vs traslados.***

¿Tienen el mismo impacto en la población de estudiantes el aumento del número de ingresos que el del número de traslados? Explore y compare el impacto de modificar el número de ingresos y traslados sobre la población final y los tiempos de estabilización.

***Experimento 10.5: Poblaciones por semestre.***

¿Qué condiciones hacen que las poblaciones de los diferentes semestres sean más parecidas? Explore el efecto de los parámetros de desempeño académico, traslados e ingresos para determinar en qué condiciones es menor la diferencia del número de estudiantes de los diferentes semestres.

Tabla 11.1 Parámetros del experimento 10, “Respuesta al escalón”

Título:	Ingreso constante		
Descripción:	Evolución del número de estudiantes de un programa curricular, con ingreso constante de estudiantes al primer semestre		
Créditos	Implementación		
	e-mail		
Parámetros			
Grupo	Nombre Modelica	Nombre	Descripción

<sup>2</sup>La información en las tablas siguientes se muestra tal como aparece en la base de datos. Por esta razón hay datos tanto en español como en inglés.

<b>Rendimiento</b>	<i>aprobacion</i>	Aprobación	Tasa de aprobación de las asignaturas
	<i>cancelacion</i>	Cancelación	Tasa de cancelación de asignaturas
	<i>perdidacalidad</i>	Deserción	Tasa de pérdida de la calidad de estudiante
<b>Plan de estudios</b>	<i>ingreso</i>	Ingreso	Número de estudiantes que ingresan a primer semestre
	<i>traslado</i>	Traslados	Número de trasladados desde otras carreras. Se distribuyen homogéneamente en todos los semestres
<b>Parámetros de simulación</b>	<i>stopTime</i>	Semestres	Número de semestres simulados

Tabla 11.2 Figuras del experimento 10, “Respuesta al escalón”

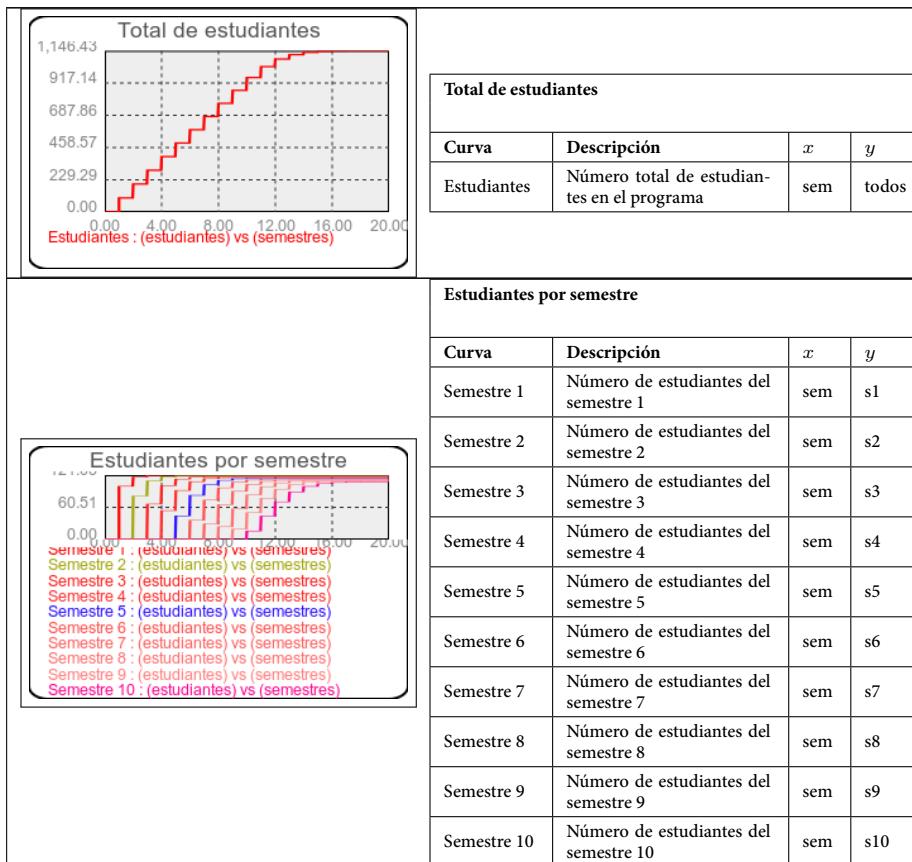


Tabla 11.3 Variables en la tabla de resultados del experimento 10, "Respuesta al escalón"

Variable	Descripción	Unidades
sem		semestres
todos		estudiantes
s1		estudiantes
s2		estudiantes
s3		estudiantes
s4		estudiantes
s5		estudiantes
s6		estudiantes
s7		estudiantes
s8		estudiantes
s9		estudiantes
s10		estudiantes

### Planta de experimentación 11: Respuesta al impulso.

**Presentación:** esta planta implementa el modelo reducido que se explica en la sección 11.1.1. La planta considera un plan de estudios nuevos al que ingresa una única cohorte. En otras palabras, la planta permite realizar simulaciones de la respuesta al impulso del modelo reducido. El número de estudiantes que ingresan al plan por concepto de traslados se distribuye uniformemente en todos los semestres.

El tiempo medio de graduación se calcula recursivamente. La gráfica respectiva permite véase la evolución de la recursividad. El tiempo medio de graduación  $\bar{G}_0$ , definido en la sección 11.1.6.1 es el valor estacionario de esa evolución.

**Instrumentación<sup>3</sup>:** el modelo cuenta con 6 parámetros ajustables organizados en 3 grupos de controles (véase tabla 11.4). Como resultado del experimento, el programa despliega:

- 3 curvas organizadas en 2 gráficos (véase tabla 11.5).
- Una tabla de datos del comportamiento de 4 variables (véase tabla 11.6).

**Experimentos sugeridos:** el siguiente es el listado de experimentos sugeridos:

***Experimento 11.1: Tiempo medio de graduación.***

¿Cómo afectan los parámetros de desempeño académico el tiempo medio de graduación? Explore el efecto que tiene la acción de modificar los parámetros de desempeño académico en el tiempo medio de graduación.

***Experimento 11.2: Dispersión del tiempo de graduación.***

¿Cómo afectan los parámetros de desempeño académico la dispersión del tiempo medio de graduación? Explore el efecto que tiene la acción de modificar los parámetros de desempeño académico en la dispersión del tiempo medio de graduación.

***Experimento 11.3: Tasa de graduación.***

¿De qué depende la tasas de graduación? Explore el efecto de los parámetros del desempeño académico en el número total de graduados.

***Experimento 11.4: Traslados.***

¿Qué efecto tienen los traslados sobre la graduación? Explore el impacto de los traslados sobre el tiempo medio de graduación, la dispersión del tiempo de graduación y el número total de graduados. Compare su impacto con el de los ingresos.

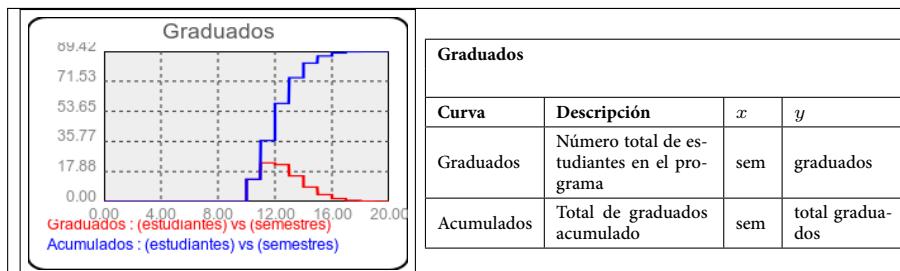
### **Experimento 11.5: Condiciones equivalentes de referencia.**

¿Las condiciones equivalentes de referencia implican graduaciones equivalentes? Explore si las combinaciones de parámetros que generan poblaciones iguales a las de referencia tienen condiciones de graduación también equivalentes a las de la situación de referencia.

Tabla 11.4 Parámetros del experimento 11, “Respuesta al impulso”

Título:	Una cohorte		
Descripción:	Evolución del número de estudiantes de un única cohorte en un programa curricular.		
Créditos	Implementación		
	e-mail		
<b>Parámetros</b>			
Grupo	Nombre Modelica	Nombre	Descripción
Rendimiento	<i>aprobacion</i>	Aprobación	Tasa de aprobación de las asignaturas
	<i>cancelacion</i>	Cancelación	Tasa de cancelación de asignaturas
	<i>perdidacalidad</i>	Deserción	Tasa de pérdida de la calidad de estudiante
Plan de estudios	<i>ingreso</i>	Ingreso	Número de estudiantes que ingresan a primer semestre
	<i>traslado</i>	Traslados	Número de trasladados desde otras carreras. Se distribuyen homogéneamente en todos los semestres
Parámetros de simulación	<i>stopTime</i>	Semestres	Número de semestres simulados

Tabla 11.5 Figuras del experimento 11, “Respuesta al impulso”



<sup>3</sup>La información en las tablas siguientes se muestra tal como aparece en la base de datos. Por esta razón hay datos tanto en español como en inglés.

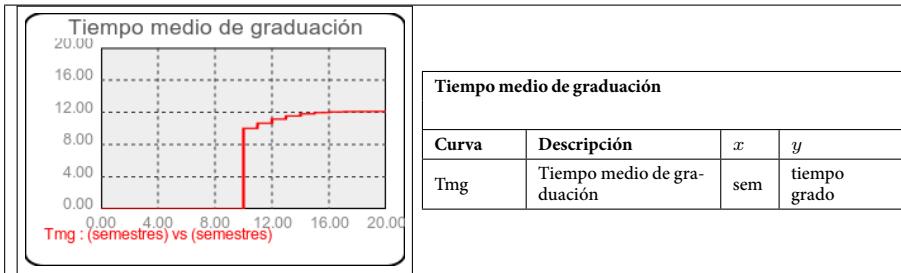


Tabla 11.6 Variables en la tabla de resultados del experimento 11, “Respuesta al impulso”

Variable	Descripción	Unidades
sem		semestres
graduados		estudiantes
total graduados		estudiantes
tiempo grado		semestres

### 11.3 LA IMPLEMENTACIÓN

Las ecuaciones 11.1 y 11.3 se implementan a través de la clase asignatura, (véase archivo asignatura.mo). Como se trata de una ecuación de diferencias, se hace uso de la estructura when del lenguaje Modelica para su implementación.

La clase plan (véase archivo plan.mo) implementa el plan de estudios completo. Para ello se construye un vector de objetos de la clase asignatura. Estos objetos están conectados entre sí, de tal manera que el número de estudiantes que aprueban la asignatura  $j$  en el periodo  $k$  se iguala al número de los que pasan a la asignatura  $j + 1$  en el periodo  $k + 1$ .

En la clase plan se ha supuesto que los parámetros de todas las asignaturas son iguales, lo que implica que se ha realizado una implementación del modelo reducido (véase sección 11.1.1). No obstante, si se desea utilizar el modelo detallado, basta con asignar los parámetros específicos a cada asignatura  $j$ , eventualmente, modificarlos a lo largo de la simulación, si es que estos no son constantes.

Las simulaciones de respuesta al escalón y respuesta al impulso se han realizado a través de las clases ingresoConstante y ingresoPulso, respectivamente (en los archivos ingresoConstante.mo e ingresoPulso.mo).

### 11.3.1 Listado de archivos

La tabla 11.7 muestra el listado de los archivos fuente de la implementación del modelo.

Tabla 11.7 Archivos del modelo

Número	Archivo
11.1	asignatura.mo
11.2	ingresoConstante.mo
11.3	ingresoPulso.mo
11.4	package.mo
11.5	plan.mo

Archivo 11.1 asignatura.mo

```
within Estudiantes;

model asignatura
  Real estudiantes(start=0);
  Real antiguos(start=0);
  Real nuevos(start=0);
  Real aprueban(start=0);
  Real noaprueban;
  Real cancelan;
  Real nocancelan;
  Real traslado;
  parameter Real aprobacion=1.0;
  parameter Real cancelacion=0.0;
  parameter Real perdidacalidad=0.00;
equation
  estudiantes=antiguos+nuevos+traslado;
  cancelan=estudiantes*cancelacion;
  nocancelan=estudiantes*(1-cancelacion);
  aprueban=nocancelan*aprobacion;
  noaprueban=nocancelan*(1-aprobacion);
  when sample(0,1) then
    antiguos=(pre(noprueban)+pre(cancelan))*(1-perdidacalidad);
  end when;
end asignatura;
```

## Archivo 11.2 ingresoConstante.mo

```

within Estudiantes;

model ingresoConstante
  extends plan;
  Modelica.Blocks.Sources.Step Entra(height=ingreso , startTime=1);
  parameter Poblacion ingreso=100;
equation
  connect(Ingresa , Entra.y);
end ingresoConstante;

```

## Archivo 11.3 ingresoPulso.mo

```

within Estudiantes;

model ingresoPulso
  extends plan;
  Modelica.Blocks.Sources.Pulse Entra(amplitude=ingreso , period=2,
    nperiod=1, startTime=1);
  Semestre tiempogrado(start=0);
  Poblacion totalgraduados(start=Modelica.Constants.small);
  parameter Poblacion ingreso=100;
equation
  connect(Ingresa , Entra.y);
  when sample(0,1) then
    totalgraduados=pre(totalgraduados)+graduados;
    tiempogrado=(pre(tiempogrado)*pre(totalgraduados)+time*graduados)/
      totalgraduados;
  end when;
end ingresoPulso;

```

## Archivo 11.4 package.mo

```

package Estudiantes
type Poblacion=Real(unit="estudiantes");
type Semestre=Real(unit="semestres");
end Estudiantes;

```

## Archivo 11.5 plan.mo

```

within Estudiantes;

model plan
  asignatura A[n](each aprobacion=aprobacion , each cancelacion=
    cancelacion , each perdidacalidad=perdidacalidad);
  parameter Integer n=10;
  Modelica.Blocks.Math.Sum total(nin=n);
  Modelica.Blocks.Sources.Step Traslados[n](each height=traslado /n);
  Poblacion Ingresa;
  Poblacion graduados;
  Poblacion s1 ,s2 ,s3 ,s4 ,s5 ,s6 ,s7 ,s8 ,s9 ,s10 ,todos;
  parameter Real aprobacion=0.85;

```

```
parameter Real cancelacion=0.03;
parameter Real perdidacalidad=0.01;
parameter Real traslado=0;
Semestre sem;
equation
when sample(0,1) then
  for i in 1:n-1 loop
    A[i+1].nuevos=pre(A[i].aprueban)*(1-A[i].perdidacalidad);
  end for;
end when;
graduados=A[n].aprueban;
connect(Ingreso,A[1].nuevos);
for i in 1:n loop
  connect(A[i].estudiantes,total.u[i]);
  connect(A[i].traslado,Traslados[i].y);
end for;
sem=time;
s1=A[1].estudiantes;
s2=A[2].estudiantes;
s3=A[3].estudiantes;
s4=A[4].estudiantes;
s5=A[5].estudiantes;
s6=A[6].estudiantes;
s7=A[7].estudiantes;
s8=A[8].estudiantes;
s9=A[9].estudiantes;
s10=A[10].estudiantes;
todos=total.y;
end plan;
```

# 12

## CENTRAL DE GENERACIÓN HIDROELÉCTRICA

*Se presenta un modelo de una central de generación hidroeléctrica con una carga eléctrica acoplada. El modelo incluye el control primario (control de frecuencia) mediante un controlador de tipo Proporcional-Integral (PI), así como la variación del volumen del embalse asociado.*

La obtención de grandes cantidades de energía eléctrica se realiza mediante la transformación de otras formas de energía. Una de las formas de energía que se utilizan para ello es la energía potencial gravitacional de grandes masas de agua. El proceso de transformación se lleva a cabo en las centrales hidroeléctricas de generación.

En estas centrales se utilizan máquinas sincrónicas que generan una diferencia de potencial eléctrico con forma sinusoidal (lo que se conoce



Figura 12.1 Central hidroeléctrica de Cheoah en Carolina del Norte, E.U.<sup>1</sup>

<sup>1</sup>Tomada de [link:15] con licencia de Creative Commons.

como corriente alterna). La frecuencia de esa señal sinusoidal está directamente relacionada con la velocidad de giro de la turbina que mueve la máquina sincrónica.

En este modelo se implementa un modelo simplificado de una central hidroeléctrica, que incluye el sistema que controla la velocidad de giro de la turbina, y por tanto la frecuencia de la señal de potencial sinusoidal generado. El controlador implementado es del tipo Proporcional-Integral (PI). La central suministra energía a una carga eléctrica, representada por un equivalente simple.

## 12.1 EL MODELO

### 12.1.1 Estructura de una central hidroeléctrica y transformaciones de energía

La figura 12.2 es un diagrama esquemático de una central hidroeléctrica genérica<sup>2</sup>. La energía eléctrica se obtiene por transformaciones sucesivas de la energía primaria, como se explica a continuación:

- Hay una masa de agua en el embalse que está a una altura superior a la del río. Respecto al nivel en el que está el río, esta masa de agua posee una energía potencial gravitacional.
- El agua entra al canal a través de la bocatoma y se desplaza a través del ducto en dirección a la turbina. En este proceso, la energía potencial gravitacional se transforma en energía hidráulica, gracias a la presión a la que está sometida el agua y a su velocidad de desplazamiento.
- La turbina es impulsada por el agua que llega a través del ducto. En este proceso, parte de la energía hidráulica que transporta el agua se transforma en energía cinética rotacional, en el conjunto turbina-generador.
- La turbina hace girar el rotor del generador, lo que origina una diferencia de potencial entre sus terminales. Si el generador está acoplado a la red de transmisión, entonces circula una corriente eléctrica, y la energía cinética rotacional se transforma en energía eléctrica.
- En la subestación la energía eléctrica sufre una transformación más, que consiste en elevar el nivel de tensión eléctrica a los valores adecuados para su transmisión a largas distancias. En este proceso, la energía eléctrica

---

<sup>2</sup>Existe otro tipo de centrales que no utilizan embalses, sino que aprovechan caudales naturales de agua. Tales centrales se denominan *a filo de agua* y no se tratan en este documento.

que proviene del generador se transforma en energía eléctrica a otro nivel de tensión.

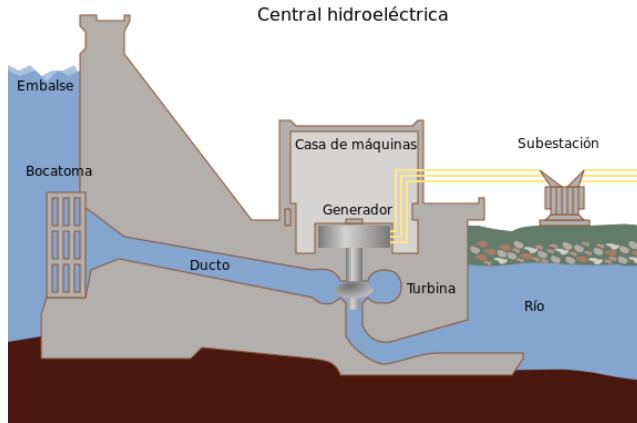


Figura 12.2 Diagrama esquemático de una central hidroeléctrica<sup>3</sup>

### 12.1.2 El embalse

El propósito del embalse es almacenar energía para utilizarla cuando se requiera. La estrategia consiste en contener una masa de agua a una altura superior a aquella en la que se ubica el conjunto turbina-generador. En esas condiciones, el agua tiene una energía potencial gravitacional que se puede mantener almacenada.

#### 12.1.2.1 Volumen de agua

El volumen de agua almacenado  $V$ , puede calcularse en función de la altura del nivel de agua del embalse como

$$V(h) = \int_{h_{min}}^h A(h)dh$$

Supóngase que la forma del embalse es la de un tronco de pirámide invertida con sección transversal cuadrada. El lado  $x(h)$  puede calcularse como

$$x(h) = x_{min} + \alpha(h - h_{min})$$

---

<sup>3</sup> Adaptada de [link:16] con licencia de Creative Commons.

en donde  $\alpha$  depende de la geometría del embalse:

$$\alpha = \frac{x_{max} - x_{min}}{h_{max} - h_{min}}$$

En esas condiciones  $A(h) = x^2(h)$  y por tanto

$$V(h) = \int_{h_{min}}^h x^2(h) dh = \int_{h_{min}}^h [(x_{min} + \alpha(h - h_{min}))^2] dh$$

Efectuando un cambio de variable:

$$\bar{h} = h - h_{min} \quad d\bar{h} = dh$$

$$V(\bar{h}) = \int_0^{\bar{h}} [(x_{min} + \alpha\bar{h})^2] d\bar{h}$$

$$V(\bar{h}) = \int_0^{\bar{h}} (x_{min}^2 + 2\alpha x_{min}\bar{h} + \alpha^2\bar{h}^2) d\bar{h}$$

$$V(\bar{h}) = x_{min}^2\bar{h} + \alpha x_{min}\bar{h}^2 + \frac{\alpha^2}{3}\bar{h}^3 \Big|_0^{\bar{h}}$$

$$V(\bar{h}) = x_{min}^2\bar{h} + \alpha x_{min}\bar{h}^2 + \frac{\alpha^2}{3}\bar{h}^3 \quad (12.1)$$

Un caso particular sucede cuando el embalse tiene forma de prisma cuadrado. En esas condiciones el área no varía con la altura ( $A(h) = A$ ) o, lo que es igual,  $\alpha = 0 \therefore V(\bar{h}) = x_{min}^2\bar{h}$ .

### 12.1.2.2 Variación de la altura de agua

El caudal neto  $Q(t)$  es la velocidad con que varía el volumen del agua almacenada:

$$Q(t) = \frac{dV(h)}{dt} = \frac{dV(h)}{dh} \frac{dh}{dt}$$

Si la forma del embalse es de tronco de pirámide cuadrada invertida, entonces se puede usar la ecuación 12.1:

$$Q(t) = (x_{min}^2 + 2\alpha x_{min}\bar{h} + \alpha^2\bar{h}^2) \frac{d\bar{h}}{dt}$$

$$\frac{dh}{dt} = \frac{Q(t)}{x_{min}^2 + 2\alpha x_{min}\bar{h} + \alpha^2\bar{h}^2}$$

Si el embalse tiene forma de prisma cuadrado, entonces  $\frac{dh}{dt} = \frac{Q(t)}{x_{min}^2}$ .

### 12.1.3 El ducto

Se define  $T_w$  como el tiempo que tarda la columna de agua en acelerar el agua que fluye en el ducto desde cero hasta la velocidad de diseño. Es posible calcular  $T_w$  como (véanse [FCDS08, NJV12]):

$$T_w = \frac{LU_m}{gH_m} = \frac{L \frac{Q_m}{\pi \times (D/2)^2}}{gH_m}$$

en donde  $L$  es la longitud del ducto,  $U_m$  es la velocidad media del agua,  $H_m$  la cabeza neta de presión,  $Q_m$  es el caudal medio y  $D$  es el diámetro del ducto.

Los fenómenos dinámicos al interior del ducto se modelan a partir de la ecuación de continuidad y la ecuación de momento. Un modelo simplificado de esos fenómenos se presenta en [Pri06]: la relación entre el caudal  $Q$  y la cabeza de presión  $H$  se modela como un sistema lineal de primer orden no minifase. La función de transferencia  $F_{QH}(s)$  resulta ser:

$$F_{QH}(s) = \frac{Q(s)}{H(s)} = \frac{b(s)}{a(s)} = \frac{1 - sT_w}{1 + \frac{sT_w}{2}}$$

que corresponde a la siguiente relación en el dominio del tiempo

$$Q(t) + \frac{T_w}{2} \frac{dQ}{dt} = H(t) - T_w \frac{H(t)}{dt}$$

### 12.1.4 La turbina

En [Hel11] se presenta un modelo dinámico linealizado de la turbina, que relaciona el caudal  $Q$ , la apertura de la válvula  $y$ , la cabeza hidráulica en la turbina  $h$ , la velocidad de giro  $w$ , el torque  $\tau$  y la potencia transformada  $p$ :

$$\left\{ \begin{array}{l} dQ = \frac{\partial Q}{\partial w}w + \frac{\partial Q}{\partial y}y + \frac{\partial Q}{\partial h}h \\ d\tau = \frac{\partial \tau}{\partial w}w + \frac{\partial \tau}{\partial y}y + \frac{\partial \tau}{\partial h}h \\ dp = \frac{\partial p}{\partial w}w + \frac{\partial p}{\partial y}y + \frac{\partial p}{\partial h}h \end{array} \right.$$

Sin embargo, es usual utilizar modelos estáticos mucho más simples. Algunos de ellos involucran la variación de la eficiencia de la turbina en función del punto de operación, es decir, según el caudal que recibe la turbina

y la velocidad de giro. Modelos aún más simplificados cercanos al punto de operación establecen una relación lineal estática entre el caudal  $q(t)$  y el par generado  $\tau(t)$ :

$$\tau(t) = G_\tau q(t)$$

### 12.1.5 El generador

El generador es la máquina encargada de realizar la conversión de energía mecánica rotacional en energía eléctrica. Típicamente se trata de una máquina sincrónica trifásica. A manera de ejemplo, la figura 12.3 muestra una foto de la casa de máquinas de la central Hoover, en Estados Unidos, y el rotor de uno de los generadores.

El adjetivo *sincrónico* significa que la onda de tensión sinusoidal inducida tiene una frecuencia  $f_e$ , que está sincronizada con la velocidad angular  $n_m$  a la que rota el eje de la máquina. La relación es

$$f_e = \frac{n_m P}{120} \quad \omega_e = 2\pi f_e \quad T_e = \frac{1}{f_e} \quad (12.2)$$

en donde  $P$  es el número de polos de la máquina,  $f_e$  está en medida en hertz,  $n_m$  en r.p.m.,  $\omega_e$  en rad/s y  $T_e$  en segundos.  $T_e$  es el periodo de la tensión inducida y  $\omega_e$  es su velocidad angular.

¿A qué se refiere el *número de polos*? Usualmente, estas máquinas tienen el circuito de campo en el rotor y el circuito de armadura en el estator. La figura 12.4 muestra un diagrama de una máquina trifásica sincrónica: acoplado al rotor hay un devanado que cumple el papel de un electroimán (en la figura 12.4 se identifica con el color amarillo). El campo magnético creado por este circuito atraviesa tres devanados acoplados al estator (en la figura 12.4 se identifican con los colores rojo, azul y verde). Al girar, el flujo magnético que atraviesa cada uno de esos devanados varía, lo que induce una diferencia de potencial eléctrico entre los terminales de cada devanado. El mismo efecto se logra si en lugar de distribuir 3 devanados en los  $360^\circ$ , se distribuyen  $3P$  devanados ( $3, 6, 9, \dots$  devanados).  $P$  es entonces el *número de polos* con los que se ha construido la máquina.

¿Qué significa el adjetivo *trifásico*? El diseño geométrico de la máquina es tal que, si el rotor gira a una velocidad angular constante, la tensión inducida en cada uno de los devanados es sinusoidal. Como los devanados están ubicados en el estator, desfasados  $120^\circ$  geométricos, las tensiones inducidas estarán desfasadas  $120^\circ$  eléctricos.

$$\left\{ \begin{array}{l} E_{A_1} = E_A \cos\left(\frac{\omega_e}{m}t\right) \\ E_{A_2} = E_A \cos\left(\frac{\omega_e}{m}t - 120^\circ\right) \\ E_{A_3} = E_A \cos\left(\frac{\omega_e}{m}t - 240^\circ\right) \end{array} \right. \quad (12.3)$$

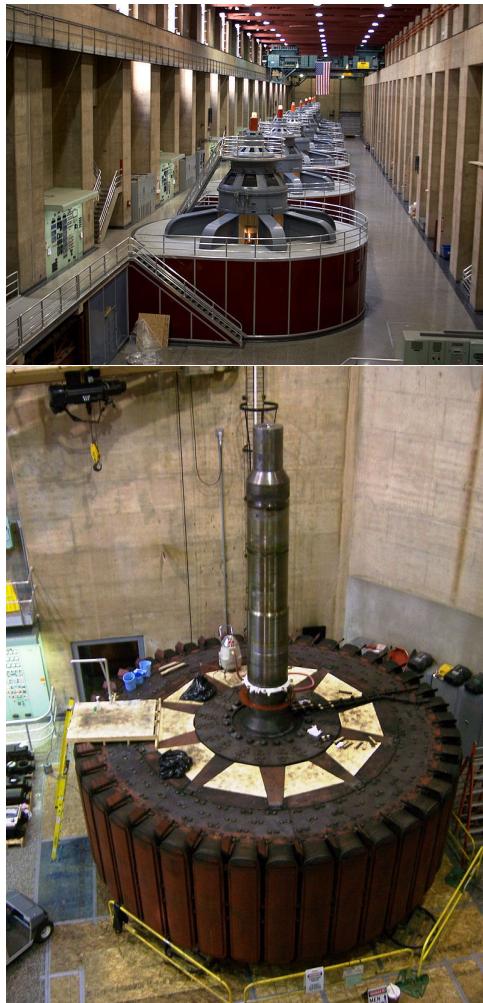
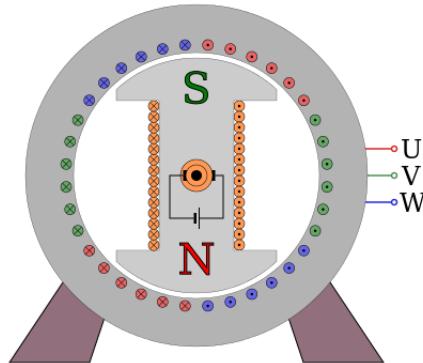


Figura 12.3 Arriba: casa de máquinas de la central asociada a la presa Hoover en Estados Unidos. Abajo: rotor de uno de los generadores<sup>4</sup>

Figura 12.4 Diagrama de una máquina sincrónica<sup>5</sup>

El conjunto de tensiones de la ecuación 12.3 se muestra en la figura 12.5. Se trata de un sistema *trifásico* de tensiones porque tienen las siguientes propiedades:

- Son tres tensiones sinusoidales.
- La frecuencia de las tres tensiones es la misma.
- La amplitud de las tres tensiones es la misma.
- El desfase relativo entre las tres tensiones es de  $1/3$  de ciclo.

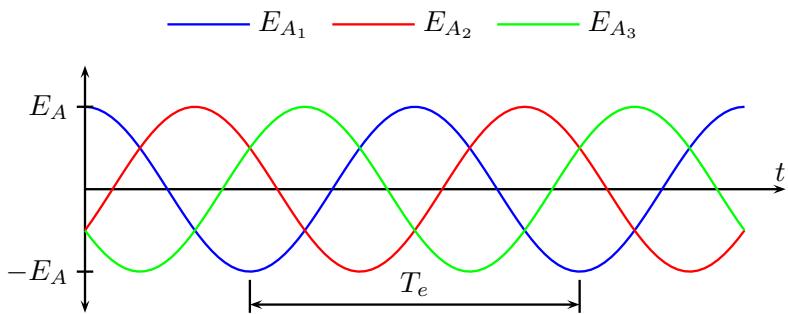


Figura 12.5 Sistema trifásico de tensiones inducidas en el circuito de armadura

<sup>4</sup>Tomadas de [link:17] y de [link:18] con licencia de Creative Commons.

<sup>5</sup>Tomada de [link:19] con licencia de Creative Commons.

El valor de la amplitud de la tensión inducida por fase  $E_A$  es directamente proporcional al flujo magnético  $\phi$  y a la velocidad angular  $\omega$ :

$$E_A = K_F \phi n_m = K \phi \omega_e \quad K = K_F \frac{120}{2\pi P}$$

en donde  $K_F$  es una constante que depende de factores constructivos. Puede reescribirse entonces 12.3 como

$$\left\{ \begin{array}{l} E_{A_1} = K \phi \omega_e \cos\left(\frac{\omega_e}{m} t\right) \\ E_{A_2} = K \phi \omega_e \cos\left(\frac{\omega_e}{m} t - 120^\circ\right) \\ E_{A_3} = K \phi \omega_e \cos\left(\frac{\omega_e}{m} t - 240^\circ\right) \end{array} \right. \quad (12.4)$$

La figura 12.6 muestra el circuito equivalente de la máquina trifásica sincrónica: a la izquierda se representa el circuito de campo, que se alimenta con una fuente de tensión continua  $V_F$ . Mediante la resistencia ajustable  $R_{adj}$  se controla la cantidad de corriente que circula y, por tanto, la magnitud del flujo magnético  $\phi$ . A la derecha se muestran los tres circuitos de armadura, uno por cada fase. Las tres tensiones entre los terminales de armadura son  $V_{\varphi_1}$ ,  $V_{\varphi_2}$  y  $V_{\varphi_3}$ .

### 12.1.6 Los lazos de control

Uno de los propósitos del sistema de generación es asegurar que el sistema trifásico de tensiones tenga una buena *calidad de potencia*. Esto significa, entre otras cosas, que las ondas generadas tengan:

- Una frecuencia de la onda constante.
- Una amplitud de tensión constante.

En las ecuaciones 12.2 y 12.4 y se observa lo siguiente:

- La frecuencia de la onda ( $\omega_e$ ) depende de la velocidad de giro de la turbina ( $n_m$ ).
- La amplitud de la tensión ( $K \phi \omega_e$ ) depende del flujo magnético  $\phi$  y de la frecuencia de la onda ( $\omega_e$ ).

Por esta razón, la estrategia de control usual(Ctl) de un sistema de generación (véase figura 12.7) tiene dos componentes:

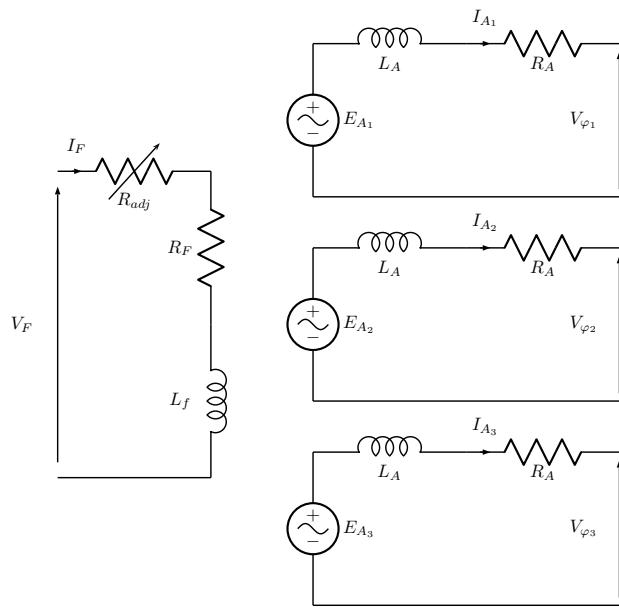


Figura 12.6 Circuito equivalente de una máquina sincrónica

- Ctl 1. Control de frecuencia: busca asegurar que la frecuencia de la onda generada sea constante. Para lograrlo, se utiliza un sistema de realimentación que mide la frecuencia de giro de la turbina (o la frecuencia de la onda generada) y corrige el par aplicado a la turbina para acelerarlo o frenarlo, según se requiera. En el caso de una central hidroeléctrica, el par se corrige abriendo o cerrando la válvula que controla el caudal que llega a la turbina. Este lazo de control se conoce como *control primario*, y el sistema que lo implementa se conoce como el *gobernador*.
- Ctl 2. Control de tensión: busca asegurar que la amplitud de tensión sea constante. Para lograrlo, se utiliza un sistema de realimentación que mide la amplitud de la tensión en los terminales de armadura y corrige la magnitud del campo magnético  $\phi$  para aumentarlo o disminuirlo según se requiera. El campo magnético se corrige controlando la tensión de alimentación del circuito de campo ( $V_F$ ). Este lazo de control se conoce como *control secundario*, y el controlador de este lazo se suele denominar el *regulador automático de tensión* (Automatic Voltage Regulator [AVR]).

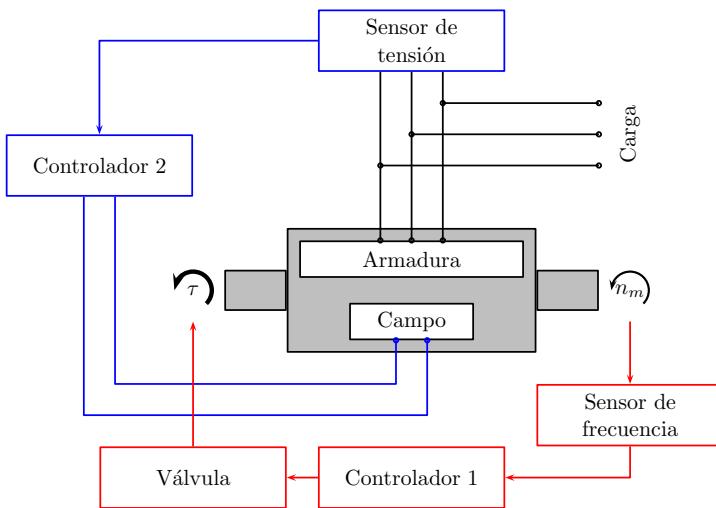


Figura 12.7 Estrategia de control de generación. En rojo está el lazo de realimentación para control de frecuencia; en azul, el lazo de realimentación para control de tensión

El diseño y ajuste de los dos lazos de control se realiza para condiciones cercanas al punto de operación, es decir, para corregir pequeñas modificaciones en la frecuencia y en la amplitud de la onda de tensión generada. Cuando estas variaciones no son pequeñas, se considera una condición anormal de operación y el sistema de protecciones (no cubierto en este documento) debe entrar a operar.

### 12.1.7 La carga eléctrica

En una carga eléctrica, la relación entre potencia  $p(t)$ , tensión  $v(t)$  y corriente  $i(t)$  es

$$p(t) = v(t)i(t)$$

y el consumo de energía entre dos instantes de tiempo  $t_1$  y  $t_2$  está dado por:

$$e(t_1, t_2) = \int_{t_1}^{t_2} p(t)dt$$

En sistemas alimentados por fuentes sinusoidales de frecuencia constante y cargas lineales, el análisis en el dominio de la frecuencia compleja permite escribir

$$\mathbf{V} = \mathbf{Z}\mathbf{I}$$

en donde  $\mathbf{V}$  e  $\mathbf{I}$  son los fasores de tensión y corriente, respectivamente, y  $\mathbf{Z}$  es la impedancia de la carga. Estas tres magnitudes son números complejos que pueden escribirse en formato polar o rectangular:

$$\begin{aligned}\mathbf{V} &= V \angle \theta_V = V_R + jV_I \\ \mathbf{I} &= I \angle \theta_I = I_R + jI_I \\ \mathbf{Z} &= Z \angle \theta_Z = R + jX\end{aligned}$$

$$\mathbf{S} = \mathbf{VI}^* = P + jQ$$

Las cargas alimentadas por un sistema de potencia no son constantes. La variación del consumo de los usuarios presenta cambios fuertes a lo largo de un día y de un día a otro. Por otra parte, la demanda de potencia (activa y reactiva) puede verse influenciada por las variaciones de la tensión de alimentación, tanto en amplitud como en frecuencia.

Para modelar estas dependencias entre la potencia demandada y la tensión de alimentación, es usual emplear dos modelos (véase [Han06]):

a. Modelo de dependencia lineal:

$$\begin{aligned}P &= P_{nominal} + \frac{\partial P}{\partial |\bar{V}|} \Delta |\bar{V}| + \frac{\partial P}{\partial f} \Delta f \\ Q &= Q_{nominal} + \frac{\partial Q}{\partial |\bar{V}|} \Delta |\bar{V}| + \frac{\partial Q}{\partial f} \Delta f\end{aligned}$$

Acá,  $P$  y  $Q$  son las demandas de potencia activa y reactiva,  $P_{nominal}$  y  $Q_{nominal}$  sus valores nominales,  $\frac{\partial P}{\partial |\bar{V}|}$ ,  $\frac{\partial P}{\partial f}$ ,  $\frac{\partial Q}{\partial |\bar{V}|}$  y  $\frac{\partial Q}{\partial f}$  sus tasas de cambio frente a variaciones de la amplitud y frecuencia de la tensión de alimentación, y  $\Delta |\bar{V}|$ ,  $\Delta f$  son las desviaciones de amplitud y frecuencia respecto a los valores nominales.

b. Modelo de dependencia exponencial:

$$P = P_{nominal} |\bar{V}|^{pv} f^{pf}$$

$$Q = Q_{nominal} |\bar{V}|^{qv} f^{qf}$$

Acá,  $|\bar{V}|$  y  $f$  son los valores p.u. (*por unidad*) de amplitud y frecuencia de alimentación, y  $pv$ ,  $pf$ ,  $qv$  y  $qf$  son los parámetros del modelo exponencial.

## 12.2 PLANTAS DE EXPERIMENTACIÓN Y EXPERIMENTOS SUGERIDOS

### Planta de experimentación 12: Central hidroeléctrica con control PI de frecuencia.

**Presentación:** la planta permite explorar el control de frecuencia de una central hidroeléctrica. Se han modelado la carga eléctrica conectada al generador como un valor conocido. La carga es una potencia activa que varía en el tiempo.

El control de la frecuencia se obtiene mediante el control del sistema hidráulico que mueve la turbina. El modelo de este sistema hidráulico es el de un sistema dinámico no minifase de primer orden.

El control se realiza mediante un controlador del tipo PI, que utiliza como referencia la frecuencia de la señal de voltaje generado.

**Instrumentación<sup>6</sup>:** el modelo cuenta con 10 parámetros ajustables organizados en 3 grupos de controles (véase tabla 12.1). Como resultado del experimento, el programa despliega:

- 4 curvas organizadas en 4 gráficos (véase tabla 12.2).
- 1 animación en 2D (véase tabla 12.3).
- Una tabla de datos del comportamiento de 6 variables (véase tabla 12.4).

**Experimentos sugeridos:** el siguiente es el listado de experimentos sugeridos:

#### *Experimento 12.1: Control de frecuencia.*

¿Cómo afectan los parámetros del controlador la frecuencia de la señal generada? Explore el impacto de los dos parámetros del controlador (ganancia proporcional k y tiempo de integración Ti) sobre el comportamiento de la frecuencia de la señal de voltaje generado.

***Experimento 12.2: Efecto de la demanda.***

¿Cómo cambia el comportamiento de la planta al modificar las condiciones de potencia demandada? En el modelo implementado, la demanda tiene una forma trapezoidal, cuyos valores mínimo y máximo se pueden modificar. Explore el efecto de estos dos parámetros en el comportamiento de la planta.

***Experimento 12.3: Embalse.***

¿Cuál es el efecto de los parámetros del embalse? En el modelo implementado, el embalse tiene una sección transversal homogénea (forma de prisma). Explore cuál es el efecto de los parámetros del embalse (área, altura inicial, tiempo de arranque de agua), en el comportamiento de la planta.

***Experimento 12.4: Planta sin control de frecuencia.***

¿Cómo se comportaría una central hidroeléctrica sin control primario? Implemente un modelo de planta sin control primario, y analice su comportamiento.

***Experimento 12.5: Oscilaciones de frecuencia.***

¿Qué aspectos inciden en la aparición de oscilaciones de frecuencia, y en las características de esa oscilación? Explore bajo qué condiciones aparecen oscilaciones en la señal de frecuencia, y determine las variables que inciden en la frecuencia y amplitud de esas oscilaciones.

***Experimento 12.6: Planta con control proporcional.***

¿Cómo se comportaría una central hidroeléctrica con control primario del tipo P (proporcional)? Implemente un modelo de planta con control primario del tipo P (proporcional), y analice su comportamiento.

Tabla 12.1 Parámetros del experimento 12, “Central hidroeléctrica con control PI de frecuencia”

Título:	Hidroeléctrica		
Descripción:	Modelo de una central de generación hidroeléctrica con control primario (control de frecuencia) mediante un controlador de tipo Proporcional-Integral (PI)		
Créditos	Implementación e-mail		
Parámetros			
Grupo	Nombre Modelica	Nombre	Descripción
Controlador	$K$	Proporcional	Ganancia proporcional
	$Ti$	Integral	Tiempo del control integral
Embalse	$AreaEmbalse$	Área	Área de la sección transversal del embalse en hectáreas (hectómetros cuadrados)
	$yo$	Nivel inicial	Nivel inicial del embalse en metros
	$Tw$	Tiempo de arranque del agua	Tiempo de inicio de agua
	$geoFlag$	Perfil	Perfil del embalse
	$areaFactor$	Factor de área	Relación entre área del fondo del embalse y área del nivel inicial
	$Operacion$	Carga base	Referencia constante de caudal
Demanda	$MinPower$	Mínimo	Demandada de potencia mínima en MW
	$MaxPower$	Máximo	Demandada de potencia máxima en MW

<sup>6</sup>La información en las tablas siguientes se muestra tal como aparece en la base de datos. Por esta razón hay datos tanto en español como en inglés.

Tabla 12.2 Figuras del experimento 12, "Central hidroeléctrica con control PI de frecuencia"

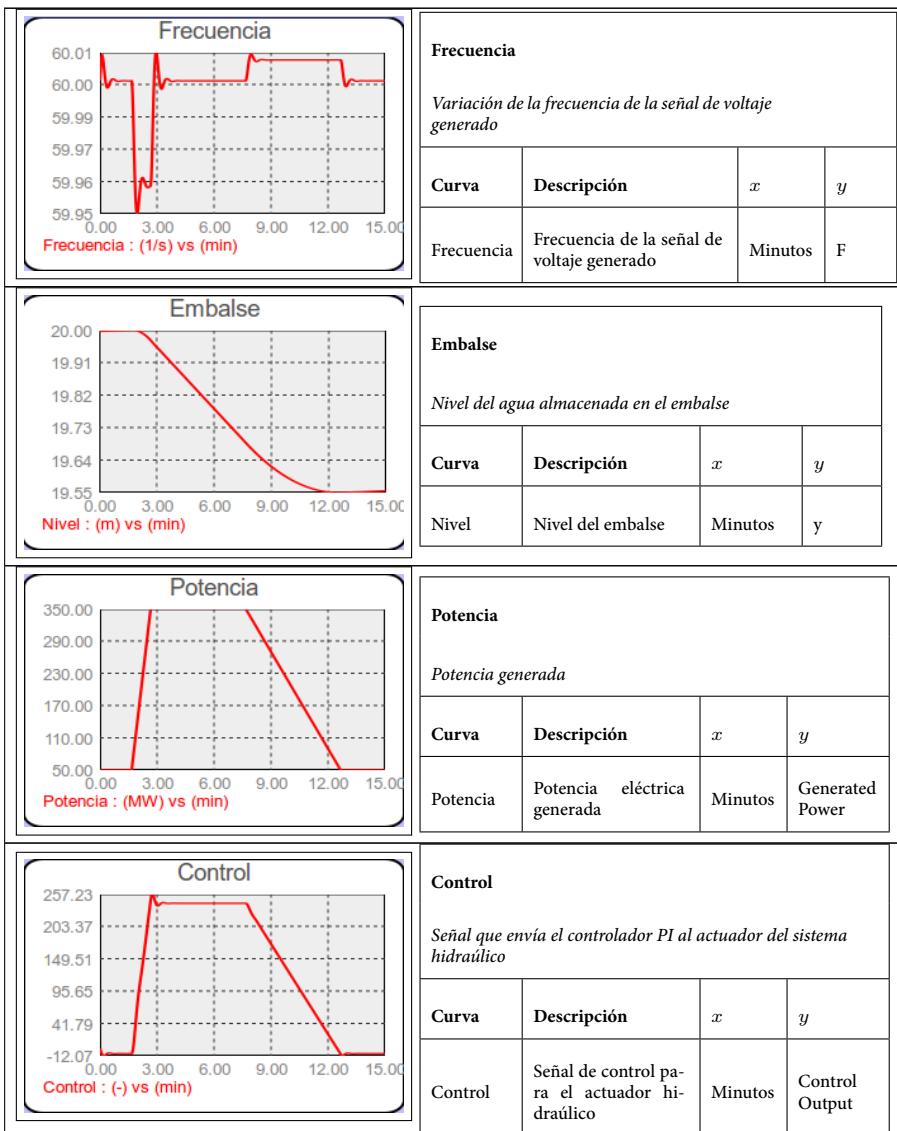


Tabla 12.3 Animaciones del experimento 12, “Central hidroeléctrica con control PI de frecuencia”

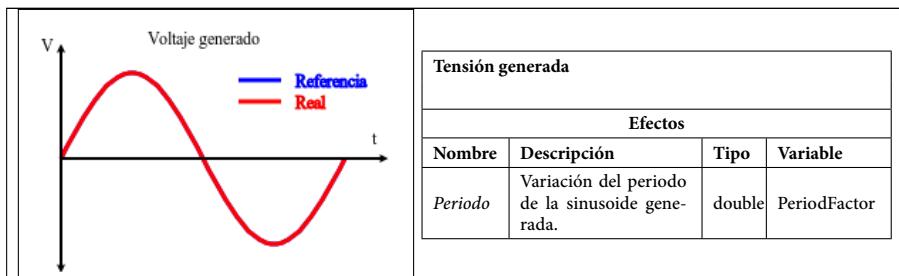


Tabla 12.4 Variables en la tabla de resultados del experimento 12, “Central hidroeléctrica con control PI de frecuencia”

Variable	Descripción	Unidades
minutes		min
F		1/s
y		m
GeneratedPower		MW
ControlOutput		-
PeriodFactor		

### 12.3 LA IMPLEMENTACIÓN

La implementación se ha basado en la librería PowerSystems<sup>7</sup>. Esta librería ha sido desarrollada sobre Dymola, y no es completamente compatible con OpenModelica. Específicamente, la implementación es una adaptación de uno de los ejemplos de prueba de la librería<sup>8</sup>. Para que el ejemplo inicial pudiera ser compilado bajo OpenModelica fue necesario realizar las siguientes modificaciones sobre la librería:

1. En la línea 118 del archivo package.mo se eliminó la declaración replaceable del paquete PackagePhaseSystem<sup>9</sup>.
2. En las líneas 165, 238, 303, 391 del archivo generic.mo se adicionó la referencia Connections a la función isRoot(terminal.theta)<sup>10</sup>.

La figura 12.8 muestra la estructura de la implementación del archivo HydroPI.mo, que se explica a continuación:

<sup>7</sup>[link:20].

<sup>8</sup>Véase PowerSystems.Examples.PowerWorld.Test.HydroPlantTest1.

<sup>9</sup>La instrucción resultante es package PackagePhaseSystem=.

<sup>10</sup>La instrucción resultante es Connections.isRoot(terminal.theta).

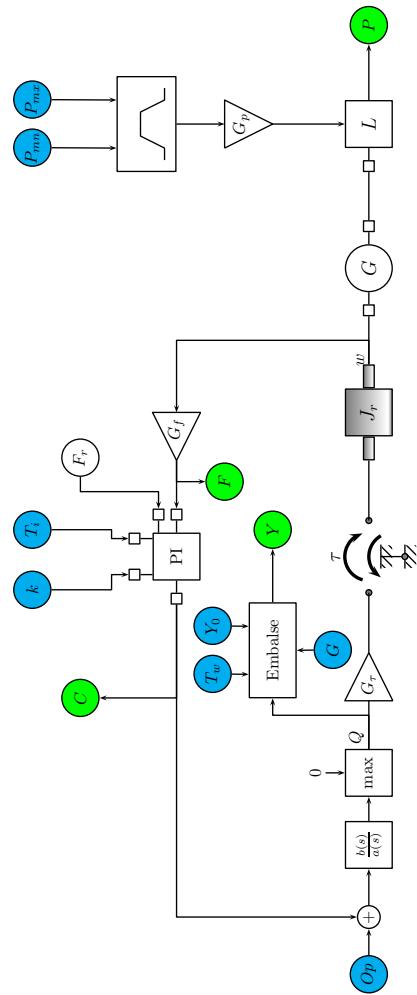


Figura 12.8 Implementación del modelo de una central hidroeléctrica con controlador de frecuencia tipo PI. En azul están los parámetros ajustables de la planta en la implementación de UNVirtualLab, y en verde, las variables graficadas

- El generador  $G$  está acoplado a una carga eléctrica  $L$ . Esta carga demanda una potencia que no es constante. El generador se ha implementado como un objeto de la clase `PowerSystems.Generic.EMF` y la carga como uno de la clase `PowerSystems.Generic.PrescribedLoad`.

- La curva de demanda de potencia se muestra en la figura 12.9. Los tiempos en los que cambia la curva están fijos ( $t_1 = 100s$ ,  $t_2 = 160s$ ,  $t_3 = 460s$ ,  $t_4 = 760s$ ), en tanto que los valores de potencia mínima y máxima pueden ser ajustados con las entradas  $P_{mn}$  y  $P_{mx}$ .

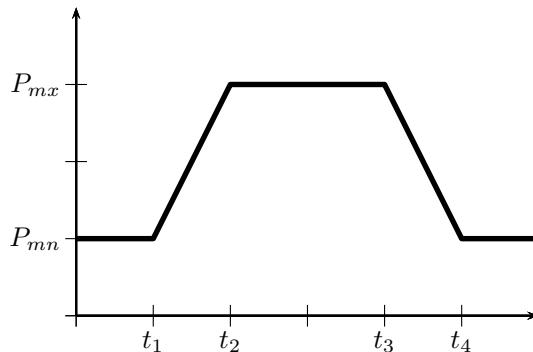


Figura 12.9 Curva de demanda de potencia

- El generador también está acoplado a un momento de inercia  $J_r$  que modela las inercias rotacionales combinadas de la turbina y el rotor del generador. El conjunto turbina-rotor es impulsado por un torque  $\tau$ .
- El torque  $\tau$  se ha modelado como una variable directamente proporcional al caudal  $Q$ :

$$\tau = G_\tau Q$$

- Se ha supuesto que el embalse tiene una sección transversal constante, es decir, que el volumen del agua acumulada puede calcularse como el producto de  $A$  (el área de esa sección) por la altura del nivel de agua. En esas condiciones, el nivel del embalse  $Y$  satisface:

$$\frac{dY}{dt} = -\frac{Q}{A} \quad Y = Y_0 + \frac{1}{A} \int_0^t Q dt$$

Por esta razón se ha utilizado un integrador para calcular el nivel del embalse. El integrador utiliza los parámetros ajustable  $A$  y  $Y_0$ .

- El rotor gira a una velocidad angular  $w$ . La velocidad angular se mide y se convierte en Hertz al multiplicarla por  $G_f = 1/2\pi$ .
- El controlador  $PI$  compara la frecuencia  $F$  con la frecuencia de referencia  $F_r$  y envía una señal al actuador del sistema hidráulico para

modificar el caudal. El controlador se ha implementado como un objeto de la clase Modelica.Blocks.Continuous.LimPID que incorpora un limitador de salida y estrategias *antiwindup*. Los parámetros  $k$  y  $T_i$  definen la salida  $C$  del controlador:

$$C = k \left( (F_r - F) + \frac{1}{T_i} \int_0^t (F_r - F) dt \right)$$

- La respuesta del sistema hidráulico, es decir del conjunto actuador, válvula y tubería, se ha modelado como un sistema dinámico no minifase de primer orden (véase [Pri06]):

$$\frac{Q(s)}{C(s)} = \frac{b(s)}{a(s)} = \frac{1 - sT_w}{1 + \frac{sT_w}{2}} \quad \therefore \quad Q(t) + \frac{T_w}{2} \frac{dQ}{dt} = C(t) - T_w \frac{C(t)}{dt}$$

$T_w$  se conoce como el tiempo de arranque del agua. Este tiempo es función de las características geométricas de la tubería, entre otras cosas. En este modelo se han omitido esas relaciones.

### 12.3.1 Listado de archivos

La tabla 12.5 muestra el listado de los archivos fuente de la implementación del modelo.

Tabla 12.5 Archivos del modelo

Número	Archivo
12.1	HidroPI.mo

Archivo 12.1 HidroPI.mo

```
model HidroPI
  extends PowerSystems.Generic.Ports.PartialSource(final
    potentialReference=false);
  parameter Real primaryControlMax(unit="MW") = 2000 "Maximum power
    for primary frequency control";
  Modelica.Mechanics.Rotational.Sources.Torque reservoirTurbine;
  Modelica.Mechanics.Rotational.Components.Inertia rotor(J=1e6, w(fixed
    =true, start=2*pi*Fref));
  PowerSystems.Generic.EMF generator(redeclare package PhaseSystem =
    PhaseSystem, definiteReference=definiteReference);
  Modelica.Mechanics.Rotational.Sensors.SpeedSensor angularVelocity;
  Modelica.Blocks.Sources.Constant reference(k=Fref);
  Modelica.Blocks.Continuous.LimPID primaryControl(controllerType=
    Modelica.Blocks.Types.SimpleController.PI,
```

```

        Ti=Ti ,k=K,yMax=
        primaryControlMax
    );
Modelica.Blocks.Math.Gain powerControl(k=1e6/50/2/pi);
Modelica.Blocks.Continuous.TransferFunction controlDynamics(a={Tw
    /2,1}, b={-Tw,1});
Modelica.Blocks.Math.Gain frequency(k=1/(2*pi));
PowerSystems.Generic.PrescribedPowerLoad prescribedLoad;
Modelica.Blocks.Sources.Trapezoid trapezoid(startTime=100,
                                              offset=MinPower,
                                              amplitude=MaxPower-
                                              MinPower,
                                              width=300,
                                              falling=300,
                                              nperiod=1,
                                              rising=60,
                                              period=900);

Modelica.Blocks.Math.Gain MW2W(k=1e6);
Modelica.Blocks.Math.Add add;
Modelica.Blocks.Sources.Constant cte1(k=Operacion);
Modelica.Blocks.Sources.Constant cte2(k=0);
Modelica.Blocks.Math.Max max;
parameter Real Tw(unit="s")=2;
parameter Real Fref=60;
parameter Real yo=20;
parameter Real Ti=5;
parameter Real K=50/0.2;
parameter Real pi=Modelica.Constants.pi;
parameter Real MinPower=50;
parameter Real MaxPower=350;
parameter Real Operacion=50;
parameter Real Qf=50;
// geometría
parameter Integer geoFlag=1;
parameter Real AreaEmbalse=5 "en hm2";
parameter Real Amax=AreaEmbalse*1e4;
parameter Real areaFactor=0.5;
parameter Real Hmax=20;
parameter Real Hmin=10;
/
Real y(unit="m", start=yo, fixed=true);
Real Period(unit="s");
Real F(unit="1/s");
Real PrescribedPower(unit="MW");
Real GeneratedPower(unit="MW");
Real minutes(unit="min");
Real ControlOutput(unit="-");
Real PeriodFactor;
Real Constant(start=1.0, fixed=true);
Real Volumen(unit="m^3", start=volumen(geoFlag, yo-Hmin, Amax,
                                         areaFactor, Hmax-Hmin), fixed=true);
equation
  der(Volumen)=-controlDynamics.y;
  Volumen=volumen(geoFlag, y-Hmin, Amax, areaFactor, Hmax-Hmin);

```

```
connect(rotor.flange_b, angularVelocity.flange);
connect(reservoirTurbine.flange, rotor.flange_a);
connect(rotor.flange_b, generator.flange);
connect(powerControl.y, reservoirTurbine.tau);
connect(controlDynamics.y, powerControl.u);
connect(cte1.y, add.u1);
connect(primaryControl.y, add.u2);
connect(add.y, max.u1);
connect(cte2.y, max.u2);
connect(max.y, controlDynamics.u);
connect(reference.y, primaryControl.u_s);
connect(angularVelocity.w, frequency.u);
connect(frequency.y, primaryControl.u_m);
connect(generator.terminal, terminal);
connect(trapezoid.y, MWZW.u);
connect(MWZW.y, prescribedLoad.P);
connect(terminal, prescribedLoad.terminal);
der(Constant)=0;
Period=1/F;
F=frequency.y;
PeriodFactor=reference.k/F;
PrescribedPower=trapezoid.y;
GeneratedPower=-generator.S[1]/MWZW.k;
minutes=time/60;
ControlOutput=primaryControl.y;
end HidroPI;
```

# 13

## DINÁMICA DE SISTEMAS. MODELO DEL MUNDO

*El modelo corresponde a la implementación de Cellier ([Cel08]) del modelo de la dinámica del mundo propuesto por Meadows ([MIM<sup>+</sup>74]), sobre un trabajo previo de Forrester ([For71]).*

Se presenta aquí el modelo desarrollado por Meadows sobre la dinámica de varios factores mundiales tales como población, producción, contaminación, etc. El modelo (que no está exento de controversias) utiliza la metodología de *dinámica de sistemas* para representar las interacciones de las variables seleccionadas.

La primera versión del modelo fue desarrollada por Forrester en el marco de la reunión del Club de Roma [Lan07] y publicada en 1971 (véase [For71]). Posteriormente, Meadows desarrolló una versión más completa del modelo en 1974



Figura 13.1 *Model of the World and Gdańsk society* de Anton Möller  
(1563-1611)<sup>1</sup>

<sup>1</sup>Tomada de [link:21]; obra de dominio público.

(véase [MIM<sup>+</sup>74]) que ha tenido subsecuentes actualizaciones. En el año 2008, Cellier publica una librería Modelica en la que implementa la versión 2002 del modelo (véase [Cel08]). La implementación que se presenta aquí es una aplicación directa de los escenarios que se encuentran en esa librería.

## 13.1 EL MODELO

### 13.1.1 Dinámica de sistemas

Con el nombre de dinámica de sistemas se conoce una propuesta de modelamiento desarrollada por Forrester en los años cincuentas y sesentas. El nombre de por sí es controversial, debido a que se refiere solo a un tipo reducido de sistemas dinámicos y no a la generalidad que sugiere la denominación.

La técnica de dinámica de sistemas busca el desarrollo de modelos matemáticos de sistemas a partir de un conocimiento muy limitado de su comportamiento, fundamentalmente a partir de las relaciones causa-efecto (véase [Lan07]). Este es otro de los aspectos controversiales de la propuesta. ¿Puede desarrollarse un modelo dinámico veraz sin datos numéricos exhaustivos? Para hacerlo, se utiliza una estructura versátil, pero limitada, de relaciones entre variables, que incluye principalmente acumuladores, lazos de realimentación, variables auxiliares, sumas, productos y funciones rectas a trazos.

En términos generales, el comportamiento de una variable dinámica  $y$  se modela como

$$\frac{dy}{dt} = y^+ - y^-$$

en donde  $y^+$  y  $y^-$  son dos ratas de cambio de la variable  $y$ , la primera es positiva y la segunda es negativa. Estas ratas dependen de uno o mas factores  $F_i$  asociados a las otras variables del modelo. Por ejemplo, para  $y^+$ :

$$y^+ = f(F_1, F_2, \dots, F_k)$$

La limitación en la estructura de los modelos aparece en la relación que asume la estructura de las funciones:

$$f(F_1, F_2, \dots, F_k) = f_1(F_1)f_2(F_2)\cdots f_k(F_k)$$

La principal virtud de la dinámica de sistemas es, a la vez, su punto más débil: brinda la posibilidad de construir un modelo matemático dinámico a partir de muy poco conocimiento; sin embargo, debido a la escasez de datos numéricos utilizados, es difícil (o imposible) realizar una validación cuantitativa, lo que le resta credibilidad al modelo resultante.

### 13.1.1.1 El proceso de modelamiento

La técnica de modelamiento de dinámica de sistemas (DS) propone los siguientes pasos (véase [Cel91]):

- DS 1. Preparar la *lista de lavandería*: en este paso se enumera la colección de variables y factores que deben aparecer en el modelo. También se intenta determinar qué factores y variables influencian a cada una de las variables.
- DS 2. Dibujar el *diagrama de influencias*: el listado anterior se lleva a un diagrama (semejante a los diagramas de bloques o a los diagramas de flujo de señal) que ilustra las influencias que recibe cada variable. Es un grafo orientado (un diagrama con flechas) que además marca el sentido (positivo o negativo) de cada influencia.
- DS 3. Dibujar el *diagrama de estructuras*: el diagrama de influencias se vuelve a dibujar, utilizando una convención de símbolos que permite distinguir los tipos de variables involucradas y los tipos de relaciones entre ellas. Algunos de los símbolos empleados en un diagrama de estructuras se muestran en la figura 13.2.
- DS 4. Simular el comportamiento: el diagrama de estructuras permite identificar las relaciones matemáticas entre las variables y, por tanto, simular el comportamiento dinámico. Generalmente esto se hace en una misma herramienta de *software*<sup>2</sup>.

### 13.1.1.2 Tipos de variables y relaciones

Un modelo de dinámica de sistemas distingue los siguientes tipos de variables (véase [Ara95]):

- De nivel: son variables que representan acumulaciones de algo (energía, dinero, masa, etc).
- De flujo: son variables que representan variaciones de las variables de nivel.
- Auxiliares: son las demás variables en el modelo.

---

<sup>2</sup>La herramienta más popular es STELLA (véase [sys07]). En [Cel08] se presenta la implementación en Modelica de la librería, que es la que aquí se ha usado.

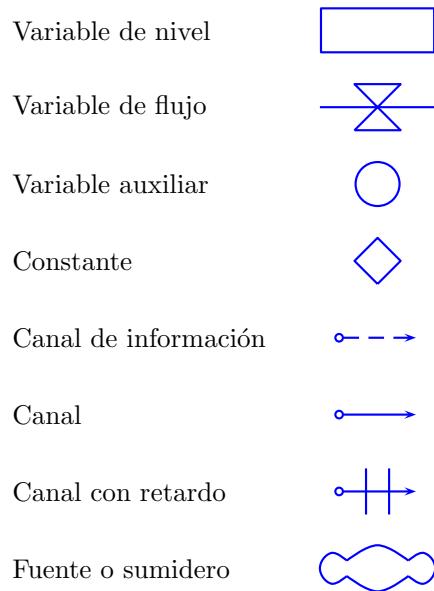


Figura 13.2 Símbolos empleados en un diagrama de estructuras, según [Ara95]

Las relaciones matemáticas que se establecen entre las diferentes variables del modelo pueden ser

- Derivadas: en esencia, una variable de flujo es la derivada de una variable de nivel.
- Producto de varias variables: las variables de flujo pueden obtenerse como producto de otras variables.
- Ganancias: una variable puede obtenerse como el producto de otra variable por un valor constante denominado ganancia.
- Funciones a trozos representadas por tablas: una variable puede obtenerse transformando otra a través de una función continua a trozos. La forma de especificar esa función es mediante una tabla que contiene los puntos extremos de cada trozo, y realizando una interpolación lineal entre dichos puntos.
- Retardos: puede considerarse que la influencia de una variable en otra incluye retardos en el tiempo.

### 13.1.2 Modelo del mundo

#### 13.1.2.1 Consideraciones generales

El modelo propuesto por Meadows en [MIM<sup>+</sup>74] es un modelo global. Se basa en comportamientos promedio o totales; es decir, el modelo no considera las diferencias e interacciones entre diferentes regiones ni grupos culturales.

Los parámetros del modelo se basan en una amplia lista de estudios desarrollados por la comunidad académica internacional. Sobre varios de los parámetros más importantes hay, no obstante, un gran margen de incertidumbre, debido fundamentalmente a la dificultad de medir o estimar parámetros globales.

La generalidad del modelo sumada a la incertidumbre en sus parámetros hace que las predicciones numéricas del modelo sean de dudosa exactitud. Los autores son conscientes de este hecho, y por ello insisten en que las simulaciones no pueden considerarse como predicciones numéricas; en su lugar, proponen una interpretación cualitativa que permite darle una explicación a los comportamientos globales emergentes.

El modelo actualizado a 2002 se presenta y analiza en [MRM05]. En [Kim16] se encuentra una herramienta en línea que permite correr las simulaciones del modelo y sus diferentes escenarios.

El modelo ha sido objeto de una gran controversia, sin duda alimentada por la popularidad alcanzada por los libros que han divulgado las conclusiones de los autores. En [dLM12, Nor73, PM12, Rod10] se encuentran algunas críticas, a favor y en contra, del modelo.

#### 13.1.2.2 Estructura del modelo

El modelo consta de varios submodelos fuertemente interconectados, que pueden agruparse de la siguiente manera:

a. Submodelos de la población: la figura 13.3 muestra el modelo general de la dinámica de la población. Pueden verse dos lazos de realimentación, uno positivo y otro negativo, asociados a los nacimientos y a las muertes. El modelo implementado es más complejo, ya que distingue cuatro franjas de edades:

- Menores de 14 años.
- Entre 14 y 44 años.
- Entre 45 y 64 años.
- Mayores de 64 años.

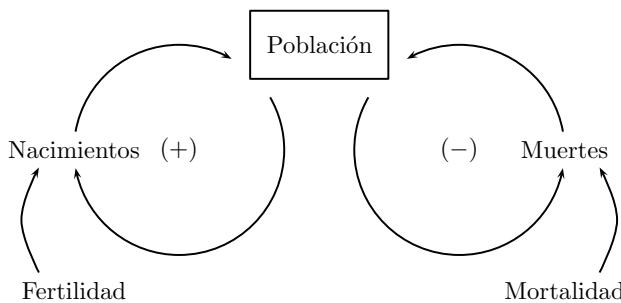


Figura 13.3 Dinámica general de la población

El modelo se apoya además en la noción de *transición demográfica*, que explica las variaciones en la fertilidad y la mortalidad de acuerdo con los siguientes postulados:

- En sociedades preindustriales, la fertilidad y mortalidad son altas y sus tasas tienen valores cercanos, razón por la cual la población crece lentamente.
- Al mejorar las condiciones de salud y nutrición, disminuye la mortalidad, mientras que la fertilidad tiene un rezago de dos generaciones. La brecha entre natalidad y mortalidad acarrea un rápido aumento de la población.
- Con niveles altos de industrialización la fertilidad cae y la población crece lentamente.

La fertilidad está además afectada por el control de la natalidad. Uno de los parámetros de control de la natalidad es el número máximo de hijos deseados por pareja.

- b. Submodelos de la producción industrial: estos submodelos se construyen bajo el concepto de economía física, en contraposición al concepto de economía monetaria. El primero se ocupa de la cuantificación de las cosas reales, mientras que el segundo se ocupa de la valoración del dinero, entendido este como una invención social.

En el modelo se define el *capital industrial* como el conjunto de elementos de *hardware* (máquinas y fábricas) empleados para la fabricación de productos manufacturados. Por otra parte, la *producción industrial* se clasifica en diferentes tipos:

- I. Capital de servicio: fabricación de equipo y edificios para escuelas, bancos, hospitales y tiendas de distribución. El capital de servicio genera intangibles como salud y educación.
- II. Capital agrícola: fabricación de maquinaria, sistemas de riego, etc. para la obtención de productos agrícolas tales como alimentos y fibras.
- III. Capital para obtención de recursos: fabricación de taladros, equipos de minería, poliductos, refinerías, etc. para la obtención de materia prima y energía que requieren las otras formas de capital.
- IV. Bienes de consumo: fabricación de vestidos, carros, electrodomésticos cuyo valor per cápita incide directamente en la medición del bienestar.
- V. Capital industrial: como inversión del mismo capital industrial en forma de producción de maquinaria, generadores eléctricos, molinos, etc.

La figura 13.4 muestra la dinámica general del capital industrial. Hay dos lazos de realimentación, uno positivo y otro negativo, que corresponden a las inversiones y a la depreciación, respectivamente. La producción industrial depende del capital industrial, que se descompone en los cinco ítem arriba señalados.

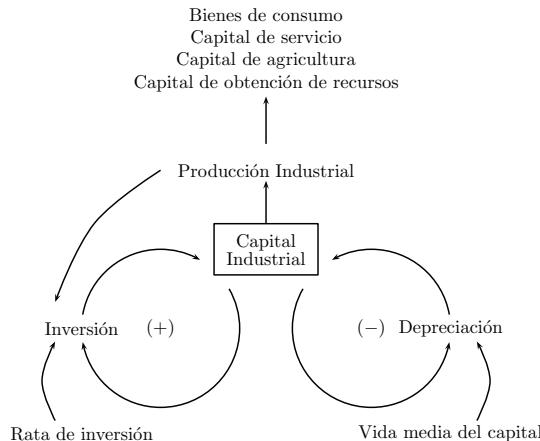


Figura 13.4 Dinámica general del capital

El valor monetario anual de los productos físicos de bienes y servicios finales es el Gross Domestic Product (GDP) o Producto Interno Bruto (PIB).

El GDPI (GDP Index o Índice del PBI) es un indicador asociado al GDP. El GDPI crece con el GDP per cápita, pero para valores a un cierto umbral crece muy lentamente. El umbral utilizado es el GDP per cápita de 1999 para los países de Europa Occidental. Se calcula como:

$$GDPI = \frac{\log_{10} GDPpc - \log_{10} 24,0}{\log_{10} 9508 - \log_{10} 24,0}$$

- c. Submodelos de la contaminación: el modelo incluye una estimación de los contaminantes depositados en el ecosistema, así como una medida del impacto ambiental. La acumulación de contaminantes es proporcional a la utilización de recursos y a la tierra cultivable utilizada para agricultura. Una fracción de los contaminantes acumulados se asimila de nuevo por el ecosistema global.

El modelo incluye también el efecto que pueden tener las nuevas tecnologías que se usen para disminuir la producción de contaminantes. El modelo se basa en la inversión que se haga en estas tecnologías. No se incluye ningún modelo físico sobre el origen de los contaminantes ni su control. Tampoco hay un modelo sobre el efecto invernadero ni el calentamiento global.

En cuanto a la medida del impacto ambiental, esta se estima utilizando el concepto de *huella ecológica* propuesto por Mathis Wackernagel en [WOB<sup>+</sup>99]. La huella ecológica se define como el área de tierra necesaria para proveer la actual forma de vida. Se mide en hectáreas (promedio). El concepto incluye áreas de cultivo, pastos, bosques, zonas de pesca y tierras de construcción necesarias para mantener la población actual de una determinada región con un estilo de vida dado. Se le suma el área de bosque requerida para absorber el dióxido de carbono emitido por la energía fósil usada.

En el modelo se define la Human Ecological Footprint (HEF) o Huella Ecológica Humana como un indicador del impacto ambiental semejante a la definición de Wackernagel. La HEF está normalizada al valor de 1970. Para su cálculo se suman tres áreas (medidas en  $10^9$  hectáreas):

- i. Tierra cultivable para la producción agrícola de alimentos.
- ii. Tierra urbana utilizada para la infraestructura urbana, industrial y de transporte.

- III. Tierra requerida para neutralizar la emisión de contaminantes. Se calcula de forma proporcional a la generación de contaminantes persistentes.
- d. Submodelos de bienestar: el modelo de bienestar humano emplea el Human Development Index (HDI) o Índice de Desarrollo Humano, que es publicado anualmente por el Programa de las Naciones Unidas para el Desarrollo (PNUD).

En el modelo se define el Human Welfare Index (HWI) o Índice de Bienestar Humano como un indicador semejante al HDI. El HWI es el promedio de tres índices:

- i. Life Expectancy Index (LEI) o Índice de Expectativa de Vida, calculado de forma proporcional a la expectativa de vida.
- ii. Education Index (EI) o Índice de Educación, proporcional a la alfabetización y al nivel de escolaridad, que además dependen del nivel de industrialización.
- iii. Bruto Gross Domestic Product Index (GDPI) o Índice del Producto Interno, ya definido antes.
- e. Submodelos de producción de alimentos: la producción de alimentos se modela en función de los siguientes aspectos:

- i. La tierra cultivable disponible.
- ii. El rendimiento de la tierra cultivable.
- iii. El desperdicio y las pérdidas del alimento producido.
- iv. Los requerimientos de alimento por persona.

El modelo de la producción de alimentos es quizás donde más fácilmente se evidencia la naturaleza global del modelo. No se modelan los mecanismos de distribución de alimentos ni se estiman las diferencias regionales de producción y acceso.

- f. Submodelos de recursos No Renovables: el modelo busca estimar la variación de las reservas de materiales y fuentes de energía no renovables. El modelo utiliza como parámetro la cantidad total de recursos naturales en todo el planeta en el año 1900. Este parámetro, no obstante, es imposible de conocer con mediana certeza. De hecho, es uno de los aspectos más criticados de este modelo y de los anteriores de los mismos autores.

La naturaleza global del modelo hace que no se puedan distinguir las variaciones entre los diferentes recursos naturales. El petróleo es, sin duda, el recurso que mayor preocupación ha causado en los autores del modelo.

La inversión en tecnología se incorpora al modelo en dos formas: como la posibilidad de extraer más eficientemente los recursos, y como la posibilidad de utilizarlos más eficientemente.

### 13.1.2.3 Escenarios de análisis

En [MRM05] se formulan diferentes posibles escenarios “futuros” (visto desde el 2002). Estos escenarios son los siguientes:

- Escenario 1: es el escenario de referencia. Corresponde a la simulación bajo las condiciones “actuales” de consumo, uso de recursos, natalidad, etc. De acuerdo con los resultados de la simulación, “la población y la producción crecen hasta que el crecimiento se detiene por la inaccesibilidad creciente de los recursos no renovables... (al caer la producción) los alimentos, y servicios de salud se reducen disminuyendo la expectativa de vida e incrementando las tasas promedio de muerte” ([MRM05], pág. 168).
- Escenario 2: es un escenario con recursos no renovables más abundantes. En este escenario se duplica la estimación de recursos no renovables iniciales (en 1900) y se suponen adelantos tecnológicos que posponen el incremento en los costos de extracción. Como resultado la producción industrial y la población continúan creciendo veinte años más que en el escenario 1, “pero los niveles de contaminación se disparan, deteriorando el rendimiento de la tierra y exigiendo enormes inversiones en recuperación de la agricultura. La población finalmente desciende a causa de la escasez de alimentos y los efectos negativos en la salud derivados de la contaminación” ([MRM05], pág. 172).
- Escenarios 3, 4, 5 y 6: suponen adelantos tecnológicos que cambian el panorama. Se construyen a partir del escenario 2. Los adelantos se producen debido a la decisión de invertir más recursos en tecnología. Se estima un tiempo de implementación de veinte años para las nuevas tecnologías. En concreto, los adelantos tecnológicos permiten: mejorar el control de la contaminación (escenarios 3 y siguientes), incrementar el rendimiento agrícola (escenarios 4 y siguientes); incrementar la protección contra la erosión (escenarios 5 y siguientes), utilizar más eficientemente los recursos (escenario 6). Como resultado, los adelantos

“permiten un mundo simulado bastante amplio y prospero, hasta que la felicidad empieza a declinar en respuesta al costo acumulado de las tecnologías” ([MRM05], pág. 218).

- Escenario 7: supone un esfuerzo por realizar un control demográfico. Se construye a partir del escenario 2. Se supone que todas las parejas deciden limitar el tamaño de sus familias a dos hijos, y logran hacerlo de forma completamente efectiva. El crecimiento de la población tarda una generación en verse disminuido, y facilita el crecimiento industrial. No obstante, el crecimiento “se detiene por el costo derivado de la contaminación, tal como en el Escenario 2” ([MRM05], pág. 240).
- Escenario 8: supone además del control demográfico una cambio en la dinámica del consumo. La decisión es la de mantener un consumo industrial per cápita estable, cercano a un 110 % del promedio mundial en el año 2000. El modelo supone una repartición homogénea del producto industrial<sup>3</sup>, lo que implicaría una mejora notabilísima entre los pobres del mundo, y un cambio en los patrones de consumo de los ricos. Como resultado se logra prolongar el estado favorable del Escenario 8, “pero la contaminación afecta crecientemente los recursos agrícolas. La producción per cápita de alimentos decae, disminuyendo la expectativa de vida y la población” ([MRM05], pág. 242).
- Escenario 9: es una combinación de los escenarios 3 al 8. En él se supone la utilización de los adelantos tecnológicos, así como la incorporación de control demográfico y consumo responsable. El resultado es una sociedad “sostenible: cerca de 8 mil millones de personas viven con un alto bienestar humano y generando una huella ecológica que disminuye continuamente” ([MRM05], pág. 244).
- Escenarios 10 y 11: se simula la incorporación de las acciones correctivas del escenario 9 en otros momentos. El escenario 10 explora la incorporación veinte años antes (en 1982), y el escenario 11 lo hace diez años después (en 2012). En el primer caso el resultado es una sociedad sostenible con “menor población, menos contaminación, más recursos No Renovables y un bienestar humano ligeramente superior para todos”<sup>4</sup> ([MRM05], pág. 248).

---

<sup>3</sup>En la documentación no hay claridad sobre la manera de incorporar esta suposición de repartición homogénea en el modelo, dada la naturaleza global de este.

<sup>4</sup>El escenario 11 no es explorado por Meadows, pero sí es implementado por Cellier en [Cel08].

## 13.2 PLANTAS DE EXPERIMENTACIÓN Y EXPERIMENTOS SUGERIDOS

### Planta de experimentación 13: World3.

**Presentación:** el modelo corresponde a la implementación de Cellier del modelo de la dinámica del mundo propuesto por Meadows, sobre un trabajo previo de Forrester.

**Instrumentación:** el modelo cuenta con 19 parámetros ajustables organizados en 6 grupos de controles (véase tabla 13.1). Como resultado del experimento, el programa despliega:

- 8 curvas organizadas en 6 gráficos (véase tabla 13.2).
- Una tabla de datos del comportamiento de 9 variables (véase tabla 13.3).

**Experimentos sugeridos:** el siguiente es el listado de experimentos sugeridos:

#### *Experimento 13.1: Recursos disponibles.*

¿Qué efectos tiene subestimar o sobreestimar la cantidad total de recurso naturales disponibles? El parámetro Recursos Iniciales, del grupo Recursos No Renovables, corresponde a la estimación (en TeraToneladas) de la cantidad de recursos naturales disponibles en 1900. Explore el efecto de este parámetro.

#### *Experimento 13.2: Eficiencia en la obtención de recursos no renovables.*

¿Qué impacto tiene la inversión en tecnología para obtener más eficientemente los recursos no renovables? Explore el impacto de incorporar una política de inversión de recursos para aumentar la eficiencia del proceso de obtención de recursos no renovables.

***Experimento 13.3: Control de la contaminación.***

¿Qué efecto tiene limitar la contaminación? Los parámetros 1 y 2 del grupo Control de Contaminación controlan el esfuerzo de una política de inversión en nuevas tecnologías para controlar la contaminación. Explore el efecto de estos parámetros.

***Experimento 13.4: Obsolescencia tecnológica.***

¿Cómo impacta el hecho de tener productos más o menos duraderos? Compare los impactos de variaciones en la vida media de cada tipo de producto, cuando se implementa una política de equilibrio del consumo.

***Experimento 13.5: Escenarios de Meadows.***

¿Qué conclusiones pueden extraerse de los diferentes escenarios propuestos por Meadows? Meadows propuso varios escenarios (modelados por Cellier). Simule los diferentes escenarios, analice el comportamiento del modelo y compare sus conclusiones con las de Meadows.

***Experimento 13.6: Sensibilidad y robustez.***

¿Qué tan sensible y robusto es el modelo? Analice la robustez y sensibilidad del modelo con combinaciones de modificaciones de los parámetros.

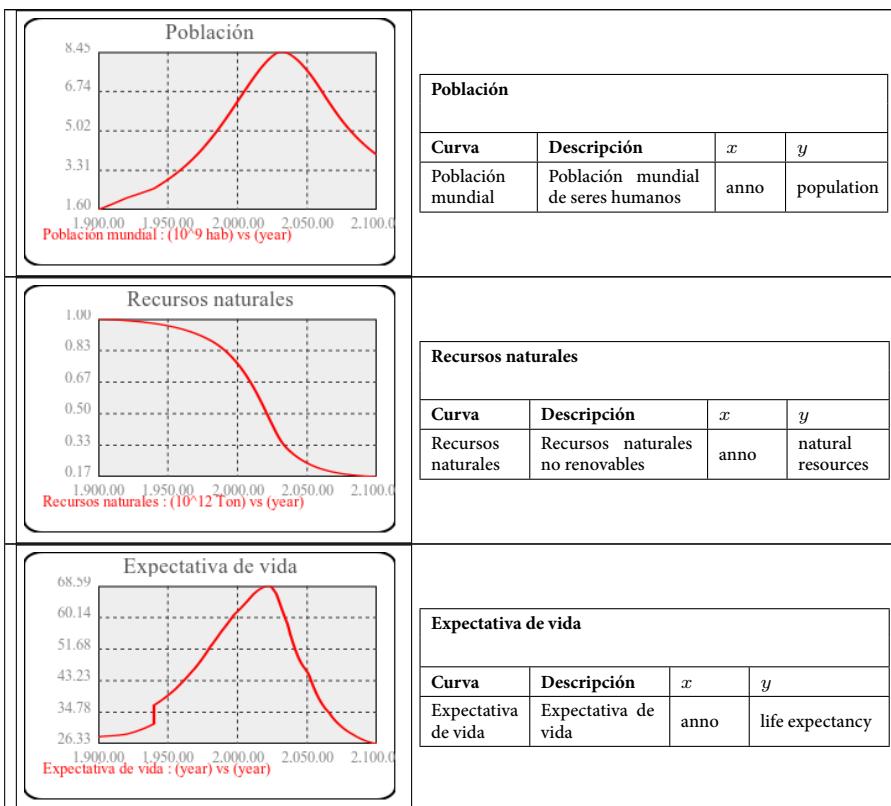
<sup>4</sup>La información en las tablas siguientes se muestra tal como aparece en la base de datos. Por esta razón hay datos tanto en español como en inglés.

Tabla 13.1 Parámetros del experimento 13, “World3”

<b>Título:</b>	World3		
<b>Descripción:</b>	El modelo corresponde a la implementación de Cellier del modelo de la dinámica del mundo propuesto por Meadows, sobre un trabajo previo de Forrester.		
	<b>Implementación</b>		
<b>Créditos</b>	<b>e-mail</b>		
<b>Parámetros</b>			
Grupo	Nombre Modelica	Nombre	Descripción
Recursos No Renovables	<i>NaturalResources</i>	Recursos iniciales	Estimación de los recursos no renovables disponibles en 1900, medidos en $10^{12}$ Ton.
Control de Contaminación	<i>poll1</i>	Parámetro 1	Multiplicador para cambio de tecnología de control de la contaminación cuando los Contaminantes son muy altos.
	<i>poll2</i>	Parámetro 2	Multiplicador para cambio de tecnología de control de la contaminación cuando los Contaminantes son bajos.
Producción agrícola eficiente	<i>yield1</i>	Parámetro 1	Multiplicador para cambio de tecnología agrícola cuando hay pocos alimentos faltantes.
	<i>yield2</i>	Parámetro 2	Multiplicador para cambio de tecnología agrícola cuando hay muchos alimentos faltantes.
Uso eficiente de recursos	<i>tech1</i>	Parámetro 1	Multiplicador para cambio de tecnología de uso de recursos no renovables cuando hay gran escasez de recursos.
	<i>tech2</i>	Parámetro 2	Multiplicador para cambio de tecnología de uso de recursos no renovables cuando hay poca escasez de recursos.
Equilibrio de consumo	<i>ind_out_pc_des</i>	Producción deseada	Meta de producción industrial en equilibrio.
	<i>p_avg_life_agr_inp_2</i>	Vida útil media de los bienes agrícolas	Vida útil media de los bienes agrícolas producidos.
	<i>p_avg_life_ind_cap_2</i>	Vida útil media de los bienes industriales	Vida útil media de los bienes industriales producidos.
	<i>p_avg_life_serv_cap_2</i>	Vida útil media de los servicios	Vida útil media de los servicios producidos.
Acciones correctivas	<i>caorFlag</i>	Obtención eficiente de recursos NR	Inversión en aumento de la eficiencia para la obtención de recursos faltantes.
	<i>pollFlag</i>	Control de contaminación	Inversión en tecnología para control de la contaminación.
	<i>landFlag</i>	Control de erosión	Inversión en tecnología para control de erosión.
	<i>fertFlag</i>	Control demográfico	Control demográfico.

<b>Acciones correctivas</b>	<i>equilFlag</i>	Equilibrio de consumo	Mantener las proporción de consumo en relación a la producción industrial.
	<i>Tyes</i>	Año Inicial	Año de inicio de las acciones correctivas.
	<i>techFlag</i>	Uso eficiente de recursos	Inversión en aumento de la eficiencia para la utilización de recursos no renovables.
	<i>yieldFlag</i>	Producción agrícola eficiente	Inversión en aumento de la eficiencia en la producción de alimentos.

Tabla 13.2 Figuras del experimento 13, "World3"



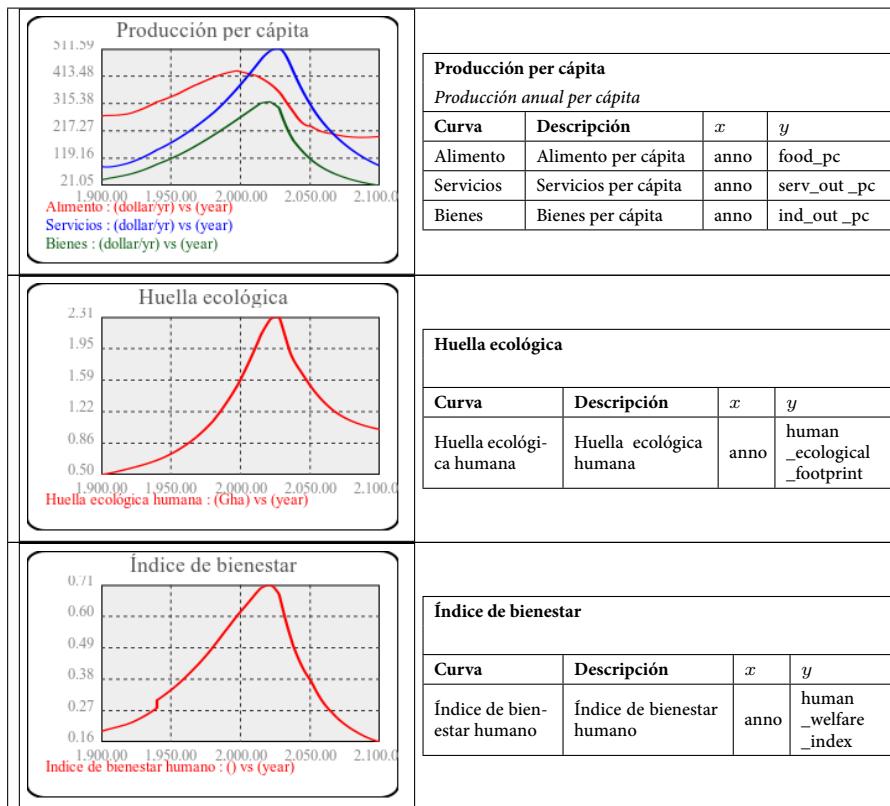


Tabla 13.3 Variables en la tabla de resultados del experimento 13, "World3"

Variable	Descripción	Unidades
anno	Año	year
population	Población humana mundial	$10^9$ hab
naturalresources	Recursos naturales no renovables	$10^{12}$ Ton
lifeexpectancy	Expectativa de vida	year
food_pc	Alimento anual per cápita	dollar/yr
serv_out_pc	Services anuales per cápita	dollar/yr
ind_out_pc	Bienes de consumo per cápita	dollar/yr
human_ecological_footprint	Huella ecológica humana	Gha
human_welfare_index	Índice de bienestar humano	

### 13.3 LA IMPLEMENTACIÓN

La librería de SystemDynamics desarrollada por Cellier (véase [Cel08]) tiene dentro de sus ejemplos la implementación de once diferentes escenarios del modelo del mundo de Meadows (véase [MIM<sup>+</sup>74]). A partir de la implementación del escenario 1 se ha desarrollado una nueva clase (véase el archivo MyWorld3.mo) con las siguientes adiciones elementales:

- Se han definido once variables de salida con nombres cortos, para facilitar su despliegue en UNVirtualLab. Cada una de estas variables corresponde a alguna de las variables previamente definidas en el modelo de la librería.
- Se han definido veintisiete parámetros que permiten obtener los once escenarios del modelo y combinaciones de ellos. Estos parámetros son fácilmente incorporables en la estructura de UNVirtualLab.
- Se ha fijado el tiempo de simulación desde el año 1900 al año 2100.

La implementación de Cellier utiliza 13 bloques fuertemente interconectados:

1. Dinámica poblacional
2. Fertilidad humana
3. Huella ecológica humana
4. Dinámica de la contaminación
5. Inversión industrial
6. Índice de Bienestar Humano
7. Dinámica de la tierra cultivable
8. Uso de la fuerza laboral
9. Expectativa de vida
10. Producción de alimentos
11. Fertilidad de la tierra
12. Uso de recursos no renovables
13. Inversión en el sector de servicios

La planta experimental que se ha diseñado permite simular los diez escenarios analizados en [MRM05], uno más propuesto por Cellier, así como variaciones y combinaciones de ellos. La tabla 13.4 muestra cómo obtener cada uno de los once escenarios en la planta experimental.

Algunos de los escenarios se obtienen fijando ciertos parámetros internos del modelo<sup>5</sup>. La planta permite ajustar esos parámetros, de tal forma que puede estudiarse la gradualidad del efecto de cada uno de ellos. Además, se han organizado los controles de la planta para destacar el efecto de incorporar políticas globales, en forma de decisiones para incorporar acciones correctivas.

Tabla 13.4 Obtención de los escenarios de Meadows en la planta de experimentación de UNVirtualLab

Nombre	Descripción	Base	Modificaciones	
			Parámetro	Valor
E 2	Abundancia de recursos no renovables	E 1	Obtención eficiente de recursos NR	SI
			Recursos Iniciales	2
E 3	Abundancia de recursos no renovables, tecnología para control de la contaminación	E 2	Control de contaminación	SI
			Control de contaminación. Parámetro 1	0,4
			Control de contaminación. Parámetro 2	0,4
E 4	Abundancia de recursos no renovables, tecnología para control de la contaminación, mejora en el rendimiento agrícola	E 3	Producción agrícola eficiente	SI
			Producción agrícola eficiente. Parámetro 1	0,4
			Producción agrícola eficiente. Parámetro 2	0,4
E 5	Abundancia de recursos no renovables, tecnología para control de la contaminación, mejora en el rendimiento agrícola, control de erosión	E 4	Control de erosión	SI
E 6	Abundancia de recurso no renovables, tecnología para control de la contaminación, mejora en el rendimiento agrícola, control de erosión, tecnología de uso eficiente de recursos	E 5	Uso eficiente de recursos	SI
			Uso eficiente de recursos. Parámetro 1	0,4
			Uso eficiente de recursos. Parámetro 2	0,4

<sup>5</sup>El significado de los parámetros ajustables del modelo que llevan la simulación de un escenario a otro no está bien documentado por los autores. En general se trata de valores que residen en tablas de datos usadas mediante interpolación para estimar factores multiplicadores o tasas de cambio.

Nombre	Descripción	Base	Modificaciones	
			Parámetro	Valor
E7	El mundo busca una población estable desde el 2002	E2	Control demográfico	SI
E 8	El mundo busca una población estable y una producción industrial per cápita estable desde el 2002	E 7	Equilibrio de consumo	SI
			Producción deseada	350
			Vida de bienes agrícolas	2,5
			Vida de bienes industriales	18
			Vida de bienes servicios	25
E 9	El mundo busca una población estable y una producción industrial per cápita estable y utiliza tecnologías para control de contaminación, mejora en el rendimiento agrícola, control de erosión, tecnología de uso eficiente de recursos, desde el 2002	E 8	Control de contaminación	SI
			Control de contaminación. Parámetro 1	0,4
			Control de contaminación. Parámetro 2	0,4
			Producción agrícola eficiente	SI
			Producción agrícola eficiente. Parámetro 1	0,4
			Producción agrícola eficiente. Parámetro 2	0,4
			Control de erosión	SI
			Uso eficiente de recursos	SI
			Uso eficiente de recursos. Parámetro 1	0,4
			Uso eficiente de recursos. Parámetro 2	0,4
E 10	Las políticas del escenario 9 fueron implementadas veinte años atrás	E 9	Año inicial	1982
E 11	Las políticas del escenario 9 se implementan diez años después	E 9	Año inicial	2012

### 13.3.1 Listado de archivos

La tabla 13.5 muestra el listado de los archivos fuente de la implementación del modelo.

Tabla 13.5 Archivos del modelo

Número	Archivo
13.1	MyWorld3.mo

## Archivo 13.1 MyWorld3.mo

```

model MyWorld3
  SystemDynamics.WorldDynamics.World3.Scenario_1
    World(nr_resources_init=NaturalResources*1e12,
          p_fr_cap_al_obt_res_2 = {1,caor1,caor2
            ,0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05},
          t_fcaor_time=t_fcaor_time,
          p_ppoll_tech_chg_mlt = {-poll1,-poll2,0,0},
          t_policy_year = t_policy_year,
          p_yield_tech_chg_mlt = {0,0,yield1,yield2},
          t_land_life_time = t_land_life_time,
          p_res_tech_chg_mlt = {-tech1,-tech2,0,0},
          t_fert_cont_eff_time = t_fert_cont_eff_time,
          t_zero_pop_grow_time = t_zero_pop_grow_time,
          ind_out_pc_des = ind_out_pc_des,
          p_avg_life_agr_inp_2 = p_avg_life_agr_inp_2,
          p_avg_life_ind_cap_2 = p_avg_life_ind_cap_2,
          p_avg_life_serv_cap_2 = p_avg_life_serv_cap_2,
          t_ind_equil_time = t_ind_equil_time);
  output Real population(unit="10^9 hab") "Población humana mundial";
  output Real naturalresources(unit="10^12 Ton") "Recursos naturales no
    renovables";
  output Real food(unit="10 ^ 12 Ton") "Producción anual de alimentos";
  output Real lifeexpectancy(unit="year") "Expectativa de vida";
  output Real industrialoutput(unit="10^12 dollar/yr") "Producción
    industrial";
  output Real food_pc(unit = "dollar/yr") "Alimento anual per cápita";
  output Real serv_out_pc(unit = "dollar/yr") "Services anuales per cá
    pita";
  output Real ind_out_pc(unit = "dollar/yr") "Bienes de consumo per cá
    pita";
  output Real human_ecological_footprint(unit = "Gha") "Huella ecoló
    gica humana";
  output Real human_welfare_index "Índice de bienestar humano";
  output Real anno(unit="year") "Año";
  parameter Real Tyes(unit="year")=2002 "Año de incorporación de
    cambios";
  parameter Real Tno(unit="year")=4000 "Un año fuera de la simulación";

// Scenario 2
parameter Real NaturalResources(unit="10^12 Ton")=1 "Recursos
  naturales no renovables iniciales (en 1900); // 2
parameter Integer caorFlag=1 "Inversión en aumento de la eficiencia
  para la producción de recursos"; // 0:yes 1: no;
parameter Real caor1 = 0.1 + caorFlag*(0.2-0.1); // parámetro 1 de
  extracción eficiente de recursos
parameter Real caor2 = 0.05 + caorFlag*(0.1-0.05); // parámetro 2 de
  extracción eficiente de recursos
parameter Real t_fcaor_time=Tyes + caorFlag*(Tno-Tyes);
// Scenario 3
parameter Integer pollFlag=1 "Inversión en tecnología para control de
  contaminación"; // 0:yes 1: no
parameter Real poll1=0 "Factor 1"; // 0.04;
parameter Real poll2=0 "Factor 2"; // 0.04;

```

```

parameter Real t_policy_year=Tyes + pollFlag*(Tno-Tyes);
// Scenario 4 "Inversión en tecnología para mejoramiento del
rendimiento agrícola"
parameter Integer yieldFlag=1 "Inversión en tecnología para
mejoramiento del rendimiento agrícola"; // 0: yes 1: no
parameter Real yield1=0.04 + yieldFlag*(0-0.04); // parámetro 1 de
yield
parameter Real yield2=0.04 + yieldFlag*(0-0.04); // parámetro 2 de
yield
// Scenario 5
parameter Integer landFlag=1 "Inversión en tecnología para control de
erosión"; // 0: yes 1: no
parameter Real t_land_life_time=Tyes + landFlag*(Tno-Tyes);
// Scenario 6 "Inversión en tecnología para uso más eficiente de los
recursos"
parameter Integer techFlag=1 "Inversión en tecnología para uso
eficiente de recursos NR";
parameter Real tech1 = 0.04 + techFlag*(0 - 0.04); // parámetro 1 de
tecnología
parameter Real tech2 = 0.04 + techFlag*(0 - 0.04); // parámetro 2 de
tecnología
// Scenario 7
parameter Integer fertFlag=1 "Control demográfico"; // 0: yes 1: no
parameter Real t_fert_cont_eff_time=Tyes + fertFlag*(Tno-Tyes);
parameter Integer growFlag=fertFlag "Estado de equilibrio poblacional
";
// 0: yes 1: no
parameter Real t_zero_pop_grow_time=Tyes + growFlag*(Tno-Tyes);
// Scenario 8
parameter Real ind_out_pc_des(unit = "dollar/yr") = 400 "producción
industrial per cápita deseada"; // 350;
parameter Real p_avg_life_agr_inp_2(unit = "yr") = 2 "vida promedio
de los bienes agrícolas producidos"; // 2.5;
parameter Real p_avg_life_ind_cap_2(unit = "yr") = 14 "vida promedio
de los bienes industriales producidos"; // 18;
parameter Real p_avg_life_serv_cap_2(unit = "yr") = 20 "vida útil de
los servicios producidos"; // 25;
parameter Integer equilFlag=1 "Estado de equilibrio industrial"; // 0:
yes 1: no
parameter Real t_ind_equil_time = Tyes + equilFlag*(Tno-Tyes);

equation
    anno                      = time;
    population                = World.population/1e9;
    naturalresources          = World.nr_resources/1e12;
    food                       = World.food/1e12;
    lifeexpectancy            = World.life_expectancy;
    industrialoutput          = World.industrial_output/1e12;
    food_pc                    = World.food_pc;
    serv_out_pc                = World.serv_out_pc;
    ind_out_pc                 = World.ind_out_pc;
    human_ecological_footprint = World.human_ecological_footprint;
    human_welfare_index        = World.human_welfare_index;

```

```
annotation(experiment(StartTime = 1900, StopTime = 2100),  
           experimentSetupOutput);  
end MyWorld3;
```

# 14

## IDENTIFICACIÓN ALGEBRAICA. MASA DESLIZANTE CON FRICCIÓN

*El modelo ilustra la utilización de la técnica de identificación de parámetros por métodos algebraicos, mediante su aplicación al caso elemental de una masa deslizante con fricción.*

El problema de identificación de parámetros consiste en la estimación de los valores numéricos de los parámetros de modelos matemáticos de sistemas, basados en la información proveniente del comportamiento histórico de dichos sistemas. El problema usualmente se aplica al caso de sistemas dinámicos.

La identificación algebraica es una técnica recientemente propuesta por Sira y otros (véase [SRGRCCRLJ14]). Utiliza manipulaciones algebraicas de los modelos matemáticos para deducir expresiones que permiten realizar estimaciones en línea de los parámetros del modelo.

El primer ejemplo que se analiza en [SRGRCCRLJ14] es el de una masa deslizante sujetada a una fuerza. El problema de identificación consiste en estimar la masa a partir de mediciones de la fuerza y el desplazamiento. En esta implementación se simula esa planta y el mecanismo de identificación.



Figura 14.1 Masa deslizante

Se han adicionado: 1) ruido en las mediciones, y 2) una fricción de Coulomb entre la masa y la superficie.

## 14.1 EL MODELO

El desplazamiento  $x(t)$  de una masa  $m$  deslizante sobre la que se ejerce una fuerza  $u(t)$  sin rozamiento, está regido por las leyes de Newton:

$$m\ddot{x}(t) = f(t) \quad (14.1)$$

Para estimar el parámetro  $m$  usando el enfoque de la identificación algebraica es necesario obtener una estimación  $m_e(t)$  que evolucione en el tiempo, en función de las variables medibles e independiente de las condiciones iniciales. Si se tuviese un registro de la aceleración  $\ddot{x}(t)$  y de la fuerza  $f(t)$ , esta estimación podría ser:

$$m_e(t) = \frac{f(t)}{\ddot{x}(t)} \quad (14.2)$$

Sin embargo, si las variables medibles son el desplazamiento  $x(t)$  y la fuerza  $f(t)$ , es necesario obtener otra relación matemática. Para ello, se multiplica la expresión 14.1 por  $(t - t_0)^2$  a cada lado de la ecuación y se integra dos veces, en donde  $t_0$  es el tiempo inicial:

$$m(t - t_0)^2 \ddot{x}(t) = (t - t_0)^2 f(t) \quad (14.3)$$

### 14.1.1 Primera integración

Para realizar una primera integración utilizamos la variable auxiliar  $\sigma$ :

$$m \int_{t_0}^t (\sigma - t_0)^2 \ddot{x}(\sigma) d\sigma = \int_{t_0}^t (\sigma - t_0)^2 f(\sigma) d\sigma \quad (14.4)$$

La integral de la izquierda puede analizarse utilizando el método de integración por partes:

$$u = (\sigma - t_0)^2 \quad dv = \ddot{x}(\sigma) d\sigma \quad du = 2(\sigma - t_0) d\sigma \quad v = \dot{x}(\sigma) \quad (14.5)$$

$$d(uv) = udv + vdu \quad \int ud(v) = uv - \int vdu \quad (14.6)$$

$$\int_{t_0}^t (\sigma - t_0)^2 \ddot{x}(\sigma) d\sigma = (\sigma - t_0)^2 \dot{x}(\sigma) \Big|_{t_0}^t - 2 \int_{t_0}^t (\sigma - t_0) \dot{x}(\sigma) d\sigma \quad (14.7)$$

$$\int_{t_0}^t (\sigma - t_0)^2 \ddot{x}(\sigma) d\sigma = \left[ (t - t_0)^2 \dot{x}(t) - 2 \int_{t_0}^t (\sigma - t_0) \dot{x}(\sigma) d\sigma \right] \quad (14.8)$$

Y por tanto, al retomar 14.4:

$$m \left[ (t - t_0)^2 \dot{x}(t) - 2 \int_{t_0}^t (\sigma - t_0) \dot{x}(\sigma) d\sigma \right] = \int_{t_0}^t (\sigma - t_0)^2 f(\sigma) d\sigma \quad (14.9)$$

Para analizar la integral de la izquierda, podemos de nuevo hacer uso del método de integración por partes, definiendo ahora  $u$  y  $dv$  como

$$u = (\sigma - t_0) \quad dv = \dot{x}(\sigma) d\sigma \quad du = d\sigma \quad v = x(\sigma) \quad (14.10)$$

$$d(uv) = udv + vdu \quad \int ud(v) = uv - \int vdu \quad (14.11)$$

$$\int_{t_0}^t (\sigma - t_0) \dot{x}(\sigma) d\sigma = (\sigma - t_0) x(\sigma)|_{t_0}^t - \int_{t_0}^t x(\sigma) d\sigma \quad (14.12)$$

$$\int_{t_0}^t (\sigma - t_0) \dot{x}(\sigma) d\sigma = (t - t_0) x(t) - \int_{t_0}^t x(\sigma) d\sigma \quad (14.13)$$

Y por tanto 14.9 se transforma en

$$m \left[ (t - t_0)^2 \dot{x}(t) - 2(t - t_0)x(t) + 2 \int_{t_0}^t x(\sigma) d\sigma \right] = \int_{t_0}^t (\sigma - t_0)^2 f(\sigma) d\sigma \quad (14.14)$$

La expresión 14.14 permitiría desarrollar una estimación de  $m$  a partir de las mediciones de  $x(t)$ ,  $\dot{x}(t)$  y  $f(t)$ :

$$m_e(t) = \frac{\int_{t_0}^t (\sigma - t_0)^2 f(\sigma) d\sigma}{(t - t_0)^2 \dot{x}(t) - 2(t - t_0)x(t) + 2 \int_{t_0}^t x(\sigma) d\sigma} \quad (14.15)$$

### 14.1.2 Segunda integración

Para obtener una estimación que no requiera el registro de  $\dot{x}(t)$  se integra nuevamente la expresión 14.14

$$m \int_{t_0}^t \left[ (\lambda - t_0)^2 \dot{x}(\lambda) - 2(\lambda - t_0)x(\lambda) + 2 \int_{t_0}^\lambda x(\sigma) d\sigma \right] d\lambda = \\ \int_{t_0}^t \int_{t_0}^\lambda (\sigma - t_0)^2 f(\sigma) d\sigma d\lambda \quad (14.16)$$

$$m \left[ \int_{t_0}^t (\lambda - t_0)^2 \dot{x}(\lambda) d\lambda - 2 \int_{t_0}^t (\lambda - t_0) x(\lambda) d\lambda + 2 \int_{t_0}^t \int_{t_0}^\lambda x(\sigma) d\sigma d\lambda \right] = \\ \int_{t_0}^t \int_{t_0}^\lambda (\sigma - t_0)^2 f(\sigma) d\sigma d\lambda \quad (14.17)$$

La primera de las integrales de la izquierda se analiza utilizando de nuevo el método de integración por partes:

$$u = (\lambda - t_0)^2 \quad dv = \dot{x}(\lambda) d\lambda \quad du = 2(\lambda - t_0) d\lambda \quad v = x(\lambda) \quad (14.18)$$

$$d(uv) = u dv + v du \quad \int u d(v) = uv - \int v du \quad (14.19)$$

$$\int_{t_0}^t (\sigma - t_0)^2 \dot{x}(\sigma) d\sigma = (\lambda - t_0)^2 x(\lambda) \Big|_{t_0}^t - 2 \int_{t_0}^t (\lambda - t_0) x(\lambda) d\lambda \quad (14.20)$$

$$\int_{t_0}^t (\sigma - t_0)^2 \dot{x}(\sigma) d\sigma = (t - t_0)^2 x(t) - 2 \int_{t_0}^t (\lambda - t_0) x(\lambda) d\lambda \quad (14.21)$$

La ecuación 14.22 se transforma en

$$m \left[ (t - t_0)^2 x(t) - 4 \int_{t_0}^t (\lambda - t_0) x(\lambda) d\lambda + 2 \int_{t_0}^t \int_{t_0}^\lambda x(\sigma) d\sigma d\lambda \right] = \\ \int_{t_0}^t \int_{t_0}^\lambda (\sigma - t_0)^2 f(\sigma) d\sigma d\lambda \quad (14.22)$$

lo que permite obtener una estimación del parámetro  $m$  en términos del registro histórico de  $x(t)$  y  $f(t)$ <sup>1</sup>:

$$m_e(t) = \frac{d(t)}{n(t)} \quad (14.23)$$

$$d(t) = \int_{t_0}^t \int_{t_0}^\lambda (\sigma - t_0)^2 f(\sigma) d\sigma d\lambda \quad (14.24)$$

$$n(t) = (t - t_0)^2 x(t) - 4 \int_{t_0}^t (\lambda - t_0) x(\lambda) d\lambda + 2 \int_{t_0}^t \int_{t_0}^\lambda x(\sigma) d\sigma d\lambda \quad (14.25)$$

---

<sup>1</sup>Seguimos aquí parcialmente la notación usada en [SRGRCRLJ14], que estima  $1/m_e = n(t)/d(t)$ , numerador sobre denominador.

### 14.1.3 Realizaciones

Las realizaciones de 14.24 y 14.25 se obtienen con derivaciones sucesivas:

$$\begin{cases} d(t) = \eta_1(t) \\ \dot{\eta}_1(t) = \eta_2(t) \\ \dot{\eta}_2(t) = (t - t_0)^2 f(t) \end{cases} \quad \begin{cases} n(t) = (t - t_0)^2 x(t) + z_1(t) \\ \dot{z}_1(t) = -4(t - t_0)x(t) + z_2(t) \\ \dot{z}_2(t) = 2x(t) \end{cases} \quad (14.26)$$

Las realizaciones 14.26 pueden visualizarse como se muestra en la figura 14.2.  $\mathcal{D}$  y  $\mathcal{N}$  pueden interpretarse como dos filtros variantes en el tiempo especificados por 14.26. Sobre este esquema, en [SRGRCLJ14] se proponen además tres variaciones (véase figura 14.3):

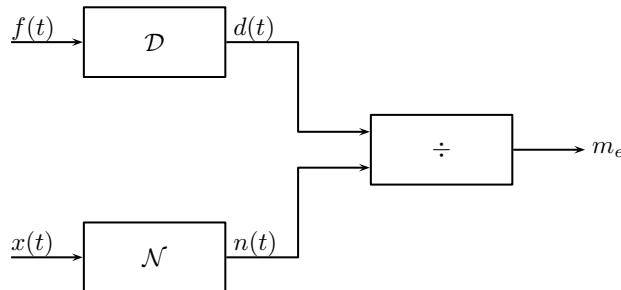


Figura 14.2 Diagrama de bloques del proceso de estimación básico

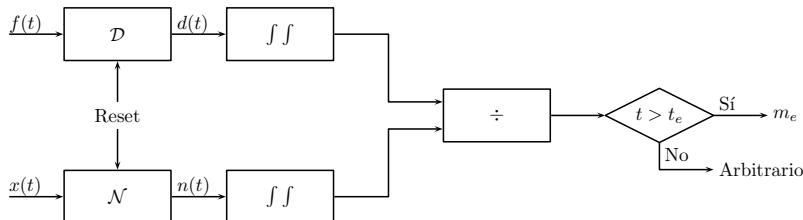


Figura 14.3 Diagrama de bloques del proceso de estimación con filtrado de señal

- Para evitar que la estimación de  $m_e$  en 14.23 sea indeterminada cuando  $n(t) = 0$ , el proceso de identificación se inicia después de un tiempo  $t_e$ . Para tiempos menores a  $t_e$  se utiliza como estimación un valor arbitrario.
- Debido a que los filtros  $\mathcal{D}$  y  $\mathcal{N}$  son inestables, el proceso debe reiniciarse de cuando en cuando.

- c. Para suavizar la estimación, se propone un filtrado de las señales  $d(t)$  y  $n(t)$  consistente en una doble integración.

## 14.2 PLANTAS DE EXPERIMENTACIÓN Y EXPERIMENTOS SUGERIDOS

### **Planta de experimentación 14: Masa con fricción.**

**Presentación:** la planta consiste en un modelo de masa deslizante con fricción de Coulomb al que se le aplica una fuerza que puede ser de dos tipos: constante o sinusoidal (de periodo 1s). La magnitud de la fuerza se puede modificar así como la masa nominal. La fricción de Coulomb se modela como un valor absoluto constante que se puede modificar; el sentido de la fuerza de fricción es opuesto al sentido del desplazamiento. La masa, además, se detiene si el desplazamiento excede un valor máximo que puede ser ajustado por el usuario.

Sobre esta planta se toman las señales de desplazamiento y fuerza. La planta incorpora ruido aditivo sobre las mediciones, cuya amplitud se puede modificar.

El proceso de identificación puede o no incorporar el filtrado de  $n(t)$  y de  $d(t)$ . Para evitar la indeterminación en la expresión 14.23, la división solo se realiza si  $|n(t)| > n_{min}$ , en donde  $n_{min}$  es un valor modificable.

**Instrumentación<sup>2</sup>:** el modelo cuenta con 10 parámetros ajustables organizados en 3 grupos de controles (véase tabla 14.1). Como resultado del experimento, el programa despliega:

- 6 curvas organizadas en 4 gráficos (véase tabla 14.2).
- Una tabla de datos del comportamiento de 7 variables (véase tabla 14.3).

**Experimentos sugeridos:** el siguiente es el listado de experimentos sugeridos:

#### ***Experimento 14.1: Ruido en las mediciones.***

¿Qué efecto tiene el ruido en las mediciones sobre la estimación de la masa? Explore el efecto que tiene la amplitud del ruido en las mediciones del desplazamiento y la fuerza en la precisión de la estimación y el tiempo de estabilización de la estimación.

**Experimento 14.2: Estímulo.**

¿Qué efecto tiene el estímulo del sistema sobre el proceso de identificación? Explore el impacto de alterar la forma y magnitud de la fuerza que estimula la planta en el proceso de identificación.

**Experimento 14.3: Fricción.**

¿Cómo afecta la fricción de Coulomb la calidad de la identificación? Explore el efecto de la fricción de Coulomb en la precisión de la identificación de la masa, así como en el tiempo de estabilización de la identificación.

**Experimento 14.4: Reinicio de la identificación.**

¿Cada cuánto tiempo debería reiniciarse la identificación? El proceso implementado utiliza dos variables  $n(t)$  y  $d(t)$  que tienen un comportamiento inestable. Explore qué factores inciden en la evolución de estas variables y estime el tiempo en que deberían reiniciarse sus valores.

**Experimento 14.5: Filtrado de señales.**

¿Qué efecto tiene suavizar las mediciones de fuerza y desplazamiento? Explore el efecto de filtrar las señales de numerador y denominador. El filtro implementado consiste en un doble integrador.

**Experimento 14.6: Filtrado de mediciones.**

¿Qué efecto tiene filtrar las señales medidas? Implemente un filtro pasabajos para la señal de fuerza y otro para la señal de desplazamiento, y explore su impacto sobre el proceso de identificación.

**Experimento 14.7: Tiempo mínimo de identificación.**

¿A partir de qué instante de tiempo está disponible la identificación? En la implementación se utiliza el parámetro  $N_{min}$  para controlar el momento en que la relación  $d/n$  es útil para dar una identificación de la masa. Explore el efecto de este parámetro sobre el tiempo de identificación.

Tabla 14.1 Parámetros del experimento 14, “Masa con fricción”

<b>Título:</b>	Masa con fricción		
<b>Descripción:</b>	Identificación algebraica del parámetro de masa en un arreglo simple de masa deslizante con fricción de Coulomb		
<b>Créditos</b>	<b>Implementación</b> e-mail		
<b>Parámetros</b>			
Grupo	Nombre Modelica	Nombre	Descripción
<b>Fuerza</b>	<i>M.flagFuerza</i>	Tipo	Tipo de fuerza aplicada
	<i>M.Fa</i>	Amplitud	Magnitud de la fuerza aplicada
<b>Planta</b>	<i>M.mo</i>	Masa	Masa real del cuerpo
	<i>M.M.F_Coulomb</i>	Fricción	Fricción de Coulomb
	<i>M.smax</i>	X máxima	Máximo desplazamiento de la masa
<b>Identificación</b>	<i>stopTime</i>	Tiempo de simulación	Tiempo de simulación
	<i>M.Nx.stdev</i>	Ruido en x	Ruido en la medición del desplazamiento
	<i>M.Nu.stdev</i>	Ruido en F	Ruido en la medición de la fuerza
	<i>I.FlagFiltro</i>	Filtrado de señales	Filtrado de numerador y denominador
	<i>I.ne</i>	N min	Valor mínimo de n(t)

<sup>2</sup>La información en las tablas siguientes se muestra tal como aparece en la base de datos. Por esta razón hay datos tanto en español como en inglés.

Tabla 14.2 Figuras del experimento 14, "Masa con fricción"

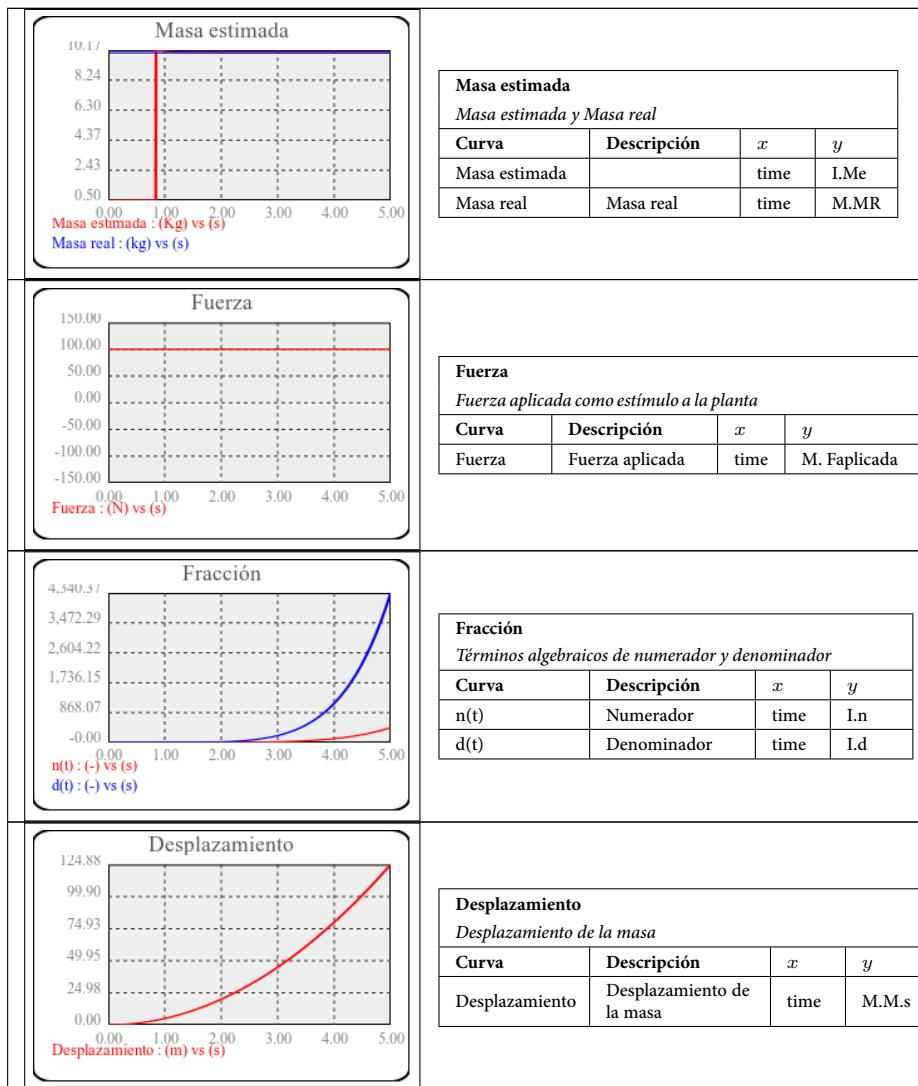


Tabla 14.3 Variables en la tabla de resultados del experimento 14, "Masa con fricción"

Variable	Descripción	Unidades
time	time	s
I.Me	Masa estimada	kg
M.M.R		kg

M.Faplicada	Amplitud de la fuerza aplicada	N
I.n	numerador de 1/M <sub>e</sub>	-
I.d	denominador de 1/M <sub>e</sub>	-
M.M.s	Absolute position of center of component (s = flange_a.s + L/2 = flange_b.s - L/2)	m

### 14.3 LA IMPLEMENTACIÓN

La figura 14.4 ilustra la implementación que se explica a continuación:

- La clase `idmasa` (véase archivo `idmasa.mo`) contiene un objeto de la clase `masa` y otro de la clase `identificador` que se conectan a través de las variables  $f_m(t)$  y  $x_m(t)$ , que representan las mediciones de fuerza y desplazamiento, respectivamente.
- $M$  es un objeto de la clase `MassWithStopAndFriction` de la Modelica Standard Library. Esta clase implementa una masa deslizante con varios tipos de fricción y un deslizamiento acotado entre  $-S_m$  y  $S_m$ . En esta implementación solo se utiliza la fricción de Coulomb, cuyo valor se ajusta a través de  $F_C$ . El valor de la masa se ajusta mediante  $M_o$ .
- La fuerza aplicada  $f(t)$  y el desplazamiento  $x(t)$  se miden. En cada una de las mediciones se adiciona ruido ( $n_f(t)$  y  $n_x(t)$ ), cuyas amplitudes se ajustan con  $R_f$  y  $R_x$ . Para generar el ruido aditivo se ha hecho uso de la solución propuesta en [For02]; esta solución emplea archivos externos en lenguaje C para la inicialización (véase archivo `ext_initRandNormal.c`) y uso (véase archivo `ext_RandNormal.c`) del generador de números aleatorios.
- La fuerza aplicada puede ser sinusoidal o constante. La selección se realiza mediante  $TF$ . La amplitud de la fuerza se ajusta con  $F_a$ .
- La clase `identificador` (véase archivo `identificador.mo`) es del tipo `partial`. La clase `identificadorBase` (véase archivo `identificadorBase.mo`) completa la implementación mediante la definición de  $\mathcal{N}$  y  $\mathcal{D}$  (ecuación 14.26).
- En el proceso de identificación se puede utilizar o no el filtro de doble integración que opera sobre las salidas de  $\mathcal{N}$  y  $\mathcal{D}$ . La selección se controla con  $F_i$ .

- Para evitar la indeterminación de la ecuación 14.23, se utiliza el condicional que evalúa si  $|n(t)| > N_m$ . En caso de no cumplir el condicional, el valor estimado de  $M_e$  no se altera ( $Me=pre(Me)$ ).

### 14.3.1 Listado de archivos

La tabla 14.4 muestra el listado de los archivos fuente de la implementación del modelo.

Tabla 14.4 Archivos del modelo

Número	Archivo
14.1	identificador.mo
14.2	identificadorBase.mo
14.3	identificadorFriccion.mo
14.4	idmasa.mo
14.5	masa.mo
14.6	ext_initRandNormal.c
14.7	ext_RandNormal.c

Archivo 14.1 identificador.mo

```

partial block identificador
  input Real x;
  input Real u;
  Real n(start=0,unit="-") "numerador de 1/Me";
  Real d(start=0,unit="-") "denominador de 1/Me";
  Real t(unit="s") "tiempo";
  parameter Real to(unit="s")=0;
  parameter Real ne=0.01;
  parameter Real Meo(unit="g")=0.5;
  output Real Me(start=Meo,unit="kg") "Masa estimada";
  Modelica.Blocks.Math.Abs nAbs;
  Real Meinv(unit="1/Kg");
  Real nttmp,dttmp;
  Modelica.Blocks.Continuous.Integrator In,Id,In1,Id1;
  parameter Integer FlagFiltro=1;
equation
  t=time;
  connect(In1.u,nttmp);
  connect(Id1.u,dttmp);
  connect(In.u,In1.y);
  connect(Id.u,Id1.y);
  connect(nAbs.u,n);
algorithm
  n:=if FlagFiltro==0 then nttmp else In.y;
  d:=if FlagFiltro==0 then dttmp else Id.y;
  Me:=if nAbs.y<ne then pre(Me) else d/n;

```

```

Meinv:=1/Me;
end identificador;
```

Archivo 14.2 identificadorBase.mo

```

block identificadorBase
  extends identificador;
  Real z1(start=0),z2(start=0),w1(start=0),w2(start=0);
equation
  nttmp = z1 + (t-to)^2*x;
  der(z1) = z2 - 4*(t-to)*x;
  der(z2) = 2*x;
  dttmp = w1;
  der(w1) = w2;
  der(w2) = (t-to)^2*u;
end identificadorBase;
```

Archivo 14.3 identificadorFriccion.mo

```

block identificadorFriccion
  extends identificador;
  Real z3(start=0),z4(start=0),z5(start=0),w3(start=0),w4(start=0),w5(
    start=0);
equation
  nttmp = z3 - (t-to)^3*x;
  der(z3) = z4 + 9*(t-to)^2*x;
  der(z4) = z5 - 18*(t-to)^1*x;
  der(z5) = 6*x;
  dttmp = w3;
  der(w3) = w4;
  der(w4) = w5 - (t-to)^3*u;
  der(w5) = 3*(t-to)^2*u;
end identificadorFriccion;
```

Archivo 14.4 idmasa.mo

```

model idmasa
  masa M;
  identificadorBase I;
equation
  connect(M.Um.y, I.u);
  connect(M.Xm.y, I.x);
end idmasa;
```

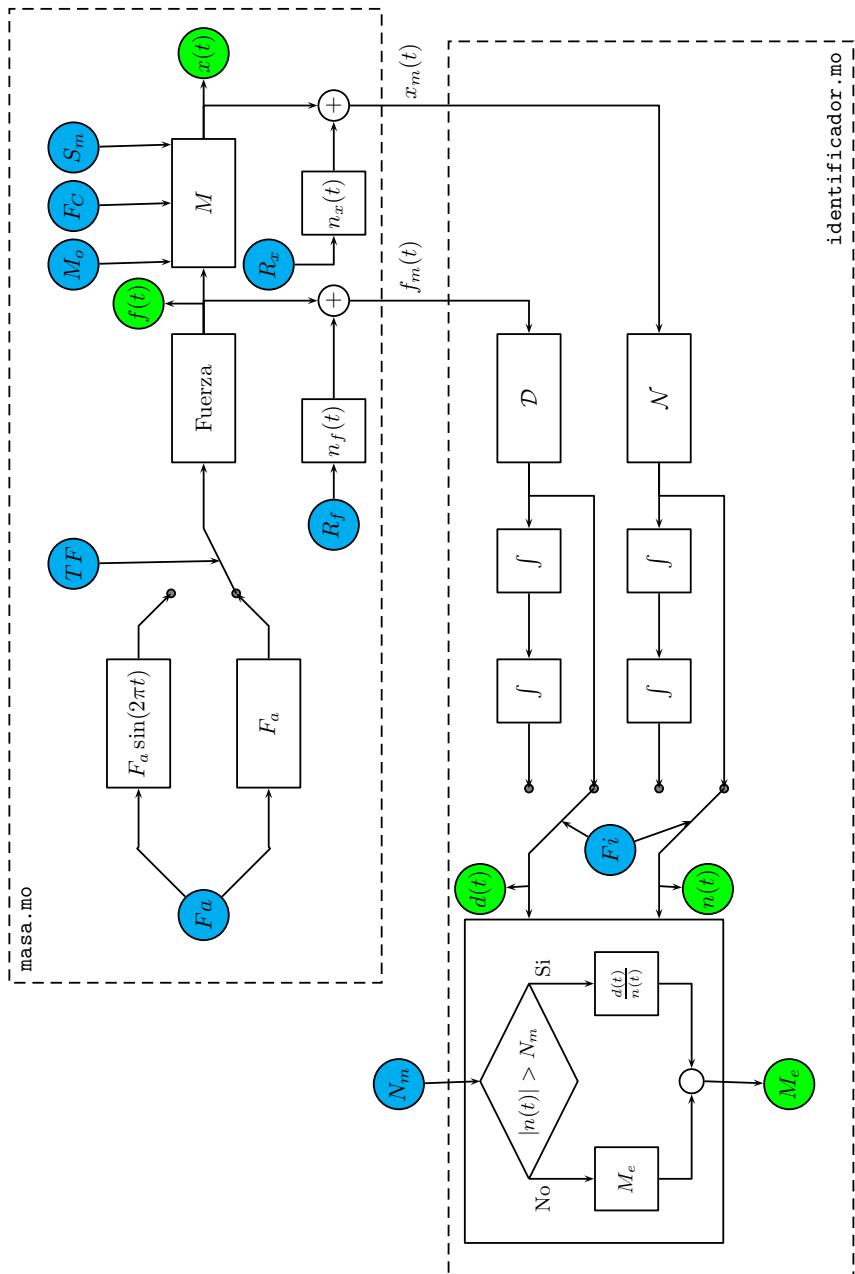


Figura 14.4 Esquema de la implementación. En azul los parámetros ajustables. En verde las variables graficadas

## Archivo 14.5 masa.mo

```

model masa
  Modelica.Blocks.Sources.Sine C1(amplitude=Fa) "Fuente sinusoidal";
  Modelica.Blocks.Sources.Constant C2(k=Fa) "Fuente constante";
  Modelica.Mechanics.Translational.Sources.Force F "Fuerza aplicada";
  Modelica.Mechanics.Translational.Components.Fixed fixed "soporte";
  Modelica.Mechanics.Translational.Components.MassWithStopAndFriction M
  (
    L=1,
    s(start=0,fixed=true),
    v(start=0,fixed=true),
    smax=smax,
    smin=-smax,
    m=mo,
    F_prop=0,
    F_Coulomb=0.1,
    F_Stribeck=0,
    fexp=0) "masa con fricción y parada";
  Modelica.SIunits.Force Faplicada "Amplitud de la fuerza aplicada";
  parameter Real mo=10;
  parameter Real Fa(unit="N")=100;
  parameter Real smax(unit="m")=250;
  Modelica.SIunits.Mass MR(start=mo);
  WhiteNoise.NoiseNormal Nu(stdev=0.01,tSample=0.01) "Ruido en medición
  de la fuerza";
  WhiteNoise.NoiseNormal Nx(stdev=0.01,tSample=0.01) "Ruido en medición
  del desplazamiento";
  Modelica.Blocks.Math.Add Um "Fuerza medida";
  Modelica.Blocks.Math.Add Xm "Desplazamiento medido";
  parameter Integer flagFuerza=1 "selector de fuerza";
equation
  F.f=if flagFuerza==0 then C1.y else C2.y ;
  Faplicada=F.f;
  connect(F.flange , M.flange_a);
  connect(fixed.flange , F.support);
  connect(M.s,Xm.u1);
  connect(Nx.y,Xm.u2);
  connect(F.f , Um.u1);
  connect(Nu.y,Um.u2);
  der(MR)=0;
end masa;

```

## Archivo 14.6 ext\_initRandNormal.c

```

#include <math.h>
#include <limits.h>

void ext_initRandomNormal()
{
    srand(time(NULL));
}

```

Archivo 14.7 ext\_RandNormal.c

```
#include <math.h>
#include <limits.h>

double ext_RandomNormal(double timein)
{
    unsigned int seed = 0;
    double v1, v2, r;

    timein /= 100;
    seed = (timein - floor(timein)) * UINT_MAX;

    do
    {
        v1 = 2 * ((double) rand()) /((double) RAND_MAX) - 1;
        v2 = 2 * ((double) rand()) /((double) RAND_MAX) - 1;
        r = v1 * v1 + v2 * v2;
    } while((r >= 1.0) || (r == 0.0));

    return v1 * sqrt( - 2.0 * log(r) / r );
}
```



# A

## LICENCIAMIENTO

UNVirtualLab y los modelos alojados en UNVirtualLab se distribuye bajo licencia GNU Versión 3.0 (ver [Fou22]). A continuación se reproduce la licencia oficial en inglés, y una traducción no oficial al español tomada de [AAM<sup>+</sup>22].

### VERSIÓN OFICIAL EN INGLÉS

Copyright © 2007 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

**Abstract:** The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program—to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is

precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

## TERMS AND CONDITIONS

### 0. Definitions

“This License” refers to version 3 of the GNU General Public License.

“Copyright” also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

“The Program” refers to any copyrightable work licensed under this License. Each licensee is addressed as “you”. “Licensees” and “recipients” may be individuals or organizations.

To “modify” a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version” of the earlier work or a work “based on” the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate” a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey” a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

### 1. Source Code

The “source code” for a work means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component”, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work’s System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

## 2. Basic Permissions

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

## 3. Protecting Users' Legal Rights From Anti-Circumvention Law

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

## 4. Conveying Verbatim Copies

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

## 5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

## 6. Conveying Non-Source Forms

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

## 7. Additional Terms

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered “further restrictions” within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

#### 8. Termination

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

#### 9. Acceptance Not Required for Having Copies

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

#### 10. Automatic Licensing of Downstream Recipients

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party’s predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

#### 11. Patents

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor’s “contributor version”.

A contributor’s “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor’s essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant” such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient’s use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

#### 12. No Surrender of Others’ Freedom

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

**13. Use with the GNU Affero General Public License**

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

**14. Revised Versions of this License**

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

**15. Disclaimer of Warranty**

There is no warranty for the program, to the extent permitted by applicable law, except when otherwise stated in writing the copyright holders and/or other parties provide the program "as is" without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. the entire risk as to the quality and performance of the program is with you. should the program prove defective, you assume the cost of all necessary servicing, repair or correction.

**16. Limitation of Liability**

In no event unless required by applicable law or agreed to in writing will any copyright holder, or any other party who modifies and/or conveys the program as permitted above, be liable to you for damages, including any general, special, incidental or consequential damages arising out of the use or inability to use the program (including but not limited to loss of data or data being rendered inaccurate or losses sustained by you or third parties or a failure of the program to operate with any other programs), even if such holder or other party has been advised of the possibility of such damages.

**17. Interpretation of Sections 15 and 16**

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

## END OF TERMS AND CONDITIONS

### How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.>

Copyright (C) <text><year> <name of author>

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <<http://www.gnu.org/licenses/>>.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

<program> Copyright (C) <year> <name of author>

This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'. This is free software, and you are welcome to redistribute it under certain conditions; type `show c' for details.

The hypothetical commands show w and show c should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an “about box”.

You should also get your employer (if you work as a programmer) or school, if any, to sign a “copyright disclaimer” for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lGPL.html>.

## A.1 TRADUCCIÓN NO OFICIAL

*Esta es una traducción no oficial al español de la GNU General Public License. No ha sido publicada por la Free Software Foundation, y no establece legalmente las condiciones de distribución para el software que usa la GNU GPL –estas condiciones se establecen solamente por el texto original, en inglés, de la GNU GPL. Sin embargo, esperamos que esta traducción ayude a los hispanohablantes a entender mejor la GNU GPL.*

*This is an unofficial translation of the GNU General Public License into Spanish. It was not published by the Free Software Foundation, and does not legally state the distribution terms for software that uses the GNU GPL –only the original English text of the GNU GPL does that. However, we hope that this translation will help Spanish speakers understand the GNU GPL better.*

Copyright © 2007 Free Software Foundation, Inc. <http://fsf.org/>

Se permite la copia y distribución de copias literales de este documento, pero no se permite su modificación.

**Abstract:** La Licencia Pública General de GNU es una licencia libre, bajo “copyleft”, para software y otro tipo de obras.

Las licencias para la mayoría del software y otras obras de carácter práctico están diseñadas para privarle de la libertad de compartir y modificar las obras. Por el contrario, la Licencia Pública General de GNU pretende garantizar su libertad de compartir y modificar todas las versiones de un programa –para cerciorar que permanece como software libre para todos sus usuarios. Nosotros, la Free Software Foundation, usamos la Licencia Pública General de GNU para la mayoría de nuestro software; la cual se aplica también a cualquier otra obra publicada de esta forma por parte de sus autores. Usted también puede aplicarla a sus programas.

Cuando hablamos de software libre (free software), nos referimos a libertad, no a precio. Nuestras Licencias Públicas Generales están diseñadas para garantizar su libertad de distribuir copias de software libre (y cobrar por ellas si lo desea),

recibir el código fuente o poder obtenerlo si quiere, modificar el software o usar fragmentos de él en sus nuevos programas, y que sepa que puede hacer esas cosas.

Para proteger sus derechos, necesitamos impedir que otros le denieguen esos derechos o que le pidan que renuncie a ellos. Por ello, tiene ciertas responsabilidades si distribuye copias del software, o si lo modifica: la responsabilidad de respetar la libertad de otros.

Por ejemplo, si distribuye copias de un programa, bien sea gratis o por una tasa, debe transferirles a los que lo reciban las mismas libertades que usted recibió. Debe asegurarse que ellos, también, reciben o pueden obtener el código fuente. Y debe mostrarles estos términos para que ellos puedan conocer sus derechos.

Los desarrolladores que usan la GNU GPL protegen tus derechos con dos pasos: (1) haciendo valer el derecho de propiedad intelectual en el software, y (2) ofreciéndole esta Licencia que le da el permiso legal para copiarlo, distribuirlo y/o modificarlo.

Para la protección de autores y desarrolladores, la GPL explica claramente que no hay garantía para este software libre. Por el bien tanto de usuarios como de autores, la GPL requiere que las versiones modificadas sean marcadas como con cambios, de forma que sus problemas no puedan ser atribuidos de forma errónea a autores de versiones previas.

Algunos dispositivos están diseñados para denegar a los usuarios el acceso para instalar o ejecutar versiones modificadas del software en su interior, a pesar de que el fabricante puede hacerlo. Esto es fundamentalmente incompatible con el objetivo de proteger la libertad de los usuarios de modificar el software. El modelo sistemático de este abuso ocurre en el ámbito de los productos de uso personal, lo cual es precisamente donde es más inaceptable. Por consiguiente, hemos diseñado esta versión de la GPL para prohibir la práctica de estos productos. Si estos problemas surgen de forma substancial en otro dominios, estamos preparados para extender esta disposición a esos dominios en futuras versiones de la GPL, así como sea necesario para proteger la libertad de los usuarios.

Por último, todo programa es amenazado constantemente por las patentes de software. Los Estados no deberían permitir patentes que restringen el desarrollo y el uso de software en ordenadores de propósito general, pero en aquellos que lo hacen, deseamos evitar el peligro particular de que las patentes aplicadas a un programa libre podrían convertirlo de forma efectiva en propietario. Para prevenir esto, la GPL garantiza que las patentes no pueden ser utilizadas para hacer que el programa no sea libre.

Los términos exactos y las condiciones para la copia, distribución y modificación se exponen a continuación.

## TÉRMINOS Y CONDICIONES

### 0. Definiciones

“Esta Licencia” se refiere a la versión 3 de la Licencia Pública General de GNU.

“Derechos de Autor (“Copyright”)” también incluye a las leyes similares a la de derechos de autor (“copyright”) que se apliquen a otro tipo de obras, tales como las máscaras usadas en la fabricación de semiconductores.

“El Programa” se refiere a cualquier obra con derechos de autor (“copyright”) bajo esta Licencia. Cada licenciatario es tratado como “usted”. Los “Licenciatarios” y los “destinatarios” pueden ser individuos u organizaciones.

“Modificar” una obra quiere decir copiar de ella o adaptar parte o la totalidad de la obra de una forma que se requieren permisos de derechos de autor (“copyright”), distintos de los de hacer una copia exacta. La obra resultante es llamada “versión modificada” de la obra previa o una obra “basada en” la obra previa.

Una “obra amparada” significa o el Programa sin modificar o una obra basada en el Programa.

“Difundir” una obra significa hacer cualquier cosa con ella que, sin permiso, le haría responsable de forma directa o indirecta de infringir la ley correspondiente de derechos de autor (“copyright”), excepto ejecutarla en un ordenador o modificar una copia privada. La difusión incluye copiar, la distribución (con o sin modificación), hacerla disponible para el público, y en algunos países también otras actividades.

“Transmitir” una obra quiere decir cualquier tipo de difusión que permita a otras partes hacer o recibir copias. La mera interacción con un usuario a través de una red informática, sin la transferencia de una copia, no es transmitir.

Una interfaz interactiva de usuario muestra “Avisos Legales Apropriados” en la medida que incluye una característica visible práctica y destacable que (1) muestra un aviso apropiado de derechos de autor (“copyright”), e (2) informa al usuario de que no hay garantía para la obra (excepto las garantías proporcionadas), que los licenciatarios pueden transmitir la obra bajo esta Licencia, y cómo ver una copia de esta Licencia. Si la interfaz presenta una lista de comandos de usuario u opciones, como un menú, un elemento destacado en la lista satisface este criterio.

### 1. Código fuente

El “código fuente” de una obra significa la forma preferida de trabajo para hacerle modificaciones. “Código objeto” es cualquier forma no-fuente de una obra.

Una “Interfaz Estándar” significa una interfaz que es un estándar oficial definido por un cuerpo de estándares reconocido o, en el caso de interfaces especificadas para un lenguaje de programación en particular, una que es extensamente utilizada entre los desarrolladores que trabajan en ese lenguaje.

Las “Bibliotecas del Sistema” de una obra ejecutable incluyen cualquier cosa, diferente de la obra como un todo, que (a) están incluidas en la forma normal de paquetizado de un Componente Importante, y (b) sirve solo para habilitar el uso de la obra con ese Componente Importante, o para implementar una Interfaz Estándar para la cual la implementación está disponible para el público en forma de código fuente. Un “Componente Importante”, en este contexto, significa un componente esencial importante (kernel, sistema de ventanas, etcétera) del sistema operativo en concreto (si hubiese) en el cual el ejecutable funciona, o un compilador utilizado para producir la obra, o un intérprete de código objeto utilizado para hacerlo funcionar.

La “Fuente Correspondiente” de una obra en forma de código objeto significa todo el código fuente necesario para generar, instalar, y (para una obra ejecutable) hacer funcionar el código objeto y modificar la obra, incluyendo scripts para controlar dichas actividades. Sin embargo, ello no incluye la obra de las Bibliotecas del Sistema, o herramientas de propósito general o programas de libre disponibilidad general los cuales son usados sin modificaciones para la realización de dichas actividades, pero que no son parte de la obra. Por ejemplo, la Fuente Correspondiente incluye ficheros de definición de interfaces asociados a los ficheros fuente para la obra, y el código fuente para bibliotecas compartidas y subprogramas enlazados dinámicamente para los que la obra está específicamente diseñado para requerir, tales como comunicación de datos intrínseca o flujo de control entre aquellos subprogramas y otras partes de la obra.

La Fuente Correspondiente es necesario que no incluya nada que los usuarios puedan regenerar automáticamente desde otras partes de la Fuente Correspondiente.

La Fuente Correspondiente de una obra en forma de código fuente es la obra en sí.

## 2. Permisos básicos

Todos los derechos concedidos bajo esta Licencia se conceden durante la duración de los derechos de autor (“copyright”) del Programa, y son irrevocables siempre que se cumplan las condiciones establecidas. Esta Licencia afirma explícitamente su ilimitado permiso para ejecutar el Programa sin modificar. La salida de la ejecución de una obra amparada está amparada por esta Licencia solo si la salida, dado su contenido, constituye una obra amparada. Esta Licencia reconoce sus derechos de uso razonable u otro equivalente, según lo establecido por la ley de derechos de autor (“copyright”).

Usted podrá realizar, ejecutar y difundir obras amparadas que usted no transmite, sin condición alguna, siempre y cuando no tenga otra licencia vigente. Podrá distribuir obras amparadas a terceros con el único propósito de que ellos hagan modificaciones exclusivamente para usted, o proporcionarle ayuda para ejecutar estas obras, siempre y cuando cumpla con los términos de esta Licencia en la transmisión de todo el material del cual usted no controle los derechos de autor (“copyright”). Aquellos que realicen o ejecuten las obras amparadas por usted, deben hacerlo exclusivamente en su nombre, bajo su dirección y control, en los términos que le prohíban realizar ninguna copia de su trabajo con derechos de autor (“copyright”) fuera de su relación con usted.

La transmisión bajo otras circunstancias se permite únicamente bajo las condiciones expuestas a continuación. No está permitido sublicenciar, la sección 10 hace que sea innecesario.

## 3. Protección de los Derechos Legales de los Usuarios frente a la Ley Antievasión

Ninguna obra amparada debe considerarse parte de una medida tecnológica efectiva, a tenor de lo establecido en cualquier ley aplicable que cumpla las obligaciones expresas en el artículo 11 del tratado de derechos de autor (“copyright”) de WIPO adoptado el 20 de diciembre de 1996, o leyes similares que prohíban o restrinjan la evasión de tales medidas.

Cuando transmita una obra amparada, renuncia a cualquier poder legal para prohibir la evasión de medidas tecnológicas mientras tales evasiones se realicen en ejercicio de derechos amparados por esta Licencia respecto a la obra amparada; además, usted renunciará a cualquier intención de limitar el uso o modificación del trabajo con el objetivo de imponer, contra el trabajo de los usuarios, sus derechos legales o los de terceros para prohibir la evasión de medidas tecnológicas.

## 4. Transmisión de copias literales

Usted podrá distribuir copias literales del código fuente del Programa tal cual lo ha recibido, por cualquier medio, siempre que publique visible y apropiadamente en cada copia el correspondiente aviso de derechos de autor (“copyright”); mantenga intactos todos los avisos que establezcan que esta Licencia y cualquier cláusula no-permisiva añadida acorde con la cláusula 7 son aplicables al código; mantenga intactos todos los avisos de ausencia de garantía; y proporcione a todos los destinatarios una copia de esta Licencia junto con el Programa.

Usted podrá cobrar cualquier importe o no cobrar nada por cada copia que distribuya, y podrá ofrecer soporte o protección de garantía mediante un pago.

## 5. Transmisión de versiones modificadas de la fuente

Usted puede transmitir una obra basada en el Programa, o las modificaciones para generarla a partir del Programa, en la forma de código fuente bajo los términos de la sección 4, suponiendo que además cumpla las siguientes condiciones:

- a) La obra debe incluir avisos destacados indicando que usted la ha modificado y dando una fecha pertinente.
- b) La obra debe incluir avisos destacados indicando que está liberada bajo esta Licencia y cualquier otra condición añadida bajo la sección 7. Este requerimiento modifica los requerimientos de la sección 4 de “mantener intactos todos los avisos”.
- c) Usted debe licenciar la obra entera, como una unidad, bajo esta Licencia para cualquier persona que esté en posesión de una copia. Esta Licencia se aplicará por consiguiente, junto con cualquier término aplicable adicional de la sección 7, a la totalidad de la obra, y a todos sus componentes, independientemente de como estén empaquetados. Esta Licencia no da permiso para licenciar la obra de otra forma, pero no invalida esos permisos si usted los ha recibido de forma separada.
- d) Si la obra tiene interfaces de usuario interactivas, cada una debe mostrar los Avisos Legales Apropiados; sin embargo, si el Programa tiene interfaces interactivas que no muestren los Avisos Legales Apropiados, tampoco es necesario que su obra lo haga.

Una recopilación de una obra amparada con otras obras separadas e independientes, que no son por su naturaleza extensiones de la obra amparada, y que no se combinan con ella con el fin de formar un programa más grande, en o sobre un volumen de un medio de almacenamiento o distribución, es llamado un “agregado” si la recopilación y su resultante derechos de autor (“copyright”) no son usados para limitar el acceso o los derechos legales de los usuarios de la recopilación más allá de lo que las obras individuales permitan. La inclusión de una obra amparada en un agregado no provoca que esta Licencia se aplique a los otros componentes del agregado.

#### 6. Transmisión en forma de no-fuente

Usted puede transmitir una obra amparada en forma de código objeto bajo los términos de las secciones 4 y 5, siempre que también transmite la Fuente Correspondiente legible por una máquina bajo los términos de esta Licencia, de una de las siguientes formas:

- a) Transmitir el código objeto en, o embebido en, un producto físico (incluyendo medios de distribución físicos), acompañado de la Fuente Correspondiente en un medio físico duradero habitual para el intercambio de software.
- b) Transmitir el código objeto en, o embebido en, un producto físico (incluyendo medios de distribución físicos), acompañado de un ofrecimiento escrito, válido durante al menos tres años y válido mientras usted ofrezca recambios o soporte para ese modelo de producto, de dar a cualquiera que posea el código objeto o (1) una copia de la Fuente Correspondiente de todo el software en el producto amparado por esta Licencia, en un medio físico duradero habitual para el intercambio de software, por un precio no más elevado que el coste razonable de la realización física de la transmisión de la fuente, o (2) acceso para copiar la Fuente Correspondiente de un servidor de red sin costo alguno.
- c) Transmitir copias individuales del código objeto con una copia del ofrecimiento escrito de proveer la Fuente Correspondiente. Esta alternativa está permitida solo ocasionalmente sin fines comerciales, y solo si usted ha recibido el código objeto con ese ofrecimiento, de acuerdo con la subsección 6b.
- d) Transmitir el código objeto ofreciendo acceso desde un lugar determinado (gratuitamente o mediante pago), y ofrecer acceso equivalente a la Fuente Correspondiente de la misma manera en el mismo lugar sin cargo adicional. No es necesario exigir a los destinatarios que copien la Fuente Correspondiente junto con el código objeto. Si el lugar para copiar el código objeto es un servidor de red, la Fuente Correspondiente puede estar en un servidor diferente (gestionado por usted o un tercero) que soporte facilidades de copia equivalentes, siempre que mantenga instrucciones claras junto al código objeto especificando dónde encontrar la Fuente Correspondiente. Independientemente de qué servidor albergue la Fuente Correspondiente, usted seguirá estando obligado a asegurar que está disponible durante el tiempo que sea necesario para satisfacer estos requisitos.
- e) Transmitir el código objeto usando una transmisión peer-to-peer, siempre que informe a los otros usuarios donde se ofrece el código objeto y la Fuente Correspondiente de la obra al público general de forma gratuita bajo la subsección 6d.

Una porción separable del código objeto, cuyo código fuente está excluido de la Fuente Correspondiente, como una Biblioteca del Sistema, no necesita ser incluida en la distribución del código objeto de la obra.

Un “Producto de Usuario” es o (1) un “producto de consumo”, lo que significa cualquier propiedad tangible personal que es usada habitualmente con fines personales, familiares o domésticos, o (2) cualquier cosa diseñada o vendida para ser incorporada en una vivienda. A la hora de determinar cuando un producto es un producto de consumo, los casos dudosos serán resueltos en favor de la cobertura. Para un producto concreto recibido por un usuario concreto, “uso habitual” se refiere a un uso típico y común de esa clase de producto, sin tener en cuenta el estado del usuario concreto o la forma en la que el usuario concreto realmente use, o espera o se espera que use, el producto. Un producto es un producto de consumo independientemente de si el producto tiene usos

esencialmente comerciales, industriales o no comerciales, a menos que dicho uso constituya el único modo de uso significativo del producto.

La “Información de Instalación” de un Producto de Usuario quiere decir cualquier método, procedimiento, clave de autorización, u otra información requerida para instalar y ejecutar versiones modificadas de la obra amparada en ese Producto de Usuario a partir de una versión modificada de su Fuente Correspondiente. La información debe ser suficiente para garantizar que el funcionamiento continuado del código fuente modificado no es prevenido o interferido por el simple hecho de que ha sido modificado.

Si usted transmite una obra en código objeto bajo esta sección en, o con, o especificamente para usar en, un Producto de Usuario, y la transmisión tiene lugar como parte de una transacción en la cual el derecho de posesión y uso de un Producto de Usuario es transferido a un destinatario en perpetuidad o por un periodo establecido (independientemente de cómo se caracterice la operación), la Fuente Correspondiente transmitida bajo esta sección debe estar acompañada de la Información de Instalación. Pero este requisito no se aplica si ni usted ni ningún tercero tiene la capacidad de instalar código objeto modificado en el Producto de Usuario (por ejemplo, la obra ha sido instalada en la ROM).

El requisito de proveer de la Información de Instalación no incluye el requisito de continuar proporcionando asistencia, garantía, o actualizaciones para una obra que ha sido modificada o instalada por el destinatario, o para un Producto de Usuario en el cual ha sido modificada o instalada. El acceso a una red puede ser denegado cuando la modificación en si afecta materialmente y adversamente el funcionamiento de la red o viola las reglas y protocolos de comunicación de la red.

La Fuente Correspondiente transmitida, y la Información de Instalación proporcionada, de acuerdo con esta sección debe estar en un formato que sea documentado públicamente (y con una implementación disponible para el público en formato de código fuente), y no deben necesitar contraseñas o claves particulares para la extracción, lectura o copia.

## 7. Términos adicionales

Los “Permisos adicionales” son términos que se añaden a los términos de esta Licencia haciendo excepciones de una o más de una de sus condiciones. Los permisos adicionales que son aplicables al Programa entero deberán ser tratados como si estuvieran incluidos en esta Licencia, en la medida bajo la ley aplicable. Si los permisos adicionales solo son aplicables a parte del Programa, esa parte debe ser usada separadamente bajo esos permisos, pero el Programa completo queda bajo la autoridad de esta Licencia sin considerar los permisos adicionales.

Cuando se transmite una copia de una obra derivada, se puede opcionalmente quitar cualesquiera permisos adicionales de esa copia, o de cualquier parte de ella. Los permisos adicionales pueden ser escritos para requerir su propia eliminación bajo ciertos casos cuando se modifica la obra. Se pueden colocar permisos adicionales en material, añadidos a una obra derivada, para los cuales se establecen o se pueden establecer los permisos de derechos de autor (“copyright”) apropiados.

No obstante cualquier otra disposición de esta Licencia, para el material que se añade a una obra derivada, se puede (si está autorizado por los titulares de los derechos de autor (“copyright”) del material) añadir los términos de esta Licencia con los siguientes términos:

- a) Ausencia de garantía o limitación de responsabilidad diferente de los términos de las secciones 15 y 16 de esta Licencia; o
- b) Exigir la preservación de determinados avisos legales razonables o atribuciones de autor en ese material o en los Avisos Legales Apropiados mostrados por los obras que lo contengan; o
- c) Prohibir la tergiversación del origen de ese material, o requerir que las versiones modificadas del material se marquen de maneras razonables como diferentes de la versión original; o
- d) Limitar el uso con fines publicitarios de los nombres de los licenciantes o autores del material; o
- e) Negarse a ofrecer derechos concedidos por leyes de registro para el uso de algunos nombres comerciales, marcas registradas o marcas de servicio; o
- f) Exigir la compensación de los licenciantes y autores de ese material por cualquiera que distribuya el material (o versiones modificadas del mismo) estableciendo obligaciones contractuales de responsabilidad sobre el destinatario, por cualquier responsabilidad que estas obligaciones contractuales impongan directamente sobre los licenciantes y autores.

Todos los demás términos adicionales no permisivos son consideradas “restricciones extra” en el sentido de la sección 10. Si el Programa, tal cual se recibió, o cualquier parte del mismo, contiene un aviso indicando que se encuentra cubierto por esta Licencia junto con un término que es otra restricción, se puede quitar ese término. Si un documento de licencia contiene una restricción adicional, pero permite relicenciar o redistribuir bajo esta Licencia, se puede añadir a un material de la obra derivada bajo los términos de ese documento de licencia, a condición de que dicha restricción no sobreviva el relicenciamiento o redistribución.

Si se añaden términos a una obra derivada de acuerdo con esta sección, se debe colocar, en los archivos fuente involucrados, una declaración de los términos adicionales aplicables a esos archivos, o un aviso indicando donde encontrar los términos aplicables.

Las términos adicionales, permisivos o no permisivos, pueden aparecer en forma de una licencia escrita por separado, o figurar como excepciones; los requisitos anteriores son aplicables en cualquier forma.

#### 8. Conclusiones

Usted no podrá propagar o modificar una obra amparada salvo lo expresamente permitido por esta Licencia. Cualquier intento diferente de propagación o modificación será considerado nulo y automáticamente se anularán sus derechos bajo esta Licencia (incluyendo las licencias de patentes concedidas bajo el tercer párrafo de la sección 11).

Sin embargo, si usted deja de violar esta Licencia, entonces su licencia de un titular de los derechos de autor ("copyright") correspondiente será restituida (a) provisionalmente, a menos que y hasta que el titular de los derechos de autor ("copyright") explícita y finalmente termine su licencia, y (b) permanentemente, si el titular del copyright no le ha notificado su violación por algún medio razonable antes de los 60 días siguientes a la cesación.

Además, su licencia de un titular de los derechos de autor ("copyright") correspondiente será restituida permanentemente si el titular de los derechos de autor ("copyright") le notifica la violación por algún medio razonable, siendo ésta la primera vez que recibe la notificación de violación de esta Licencia (para cualquier obra) de ese titular de los derechos de autor ("copyright"), y usted subsana la violación antes de 30 días después de la recepción de la notificación.

La cancelación de sus derechos bajo esta sección no da por canceladas las licencias de terceros que hayan recibido copias o derechos de usted bajo esta Licencia. Si sus derechos han sido cancelados y no fueran renovados de manera permanente, usted no cumple los requisitos para recibir nuevas licencias para el mismo material bajo la sección 10.

#### 9. Aceptación no obligatoria por tenencia de copias

Usted no está obligado a aceptar esta Licencia por recibir o ejecutar una copia del Programa. La propagación adicional de una obra amparada surgida únicamente como consecuencia de usar una transmisión peer-to-peer para recibir una copia tampoco requiere aceptación. Sin embargo, esta Licencia solo le otorga permiso para propagar o modificar cualquier obra amparada. Estas acciones infringen los derechos de autor ("copyright") si usted no acepta esta Licencia. Por lo tanto, al modificar o distribuir una obra amparada, usted indica que acepta esta Licencia para poder hacerlo.

#### 10. Herencia automática de licencia para destinatarios

Cada vez que transmita una obra amparada, el destinatario recibirá automáticamente una licencia de los licenciatarios originales, para ejecutar, modificar y distribuir esa obra, sujeto a esa Licencia. Usted no será responsable de asegurar el cumplimiento de esta Licencia por terceros.

Una "transacción de entidad" es una transacción que transfiere el control de una organización, o sustancialmente todos los bienes de una, o subdivide una organización, o fusiona organizaciones. Si la propagación de una obra amparada surge de una transacción de entidad, cada parte en esa transacción que reciba una copia de la obra también recibe todas las licencias de la obra que la parte interesada tuviese o pudiese ofrecer según el párrafo anterior, además del derecho a tomar posesión de las Fuentes Correspondientes de la obra a través del predecesor interesado, si el predecesor tiene o puede conseguirla con un esfuerzo razonable.

Usted no podrá imponer ninguna restricción posterior en el ejercicio de los derechos otorgados o concedidos bajo esta Licencia. Por ejemplo, usted no puede imponer un pago por licencia, derechos u otros cargos por el ejercicio de los derechos otorgados bajo esta Licencia, y no puede iniciar litigios (incluyendo demandas o contrademandas en pleitos) alegando cualquier reclamación de violación de patentes por cambiar, usar, vender, ofrecer en venta o importar el Programa o alguna parte del mismo.

#### 11. Patentes

Un "colaborador" es un titular de los derechos de autor ("copyright") que autoriza, bajo los términos de la presente Licencia, el uso del Programa o una obra en la que se base el Programa. La obra así licenciada se denomina "versión en colaboración" del colaborador.

Las "demandas de patente esenciales" del colaborador son todas las reivindicaciones de patentes poseídas o controladas por el colaborador, ya se encuentren adquiridas o hayan sido adquiridas con posterioridad, que sean infringidas de alguna manera, permitidas por esta Licencia, al hacer, usar o vender la versión en colaboración, pero sin incluir demandas que solo sean infringidas como consecuencia de modificaciones posteriores de la versión en colaboración. Para los propósitos de esta definición, "control" incluye el derecho de conceder sublicencias de patente de forma consistente con los requisitos establecidos en la presente Licencia.

Cada colaborador le concede una licencia de la patente no-exclusiva, global y libre de regalías bajo las demandas de patente esenciales del colaborador, para hacer, usar, modificar, vender, ofrecer para venta, importar y otras formas de ejecución, modificación y difusión del contenido de la versión en colaboración.

En los siguientes tres párrafos, una “licencia de patente” se define como cualquier acuerdo o compromiso expreso, cualquiera que sea su denominación, que no imponga una patente (como el permiso expreso para ejecutar una patente o acuerdos para no imponer demandas por infracción de patente). “Conceder” una licencias de patente de este tipo a un tercero significa hacer tal tipo de acuerdo o compromiso que no imponga una patente al tercero.

Si usted transmite una obra amparada, conociendo que está amparada por una licencia de patente, y las Fuentes Correspondientes no se encuentran disponibles de forma pública para su copia, sin cargo alguno y bajo los términos de esta Licencia, ya sea a través de un servidor público o mediante cualquier otro medio, entonces usted deberá (1) hacer que las Fuentes Correspondientes sean públicas, o (2) tratar de eliminar los beneficios de la licencia de patente para esta obra en particular, o (3) tratar de extender, de manera compatible con los requisitos de esta Licencia, la licencia de patente a terceros. “Conocer que está afectado” significa que usted tiene conocimiento real de que, para la licencia de patente, la distribución de la obra amparada en un país, o el uso de la obra amparada por sus destinatarios en un país, infringiría una o más patentes existentes en ese país que usted considera válidas por algún motivo.

Si en virtud de o en conexión con alguna transacción o acuerdo, usted transmite, o difunde con fines de distribución, una obra amparada, y concede una licencia de patente para algún tercero que reciba la obra amparada, y les autorice a usar, transmitir, modificar o difundir una copia específica de la obra amparada, entonces la licencia de patente que usted otorgue se extiende automáticamente a todos los receptores de la obra amparada y cualquier obra basada en ella.

Una licencia de patente es “discriminatoria” si no incluye dentro de su ámbito de cobertura, prohíbe el ejercicio de, o está condicionada a no ejercitar uno o más de los derechos que están específicamente otorgados por esta Licencia. Usted no debe transmitir una obra amparada si está implicado en un acuerdo con terceros que esté relacionado con el negocio de la distribución de software, en el que usted haga pagos a terceros relacionados con su actividad de distribución de la obra, bajo el que terceros conceden, a cualquier receptor de la obra amparada, una licencia de patente discriminatoria (a) en relación con las copias de la obra amparada transmitidas por usted (o copias hechas a partir de estas), o (b) principalmente para y en relación con productos específicos o compilaciones que contengan la obra amparada, a menos que usted forme parte del acuerdo, o que esa licencia de patente fuese concedida antes del 28 de marzo de 2007.

Ninguna cláusula de esta Licencia debe ser considerada como excluyente o limitante de cualquier otra licencia implicada u otras defensas legales a que pudiera tener derecho bajo la ley de propiedad intelectual vigente.

#### 12. No abandonar la libertad de otros

Si se le imponen condiciones (bien sea por orden judicial, acuerdo o de otra manera) que contradicen las condiciones de esta Licencia, estas no le eximen de las condiciones de esta Licencia. Si usted no puede transmitir una obra amparada de forma que pueda satisfacer simultáneamente sus obligaciones bajo esta Licencia y cualesquier otras obligaciones pertinentes, entonces, como consecuencia, usted no puede transmitirla. Por ejemplo, si usted está de acuerdo con los términos que le obligan a cobrar una regalía por la transmisión a aquellos a los que transmite el Programa, la única forma en la que usted podría satisfacer tanto esos términos como esta Licencia sería abstenerse completamente de transmitir el Programa.

#### 13. Utilización con la Licencia Pública General Afferro de GNU

A pesar de cualquier otra disposición de esta Licencia, usted tiene permiso para enlazar o combinar cualquier obra amparada con una obra licenciada bajo la Licencia Pública General Afferro de GNU en una única obra combinada, y para transmitir la obra resultante. Los términos de esta Licencia continuarán aplicándose a la parte que es la obra amparada, pero los requisitos particulares de la Licencia Pública General Afferro de GNU, sección 13, concernientes a la interacción a través de una red se aplicarán a la combinación como tal.

#### 14. Versiones revisadas de esta licencia

La Free Software Foundation puede publicar versiones revisadas y/o nuevas de la Licencia General Pública de GNU de vez en cuando. Cada nueva versión será similar en espíritu a la versión actual, pero puede diferir en detalles para abordar nuevos problemas o preocupaciones.

Cada versión recibe un número de versión distintivo. Si el Programa especifica que cierta versión numerada de la Licencia General Pública de GNU “o cualquier versión posterior” se aplica a él, usted tiene la opción de seguir los términos y condiciones de esa versión numerada o de cualquier versión posterior publicada por la Free Software Foundation. Si el Programa no especifica un número de versión de la Licencia General Pública de GNU, usted puede escoger cualquier versión publicada por la Free Software Foundation.

Si el Programa especifica que un representante puede decidir que versiones futuras de la Licencia General Pública de GNU pueden ser utilizadas, la declaración pública del representante de aceptar una versión permanentemente le autoriza a usted a elegir esa versión para el Programa.

Las versiones posteriores de la licencia pueden darle permisos adicionales o diferentes. No obstante, no se impone a ningún autor o titular de los derechos de autor obligaciones adicionales como resultado de su elección de seguir una versión posterior.

#### 15. Descargo de responsabilidad de garantía

No hay garantía para el programa, para la extensión permitida por la ley aplicable, excepto cuando se indique lo contrario por escrito, los titulares de los derechos de autor ("copyright") y/o terceros proporcionan el programa "tal cual" sin garantías de ningún tipo, bien sean explícitas o implícitas, incluyendo, pero no limitado a, las garantías implícitas de comercialización y aptitud para un propósito particular. El riesgo total en cuanto a calidad y rendimiento del programa es con usted. Si el programa presenta algún defecto, usted asume el costo de todas las revisiones necesarias, reparaciones o correcciones.

#### 16. Limitación de la responsabilidad

En ningún caso a menos que sea requerido por una ley aplicable o acuerdo escrito ninguno titular de los derechos de autor ("copyright"), o ningún tercero que modifique y/o transmita el programa como se permite anteriormente, será responsable ante usted por daños, incluyendo cualesquiera daños generales, particulares, imprevistos o derivados del uso o imposibilidad de uso del programa (incluyendo, pero no limitado a, la pérdida de datos, datos generados incorrectos, pérdidas sufridas por usted o por terceras personas, o los fallos del programa para operar con otros programas), incluso si dicho titular o un tercero ha sido advertido de la posibilidad de tales daños.

#### 17. Interpretación de las secciones 15 y 16

Si el descargo de responsabilidad de garantía y el límite de responsabilidad proporcionado anteriormente no tiene efectos legales de acuerdo a sus términos, los juzgados deberán aplicar la ley local que más se asemeje a una renuncia absoluta de la responsabilidad civil concerniente al Programa, a menos que una garantía o una asunción de responsabilidad acompañe a la copia del Programa como resultado del pago de una tasa.

## FIN DE LOS TÉRMINOS Y CONDICIONES

### Cómo aplicar estos términos a sus nuevos programas

Si desarrolla un nuevo programa, y quiere que sea lo más usado posible por el público, la mejor manera de conseguirlo es hacerlo software libre para que cualquiera pueda redistribuirlo y modificarlo bajo estos términos.

Para ello, añada la siguiente nota al programa. Lo más seguro es añadirla al principio de cada fichero fuente para declarar más efectivamente la exclusión de garantía; y cada fichero debe tener al menos la línea de "derechos de autor ("copyright")" y un puntero a donde se pueda encontrar la anotación completa.

<una línea para dar el nombre del programa y una breve idea de lo que hace>  
Copyright (C) <año> <nombre del autor>

Este programa es software libre: puede redistribuirlo y/o modificarlo bajo los términos de la Licencia General Pública de GNU publicada por la Free Software Foundation, ya sea la versión 3 de la Licencia, o (a su elección) cualquier versión posterior.

Este programa se distribuye con la esperanza de que sea útil pero SIN NINGUNA GARANTÍA; incluso sin la garantía implícita de MERCANTIBILIDAD o CALIFICADA PARA UN PROPÓSITO EN PARTICULAR. Vea la Licencia General Pública de GNU para más detalles.

Usted ha debido de recibir una copia de la Licencia General Pública de GNU junto con este programa. Si no, vea <<http://www.gnu.org/licenses/>>.

También añada información sobre cómo contactarle por correo electrónico u ordinario.

Si el programa es interactivo, haga que muestre un breve aviso como el siguiente cuando se inicie en modo interactivo:

<programa> Copyright (C) <año> <nombre del autor>  
Este programa se ofrece SIN GARANTÍA ALGUNA; escriba `show w' para  
consultar los detalles. Este programa es software libre, y usted puede  
redistribuirlo bajo ciertas condiciones; escriba `show c' para más  
información.

Los hipotéticos comandos `show w` y `show w` deberán mostrar las partes correspondientes de la Licencia General Pública. Por supuesto, los comandos en su programa pueden ser diferentes; para una interfaz gráfica de usuario, puede usar un mensaje del tipo "Acerca de".

También debería conseguir que su empresa (si trabaja como programador) o escuela, en su caso, firme una "renuncia de derechos de autor ("copyright")" sobre el programa, si fuese necesario. Para más información a este respecto, y saber cómo aplicar y cumplir la licencia GNU GPL, consulte <http://www.gnu.org/licenses/>.

La Licencia General Pública de GNU no permite incorporar sus programas como parte de programas propietarios. Si su programa es una subrutina en una biblioteca, podría considerar mucho más útil permitir el enlace de aplicaciones propietarias con la biblioteca. Si esto es lo que quiere hacer, utilice la GNU Lesser General Public License en vez de esta Licencia. Pero primero, por favor consulte

<http://www.gnu.org/philosophy/why-not-lgpl.html>.

# B

## ESTRUCTURA DE TABLAS EN LA BASE DE DATOS

Tabla B.1 Tablas y campos en la base de datos

Sections					
Campo	Tipo	Llave primaria	Único	No NULL	Autoincremento
<code>id</code>	int	X	X	X	X
<code>section_id</code>	int	-	-	-	-
<code>name</code>	varchar(255)	-	-	X	-
<code>description</code>	text	-	-	-	-
<code>enabled</code>	tinyint	-	-	-	-

Models					
Campo	Tipo	Llave primaria	Único	No NULL	Autoincremento
<code>id</code>	int	X	X	X	X
<code>section_id</code>	int	-	-	X	-
<code>name</code>	varchar(255)	-	-	X	-
<code>title</code>	varchar(255)	-	-	-	-
<code>description</code>	text	-	-	-	-
<code>bibliography</code>	text	-	-	-	-
<code>exename</code>	varchar(255)	-	-	-	-
<code>enabled</code>	tinyint	-	-	-	-

<b>Variables</b>					
Campo	Tipo	Llave primaria	Único	No NULL	Autoincremento
<code>id</code>	int	X	X	X	X
<code>model_id</code>	int	-	-	X	-
<code>modelicaname</code>	varchar(255)	-	-	X	-
<code>alias</code>	varchar(255)	-	-	-	-
<code>description</code>	text	-	-	-	-
<code>units</code>	varchar(255)	-	-	-	-
<code>type</code>	varchar(255)	-	-	-	-
<code>value</code>	varchar(255)	-	-	-	-

<b>Parameters</b>					
Campo	Tipo	Llave primaria	Único	No NULL	Autoincremento
<code>id</code>	int	X	X	X	X
<code>model_id</code>	int	-	-	X	-
<code>modelicaname</code>	varchar(255)	-	-	X	-
<code>alias</code>	varchar(255)	-	-	-	-
<code>description</code>	text	-	-	-	-
<code>units</code>	varchar(255)	-	-	-	-
<code>type</code>	varchar(255)	-	-	-	-
<code>value</code>	varchar(255)	-	-	-	-

<b>Modellers</b>					
Campo	Tipo	Llave primaria	Único	No NULL	Autoincremento
<code>id</code>	int	X	X	X	X
<code>firstname</code>	varchar(255)	-	-	X	-
<code>lastname</code>	varchar(255)	-	-	X	-
<code>email</code>	varchar(255)	-	-	-	-

<b>Modellers_models</b>					
Campo	Tipo	Llave primaria	Único	No NULL	Autoincremento
<code>modeller_id</code>	int	-	-	X	-
<code>model_id</code>	int	-	-	X	-

<b>Controls</b>					
Campo	Tipo	Llave primaria	Único	No NULL	Autoincremento
<code>id</code>	int	X	X	X	X
<code>control_group_id</code>	int	-	-	X	-
<code>name</code>	varchar(255)	-	-	-	-
<code>parameter_id</code>	int	-	-	X	-
<code>description</code>	text	-	-	-	-
<code>allowedvalues</code>	text	-	-	-	-

value	varchar(255)	-	-	-	-
min	float	-	-	-	-
max	float	-	-	-	-
step	float	-	-	-	-
enabled	tinyint	-	-	-	-

<b>Control_groups</b>					
Campo	Tipo	Llave primaria	Único	No NULL	Autoincremento
id	int	X	X	X	X
model_id	int	-	-	X	-
name	varchar(255)	-	-	X	-
description	text	-	-	-	-
enabled	tinyint	-	-	-	-

<b>Plots</b>					
Campo	Tipo	Llave primaria	Único	No NULL	Autoincremento
id	int	X	X	X	X
model_id	int	-	-	X	-
name	varchar(255)	-	-	X	-
description	text	-	-	-	-
variable_id	int	-	-	-	-
minX	float	-	-	-	-
maxX	float	-	-	-	-
gridX	tinyint	-	-	-	-
autoscaleX	char(1)	-	-	-	-
minY	float	-	-	-	-
maxY	float	-	-	-	-
gridY	tinyint	-	-	-	-
autoscaleY	char(1)	-	-	-	-
firstdata	int	-	-	-	-
enabled	tinyint	-	-	-	-

<b>Curves</b>					
Campo	Tipo	Llave primaria	Único	No NULL	Autoincremento
id	int	X	X	X	X
plot_id	int	-	-	X	-
name	varchar(255)	-	-	X	-
legend	varchar(255)	-	-	X	-
description	text	-	-	-	-
variable_id	int	-	-	X	-
colorRGB	varchar(255)	-	-	-	-
enabled	tinyint	-	-	-	-

<b>2danimations</b>					
Campo	Tipo	Llave primaria	Único	No NULL	Autoincremento
<b>id</b>	int	X	X	X	X
<b>name</b>	varchar(255)	-	-	X	-
<b>description</b>	text	-	-	-	-
<b>model_id</b>	int	-	-	X	-
<b>svg_file</b>	varchar(255)	-	-	-	-
<b>duration</b>	float	-	-	-	-
<b>sample_time</b>	float	-	-	-	-
<b>enabled</b>	tinyint	-	-	-	-

<b>2deffects</b>					
Campo	Tipo	Llave primaria	Único	No NULL	Autoincremento
<b>id</b>	int	X	X	X	X
<b>name</b>	varchar(255)	-	-	-	-
<b>description</b>	text	-	-	-	-
<b>2danimation_id</b>	int	-	-	-	-
<b>svganimation_id</b>	varchar(255)	-	-	-	-
<b>variable_id</b>	int	-	-	-	-
<b>variable_aux_id</b>	int	-	-	-	-
<b>sequence</b>	int	-	-	-	-
<b>type</b>	varchar(255)	-	-	-	-
<b>offset</b>	decimal	-	-	-	-
<b>scale</b>	decimal	-	-	-	-
<b>colorRGBmin</b>	varchar(255)	-	-	-	-
<b>colorRGBmax</b>	varchar(255)	-	-	-	-
<b>colorMin</b>	decimal	-	-	-	-
<b>colorMax</b>	decimal	-	-	-	-
<b>enabled</b>	tinyint	-	-	-	-

<b>Practices</b>					
Campo	Tipo	Llave primaria	Único	No NULL	Autoincremento
<b>id</b>	int	X	X	X	X
<b>model_id</b>	int	-	-	-	-
<b>name</b>	text	-	-	-	-
<b>header</b>	text	-	-	-	-
<b>description</b>	text	-	-	-	-
<b>enabled</b>	tinyint	-	-	-	-

<b>Users</b>					
Campo	Tipo	Llave primaria	Único	No NULL	Autoincremento
<b>id</b>	int	X	X	X	X
<b>user_name</b>	varchar(255)	-	-	X	-
<b>password</b>	varchar(255)	-	-	X	-



# C

## VALORES DE LAS PROPIEDADES DE LAS CLASES DE ADMINISTRACIÓN

En la sección 6.5.2 se explica el propósito de los principales miembros de la clase `adminblock`. El valor que toman esos miembros en cada una de las clases hijas es diferente. En este apéndice se muestran esos valores de la siguiente forma:

- La tabla C.1 muestra el contenido de la variable `table`.
- La tabla C.2 muestra el contenido del arreglo `fields`.
- La tabla C.3 muestra el contenido del arreglo `Relations1N`.
- La tabla C.4 muestra el contenido del arreglo `Relations1N1N`.
- La tabla C.5 muestra el contenido del arreglo `Relations1NN1`.

Tabla C.1 Contenido de la propiedad table para las clases de administración

Clase	Table
adminmodel	models
adminplot	plots
admin2danim	2danimations
admin2deffect	2deffects
admincontrol	control_groups
adminpractice	practices
adminsection	sections

Tabla C.2 Contenido del arreglo Fields para las clases de administración

Arreglo Fields de la clase adminmodel		
#	Clave	Valor
0	dbname	id
	showname	Identificador
	type	fixed
1	dbname	exename
	showname	Nombre del archivo ejecutable
	type	fixed
2	dbname	name
	showname	Nombre del modelo
	type	text
3	dbname	title
	showname	Título del modelo
	type	text
4	dbname	description
	showname	Descripción
	type	longtext
5	dbname	enabled
	showname	Habilitado
	type	bool
6	dbname	section_id
	showname	Sección padre
	type	select section
Arreglo Fields de la clase adminplot		
#	Clave	Valor
0	dbname	id
	showname	Identificador
	type	fixed

1	dbname	model_id
	showname	Identificador del modelo
	type	model link
2	dbname	name
	showname	Nombre de la figura
	type	text
3	dbname	description
	showname	Descripción
	type	longtext
4	dbname	enabled
	showname	Habilitado
	type	bool
5	dbname	variable_id
	showname	Variable horizontal
	model_id	0
	type	select variable
	dbname	minX
	showname	Valor mínimo de la variable horizontal
6	type	float
	dbname	maxX
	showname	Valor máximo de la variable horizontal
	type	float
7	dbname	gridX
	showname	Divisiones horizontales
	type	int
8	dbname	autoscaleX
	showname	Auto-escala horizontal
	value	0
	type	bool
9	dbname	minY
	showname	Valor mínimo de la variable vertical
	type	float
10	dbname	maxY
	showname	Valor máximo de la variable vertical
	type	float
11	dbname	gridY
	showname	Divisiones verticales
	type	int
12	dbname	autoscaleY
	showname	Auto-escala vertical
13	showname	

	value	0
	type	bool
14	dbname	firstdata
	showname	Primer dato
	type	int
	<b>Arreglo Fields de la clase admin2danim</b>	
	#	<b>Clave</b>
0	dbname	id
	showname	Identificador
	type	fixed
1	dbname	model_id
	showname	Identificador del modelo
	type	model link
2	dbname	name
	showname	Nombre de la animación
	type	text
3	dbname	description
	showname	Descripción
	type	longtext
4	dbname	enabled
	showname	Habilitado
	type	bool
5	dbname	duration
	showname	Duración
	type	float
6	dbname	sample_time
	showname	Tiempo de muestreo
	type	float
<b>Arreglo Fields de la clase admin2deffect</b>		
#	<b>Clave</b>	<b>Valor</b>
0	dbname	id
	showname	Identificador
	type	fixed
1	dbname	2danimation_id
	showname	Identificador de animación
	type	fixed
2	dbname	name
	showname	Nombre
	type	text

	dbname	description
3	showname	Descripción
	type	longtext
	dbname	enabled
	showname	Habilitado
	type	bool
	dbname	type
4	showname	Tipo
	options	Array
	table	2defects
	type	select options
5	dbname	svganimation_id
	showname	Identificador de animación SVG
	type	text
6	dbname	variable_id
	showname	Variable de Control
	model_id	0
	type	select variable
7	dbname	variable_aux_id
	showname	Variable Auxiliar
	model_id	0
	type	select variable
8	dbname	offset
	showname	Offset
	type	float
9	dbname	scale
	showname	Escala
	type	float
10	dbname	colorRGBmin
	showname	Color RGB mínimo
	type	color
11	dbname	colorRGBmax
	showname	Color RGB máximo
	type	color
12	dbname	colorMin
	showname	Valor mínimo de Color
	type	float
13	dbname	colorMax
	showname	Valor máximo de Color

	<b>type</b>	float
<b>Arreglo Fields de la clase admincontrol</b>		
#	<b>Clave</b>	<b>Valor</b>
0	dbname	id
	showname	Identificador
	type	fixed
1	dbname	model_id
	showname	Valor
	type	model link
2	dbname	name
	showname	Nombre
	type	text
3	dbname	description
	showname	Descripción
	type	longtext
4	dbname	enabled
	showname	Habilitado
	type	bool
<b>Arreglo Fields de la clase adminpractice</b>		
#	<b>Clave</b>	<b>Valor</b>
0	dbname	id
	showname	Identificador
	type	fixed
1	dbname	name
	showname	Nombre de la práctica
	type	text
2	dbname	header
	showname	Encabezado
	type	longtext
3	dbname	description
	showname	Descripción
	type	longtext
4	dbname	enabled
	showname	Habilitado
	type	bool
<b>Arreglo Fields de la clase adminsection</b>		
#	<b>Clave</b>	<b>Valor</b>
	dbname	id

0	showname	Identificador
	type	fixed
1	dbname	name
	showname	Nombre de la Sección
	type	text
2	dbname	description
	showname	Descripción
	type	longtext
3	dbname	enabled
	showname	Habilitado
	type	bool
4	dbname	section_id
	showname	Sección padre
	type	select section

Tabla C.3 Contenido del arreglo Relations1N para las clases de administración

Arreglo Relations1N de la clase adminmodel		
#	Clave	Valor
0	title	Animaciones en 2D
	table	2danimations
	linkname	2danimationid
	id1	id
	id2	model_id
	idvalue	0
	showdbname	name
	show	Nombre de la animación
1	title	Grupos de controles
	table	control_groups
	linkname	controlgroupid
	id1	id
	id2	model_id
	idvalue	0
	showdbname	name
	show	Nombre del grupo de controles
2	title	Prácticas
	table	practices
	linkname	practiceid
	id1	id
	id2	model_id
	idvalue	0
	showdbname	name
	show	Nombre de las prácticas

<b>Arreglo Relations1N de la clase adminplot</b>		
#	Clave	Valor
El arreglo está vacío		
<b>Arreglo Relations1N de la clase admin2danim</b>		
#	Clave	Valor
0	title	Animación
	table	2deffects
	linkname	2effectid
	id1	id
	id2	2animation_id
	idvalue	0
	showdbname	name
	show	Nombre
<b>Arreglo Relations1N de la clase admin2deffect</b>		
#	Clave	Valor
El arreglo está vacío		
<b>Arreglo Relations1N de la clase admincontrol</b>		
#	Clave	Valor
El arreglo está vacío		
<b>Arreglo Relations1N de la clase adminpractice</b>		
#	Clave	Valor
El arreglo está vacío		
<b>Arreglo Relations1N de la clase admininsection</b>		
#	Clave	Valor
0	title	Secciones hijas
	table	sections
	linkname	sectionid
	id1	id
	id2	section_id
	idvalue	0
	showdbname	name
	show	Nombre de la Sección
1	title	Modelos
	table	models
	linkname	modelid
	id1	id
	id2	section_id
	idvalue	0
	showdbname	name
	show	Nombre del Modelo

Tabla C.4 Contenido del arreglo Relations1N1N para las clases de administración

<b>Arreglo Relations1N1N de la clase adminmodel</b>				
#	Clave	Valor	Clave	Valor
El arreglo está vacío				
<b>Arreglo Relations1N1N de la clase adminplot</b>				
#	Clave	Valor	Clave	Valor
	title	Curvas		
	table	curves		
	linkname	curveid		
	id1	id		
	id2	plot_id		
	idvalue	0		
	showdbname	name		
	show	Nombre de la curva		
0	subfields	0	dbname	id
			showname	Identificador
			type	fixed
		1	dbname	name
			showname	Nombre de la curva
			type	text
		2	dbname	legend
			showname	Leyenda
			type	text
		3	dbname	enabled
			showname	Habilitado
			type	bool
		4	dbname	description
			showname	Descripción
			type	longtext
		variable_id	dbname	variable_id
			showname	Variable vertical
			model_id	0
			type	select variable
		5	dbname	colorRGB
			showname	Color
			type	color

<b>Arreglo Relations1N1N de la clase admin2danim</b>				
#	Clave	Valor	Clave	Valor
El arreglo está vacío				
<b>Arreglo Relations1N1N de la clase admin2deffect</b>				
#	Clave	Valor	Clave	Valor
El arreglo está vacío				
<b>Arreglo Relations1N1N de la clase adminincontrol</b>				
#	Clave	Valor	Clave	Valor
	title	Controles		
	table	controls		
	linkname	controlid		
	id1	id		
	id2	control_group_id		
	idvalue	0		
	showdbname	name		
	show	Nombre		
0	subfields	0	dbname	id
			showname	Identificador
			type	fixed
		1	dbname	name
			showname	Nombre
			type	text
		2	dbname	value
			showname	Valor
			type	text
		3	dbname	enabled
			showname	Habilitado
			type	bool
		4	dbname	min
			showname	Mínimo
			type	text
		5	dbname	max
			showname	Máximo
			type	text
		6	dbname	step
			showname	Incremento

0	subfields	7	type	text
			dbname	allowedvalues
			showname	Valores permitidos
			type	longtext
		8	dbname	description
			showname	Descripción
			type	longtext
		parameter_id	dbname	parameter_id
			showname	Parámetro
			model_id	0
			type	select parameter
<b>Arreglo Relations1N1N de la clase adminpractice</b>				
#	Clave	Valor	Clave	Valor
El arreglo está vacío				
<b>Arreglo Relations1N1N de la clase adminsection</b>				
#	Clave	Valor	Clave	Valor
El arreglo está vacío				

Tabla C.5 Contenido del arreglo Relations1NN1 para las clases de administración

<b>Arreglo Relations1N1N de la clase adminmodel</b>				
#	Clave	Valor	Clave	Valor
	title	Modeladores		
	tableLink	modellers_models		
	table2	modellers		
	linkname	modellerid		
	id1	id		
	id2	id		
	idLink1	model_id		
	idLink2	modeller_id		
	idvalue	0		
	showdbname	concat(lastname,firstname)		
	show	Selección		
0	order	0	lastname	
		1	firstname	

subfields	0	dbname	id	
		showname	Identificador	
		type	fixed	
	1	dbname	lastname	
		showname	Apellidos	
		type	text	
	2	dbname	firstname	
		showname	Nombres	
		type	text	
	3	dbname	email	
		showname	Correo	
		type	text	
<b>Arreglo Relations1N1N de la clase adminplot</b>				
#	Clave	Valor	Clave	Valor
El arreglo está vacío				
<b>Arreglo Relations1N1N de la clase admin2danim</b>				
#	Clave	Valor	Clave	Valor
El arreglo está vacío				
<b>Arreglo Relations1N1N de la clase admin2deffect</b>				
#	Clave	Valor	Clave	Valor
El arreglo está vacío				
<b>Arreglo Relations1N1N de la clase admincontrol</b>				
#	Clave	Valor	Clave	Valor
El arreglo está vacío				
<b>Arreglo Relations1N1N de la clase adminpractice</b>				
#	Clave	Valor	Clave	Valor
El arreglo está vacío				
<b>Arreglo Relations1N1N de la clase adminsection</b>				
#	Clave	Valor	Clave	Valor
El arreglo está vacío				

# D

## PARÁMETROS DEL TEMA UNVLBLASIC

El tema unvblbasic se define en los archivos `style.css` y `styleSVG.css`. El primero de esos archivos contiene las definiciones generales de toda la apariencia de la interfaz, mientras que el segundo se ha dedicado a las definiciones asociadas con las figuras que muestran los resultados de las simulaciones.

En este apéndice se documentan las especificaciones CSS de la siguiente forma:

- La tabla D.2 muestra qué parámetros se han definido en cada uno de las clases CSS en el archivo `style.css`.
- La tabla D.1 muestra el valor asignado a cada uno de los parámetros al interior de cada clase CSS, en el archivo `style.css`.
- La tabla D.4 muestra qué parámetros se han definido en cada uno de las clases CSS en el archivo `styleSVG.css`.
- La tabla D.3 muestra el valor asignado a cada uno de los parámetros al interior de cada clase CSS, en el archivo `styleSVG.css`.

Tabla D.1 Clases y parámetros CSS personalizados en el theme unvbasic en el archivo style.css

Clase	width	vertical-align	text-align	stroke	position	right	padding	overflow-x	overflow-y	outline	margin-top	margin-left	margin-bottom	margin	overflow	overflow-x	overflow-y	outline	stroke-width	text-align	width
a.logo	x	x			x	x				x									x	x	
a.section_name		x			x					x									x	x	
body.unvl	x				x	x				x	x							x	x	x	
body.unvl_admin	x				x	x				x	x							x	x	x	
div.admin_general	x	x			x	x				x	x							x	x	x	
div.admin_logout	x	x			x	x				x	x							x	x	x	
div.admin_modfiles	x	x			x	x				x	x							x	x	x	
div.anim2d	x	x			x	x				x	x							x	x	x	
div.animation	x	x			x	x				x	x							x	x	x	
div.animationTitle					x	x				x	x							x	x	x	
div.body	x	x			x	x				x	x							x	x	x	
div.body_admin	x	x			x	x				x	x							x	x	x	
div.controls	x	x			x	x				x	x							x	x	x	
div.controlsEmbedded	x	x			x	x				x	x							x	x	x	
div.controls_buttons																		x	x	x	
div.controls_change											x							x	x	x	
div.controls_change_minus											x	x						x	x	x	
div.controls_change_plus											x	x						x	x	x	
div.credits	x										x							x	x	x	
div.documentation	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	

Clase																													
div.documentationEmbed																													
div.downdata																													
div.logo	x																												
div.menu	x																												
div.messageonmodel	x						x	x																					
div.outputs		x																											
div.outputsEmbed		x																											
div.ploys	x		x																										
div.section_name		x																											
div.tables	x	x																	x	x	x	x	x	x	x	x	x	x	
div.title	x																	x	x	x	x	x	x	x	x	x	x	x	
div.title_frame	x	x															x					x	x	x	x	x	x	x	
div.wellcone											x																		
div.wellcome_box	x	x	x						x								x	x	x	x	x	x	x	x	x	x	x	x	
div.wellcome_buttons	x		x						x								x	x	x	x	x	x	x	x	x	x	x	x	
div.wellcome_explain												x																	
div.wellcome_sub			x																										
input.checkUpdate				x																									
input.color				x																									
input.controls			x																x	x	x	x	x	x	x	x	x	x	
input.controls_button	x		x																										
input.controls_button_file			x																x	x	x	x	x	x	x	x	x	x	

Clase	background	background-color	background-image	border	color	display	float	font-family	font-size	font-weight	height	left	margin	margin-bottom	margin-left	margin-right	margin-top	overflow	overflow-x	overflow-y	padding	position	right	stroke	text-align	top	vertical-align	width
input.downdata																												
input.login																												
input.logout_button	x			x	x				x	x		x						x	x	x	x	x	x	x	x	x	x	
input.mediumUpdate																												
input.miniUpdate																												
input.normalUpdate																												
input.welcome_button	x			x	x				x	x		x						x	x	x	x	x	x	x	x	x	x	
object.documentation																												
rect.control_back											x																	
rect.control_bar											x																	
select.controls					x						x							x	x	x	x	x	x	x	x	x	x	
select.miniUpdate																												
select.normalUpdate																												
svg.controls																		x										
table.about					x						x							x	x	x	x	x	x	x	x	x	x	
table.controls						x	x				x							x	x	x	x	x	x	x	x	x	x	
table.outputs						x	x				x							x	x	x	x	x	x	x	x	x	x	
table.update							x				x							x	x	x	x	x	x	x	x	x	x	
table.updatefile						x	x				x							x	x	x	x	x	x	x	x	x	x	
td.about_name	x	x	x	x	x													x	x	x	x	x	x	x	x	x	x	
td.about_value	x	x	x	x	x													x	x	x	x	x	x	x	x	x	x	
td.about_video	x	x	x	x	x													x	x	x	x	x	x	x	x	x	x	

	background		x																	
	background-color		x																	
	background-image		x																	
	background-position		x																	
	background-repeat		x																	
	background-size		x																	
	border		x																	
	border-collapse		x																	
	border-color		x																	
	border-radius		x																	
	border-style		x																	
	border-width		x																	
	color		x																	
	display		x																	
	float		x																	
	font-family		x																	
	font-size		x																	
	font-weight		x																	
	height		x																	
	left		x																	
	margin		x																	
	margin-bottom		x																	
	margin-left		x																	
	margin-right		x																	
	margin-top		x																	
	overflow-x		x																	
	overflow-y		x																	
	padding		x																	
	position		x																	
	right		x																	
	stroke		x																	
	text-align		x																	
	vertical-align		x																	
	top		x																	
td.controls																				
td.login_button																				
td.login_data																				
td.login_label																				
td.outputs																				
td.outputsTitle																				
td.update_name																				
td.update_value																				
td.updatefile_button																				
td.updatefile_file																				
td.updatefile_name																				
textarea.normalUpdate																x				
th.about_title													x							
th.controls													x							
th.update_title												x	x							
tr.outputsHover												x								

Tabla D.2 Valores de los parámetros CSS en el theme `unvlbasic` en el archivo `style.css`

Tipo	Clase	Parámetro	Valor
a	logo	background	url('img/logo.svg')
		background-color	inherit
		height	30px
		left	0px
		position	absolute
		top	0px
		width	209px
	section_name	color	#000000
		left	0px
		position	relative
body	unvl	background-color	#f0f0f0
		font-family	Arial, Helvetica, sans-serif
		font-size	.8em
		height	2000px
		overflow-x	scroll
		overflow-y	scroll
		text-align	left
	unvl_admin	background-color	#ff0000
		font-family	Arial, Helvetica, sans-serif
		font-size	.8em
		height	2000px
		overflow-x	scroll
		overflow-y	scroll
		text-align	left
div	admin_general	background-color	#e0e0e0
		border	1px solid #000000
		position	absolute
		right	10px
		top	10px
		width	981px
	admin_logout	background-color	transparent
		border	0px solid #000000
		height	25px
		left	10px
		position	absolute
		text-align	center
		top	320px
		width	209px

Tipo	Clase	Parámetro	Valor
div	admin_modelfiles	background-color	transparent
		border	0px solid #000000
		left	10px
		position	absolute
		top	350px
		width	209px
	anim2d	background-color	inherit
		border	solid 0px #000000
		margin-right	auto
		position	relative
		right	10px
		top	0px
	animation	background-color	inherit
		border	solid 1px #000000
		width	300px
	animationTitle	float	center
		font-weight	bold
		position	relative
		text-align	center
	body	background-color	#c0c0ff
		border	1px solid #000000
		color	inherit
		float	center
		height	100 %
		margin-left	auto
		margin-right	auto
		padding	10px
		position	relative
		width	900px
	body_admin	background-color	#ff7777
		border	1px solid #000000
		color	inherit
		float	center
		height	100 %
		margin-left	auto
		margin-right	auto
		padding	10px
		position	relative
		width	1200px
	controls	background-color	#aaaaaa

Tipo	Clase	Parámetro	Valor
div	controls	border	solid 1px #000000
		left	10px
		margin-left	auto
		margin-right	auto
		position	absolute
		top	320px
		width	209px
	controlsEmbed	background-color	#aaaaaa
		border	solid 1px #000000
		left	10px
		margin-left	auto
		margin-right	auto
		position	absolute
		top	50px
	controls_buttons	width	209px
		text-align	center
div	controls_change	height	30px
		position	relative
		top	0px
		width	90px
	controls_change_minus	background-image	url('img/menos.gif')
		height	10px
		left	0px
		position	absolute
		top	12px
		width	20px
	controls_change_plus	background-image	url('img/mas.gif')
		height	10px
		left	0px
		position	absolute
		top	2px
		width	20px
div	credits	background	url('img/info.gif')
		height	30px
		position	absolute
		right	0px
		width	30px
	documentation	background-color	#ffffff
		border	1px solid #000000
		color	inherit

Tipo	Clase	Parámetro	Valor
div		display	table
		height	260px
		padding	0px
		position	absolute
		right	10px
		top	50px
		width	680px
	documentationEmbed	background-color	#ffffff
		border	1px solid #000000
		color	inherit
		height	260px
		padding	0px
		position	absolute
		right	10px
		top	50px
	downdata	width	900px
		margin-top	10px
		text-align	center
	logo	background-color	inherit
		height	30px
		left	10px
		position	absolute
		top	10px
		width	209px
	menu	background-color	#ffffff
		border	solid 1px #000000
		font-size	1.1em
		height	260px
		left	10px
		overflow-x	scroll
		overflow-y	scroll
		position	absolute
		top	50px
		width	209px
	messagenomodel	background-color	#ffffff
		display	table-cell
		font-family	Arial, Helvetica, sans-serif
		font-size	2em
		text-align	center
		vertical-align	middle

Tipo	Clase	Parámetro	Valor
div	outputs	color	inherit
		height	220px
		margin	0px
		padding	0px
		position	absolute
		right	10px
		top	320px
		width	682px
	outputsEmbed	color	inherit
		height	220px
		margin	0px
		padding	0px
		position	absolute
		right	10px
		top	50px
		width	682px
div	plots	background-color	inherit
		border	solid 0px #000000
		left	10px
		margin-left	10px
		margin-right	auto
		position	relative
		top	0px
	section_name	color	#000000
		left	10px
		position	relative
div	tables	background-color	inherit
		border	solid 0px #000000
		height	220px
		left	10px
		margin-left	10px
		margin-right	auto
		overflow	auto
		position	relative
		text-align	center
		top	0px
	title	width	302px
		background-color	inherit
		font-size	2em
		height	30px

Tipo	Clase	Parámetro	Valor
div		left	0px
		overflow-x	auto
		overflow-y	auto
		position	absolute
		text-align	center
		width	651px
	title_frame	background-color	#e0e0e0
		border	1px solid #000000
		height	30px
		position	absolute
		right	10px
		top	10px
	wellcome	width	681px
		font-size	3em
	wellcome_box	background-color	#ffff99
		border	solid 4px #000000
		color	#0000FF
		float	center
		margin-left	auto
		margin-right	auto
		padding	20px
		position	relative
		text-align	center
		top	150px
		width	350px
		background-color	#inherit
input	wellcome_buttons	color	#000000
		float	center
		margin-left	auto
		margin-right	auto
		padding	10px
		position	relative
		text-align	center
		top	150px
		width	350px
	wellcome_explain	font-size	0.8em
	wellcome_sub	font-size	2em
	checkUpdate	width	40px
	color	width	70px
	controls	height	20px

Tipo	Clase	Parámetro	Valor
input	controls_button	width	100 %
		background-color	#dddddd
		border	solid 1px #000000
		font-size	0.8em
		height	25px
		padding	0px
		width	70px
	controls_button_file	text-align	center
		width	50px
	downdata	background-color	#dddddd
		border	solid 1px #000000
		font-size	0.8em
		height	25px
		padding	0px
		width	100px
object	login logout_button		
		background-color	#dddddd
		border	solid 1px #000000
		color	#000000
		float	center
		font-size	1.0em
		height	25px
		margin-left	auto
		margin-right	auto
		position	relative
		width	150px
	mediumUpdate	width	100px
	miniUpdate	width	50px
	normalUpdate	width	300px
	wellcome_button	background-color	#dddddd
		border	solid 1px #000000
		color	#000000
		float	center
		font-size	1.0em
		height	25px
		margin-left	auto
		margin-right	auto
		position	relative
		width	150px
object	documentation	height	100 %

Tipo	Clase	Parámetro	Valor
		width	100 %
rect	control_back	fill	none
		stroke	none
	control_bar	fill	#ff0000
		stroke	none
select	controls	background-color	#ffffff
		width	100 %
	miniUpdate	text-align	center
		width	100px
	normalUpdate	width	300px
svg	controls	height	8px
		width	100 %
table	about	border-collapse	collapse
		font-size	.8em
		text-align	left
	controls	color	#0000ff
		font-size	1.0em
	outputs	border	solid 1px #000000
		border-collapse	collapse
		color	#000000
		float	center
		font-size	0.8em
		margin-bottom	10px
		margin-left	auto
		margin-right	auto
	update	margin-top	10px
		text-align	left
		border-collapse	collapse
		font-size	.8em
		margin-left	auto
		margin-right	auto
	updatefile	margin-top	10px
		text-align	left
		background-color	#FFEEEE
		border	solid 1px #000000
		border-collapse	collapse
		font-size	.8em
		margin-left	auto

Tipo	Clase	Parámetro	Valor
		width	200px
td	about_name	background-color	#FFCCCC
		border	solid 1px #000000
		color	#000000
		width	150px
	about_value	background-color	#FFEEEE
		border	solid 1px #000000
		color	#3333FF
		width	300px
	about_video	background-color	#FFEEEE
		border	solid 1px #000000
		color	#3333FF
		text-align	center
		width	60px
	controls	color	#000000
		text-align	left
		width	100px
	login_button	text-align	center
	login_data	text-align	left
	login_label	text-align	right
	outputs	border	solid 1px #000000
		text-align	center
		width	40px
	outputsTitle	border	solid 1px #000000
		font-weight	bold
		text-align	center
		width	40px
	update_name	background-color	#FFCCCC
		border	solid 1px #000000
		color	#000000
		width	150px
	update_value	background-color	#FFEEEE
		border	solid 1px #000000
		color	#3333FF
		width	300px
	updatefile_button	width	55px
	updatefile_file	width	30px
	updatefile_name	width	120px
textarea	normalUpdate	height	100px
		width	300px

Tipo	Clase	Parámetro	Valor
th	about_title	background-color	#FFCCCC
		border	solid 1px #000000
		color	#000000
		font-size	1.0em
		text-align	center
	controls	color	#0000ff
		text-align	left
		width	200px
	update_title	background-color	#FFCCCC
		border	solid 1px #000000
		color	#000000
		font-size	1.0em
		text-align	center
tr	outputs:hover	background-color	#88DDDD

Tabla D.3 Clases y parámetros CSS personalizados en el theme `unvlbasic` en el archivo `styleSVG.css`

Clase	background-color	dominant-baseline	fill	font-family	font-size	font-style	font-weight	height	stroke	stroke-dasharray	stroke-width	width
line.grid									x	x	x	
path.curve			x								x	
rect.frame			x						x		x	
rect.plot			x						x		x	
rect.tag			x						x		x	
svg.plot	x			x		x	x	x				x
text.legend		x			x							
text.tagH		x	x		x							
text.tagV		x	x		x							
text.title			x	x								

Tabla D.4 Valores de los parámetros CSS en el theme unvlabasic en el archivo styleSVG.css

Tipo	Clase	Parámetro	Valor
line	grid	stroke	#333333
		stroke-dasharray	3 3
		stroke-width	1px
path	curve	fill	none
		stroke-width	1px
rect	frame	fill	#eeeeee
		stroke	#000000
		stroke-width	2px
	plot	fill	#ffffff
		stroke	#000000
		stroke-width	2px
	tag	fill	#ffffff
		stroke	none
		stroke-width	1px
svg	plot	background-color	inherit
		font-family	arial, times, serif
		font-style	normal
		font-weight	normal
		height	200px
		width	300px
text	legend	dominant-baseline	middle
		font-size	10pt
	tagH	dominant-baseline	auto
		fill	#888888
		font-size	10pt
	tagV	dominant-baseline	middle
		fill	#888888
		font-size	10pt
	title	fill	#555555
		font-size	14pt

# E

## VERSIÓN 1.0

Como parte del proceso de desarrollo natural de UNVirtualLab se implementó una versión previa (Versión 1.0, presentada en [Dua11]) que sirvió como prueba conceptual de la herramienta. La versión actual, la 2.0, conserva muchos de los elementos de esa primera versión, pero se diferencia en algunos aspectos importantes de diseño:

- La versión 1.0 no utilizaba ninguna base de datos. En su lugar, la información necesaria se almacenaba en archivos de texto en formato XML.
- Los gráficos de la versión 1.0 se preparaban en el lado del servidor en formato png. La versión 2.0 genera los gráficos en formato svg que son construidos por el navegador (lado del cliente).
- Las animaciones de la versión 2.0 también se preparaban en formato png animado. La versión 2.0 lo hace en formato svg.
- La versión 1.0 incluía la opción de construir animaciones 3D a partir de primitivas gráficas. Esta funcionalidad se ha omitido en la versión 2.0, a la espera de un mayor desarrollo de las opciones 3D del formato svg.
- La versión 1.0 utilizaba los archivos de resultados de los ejecutables en formato plt (el único disponible en ese entonces). La versión 2.0 lo hace en formato csv.

Los efectos más notorios de estos cambios han sido:

- La modularidad del código fuente se ha incrementado.
- La mantenibilidad del código fuente se ha mejorado.
- Los tiempos de respuesta del servidor se han disminuido.
- La uniformidad de las gráficas se ha mejorado.
- La personalización de las implementaciones se ha incrementado.

## BIBLIOGRAFÍA

- [AAM<sup>+</sup>22] Gonzalo Abella, Javier Aguirre, Héctor J. Macho, Borja Menéndez, Javier Moset, and Ángel Torrado. Traducción de la licencia gplv3 al español. [link:22], Visitada en 2013-08-22.
- [AKB<sup>+</sup>01] H. Afsarmanesh, E. C. Kaletas, A. Benabdeler, C. Gariota, and L.O. Hertzberger. A reference architecture for scientific virtual laboratories. *Future Generation Computer Systems*, 17:999–1008, 2001.
- [AN09] Mahmoud Abdulwahed and Zoltan K Nagy. Applying kolb's experiential learning cycle for laboratory education. *Journal of Engineering Education*, 98:283–294, 2009.
- [AN11] Mahmoud Abdulwahed and Zoltan K. Nagy. The trilab, a novel ict based triple access mode laboratory education model. *Computers & Education*, 56:262–274, 2011.
- [And94] Mats Andersson. *Object-Oriented Modeling and Simulation of Hybrid Systems*. PhD thesis, Department of Automatic Control, Lund University, Sweden, 1994.
- [Ara95] Javier Aracil. *Dinámica de Sistemas*. Isdefe, 1995.
- [Asp16] Aspentech. Aspen - hysys. [link:23], Visitado en 2013-08-16.
- [Ass00] Modelica Association. Modelica tm - a unified object-oriented language for physical systems modeling. Technical report, Modelica Association, 2000.
- [Ass12] Modelica Association. *Modelica - A Unified Object-Oriented Language for Systems Modeling Language Specification. Version 3.3*. Modelica Association, 2012.

- [Ass18] Modelica Association. Modelica tools. [link:24], Visitado en 2014-07-18.
- [Ass07] Modelica Association. Modelica libraries. [link:25], Visitado en 2014-08-07.
- [Ass25a] Modelica Association. Modelica association. [link:26], Visitado en 2014-08-25.
- [Ass25b] Modelica Association. Modelica license version 2. [link:27], Visitado en 2014-08-25.
- [BBC<sup>+</sup>03] Karl-Frederik Berggren, Doris Brodeur, Edward F. Crawley, Ingemar Ingemarsson, William T.G. Litant, Johan Malmqvist, and Sören Östlund. Cdio: An international initiative for reforming engineering education. *World Transactions on Engineering and Technology Education*, 2:49–52, 2003.
- [BHLRP98] Peter N. Brown, Alan C. Hindmarsh, and L. Linda R. Petzold. Consistent initial condition calculation for differential-algebraic systems. *SIAM Journal on Scientific Computing*, 19:1495–1512, 1998.
- [BK91] Richard E. Boyatzis and David A. Kolb. Assessing individuality in learning: the learning skills profile. *Educational Psychology*, 11:279–295, 1991.
- [BOA<sup>+</sup>11] T. Blochwitz, M. Otter, M. Arnold, C. Bausch, C. Clauß, H. Elmqvist, A. Junghanns, J. Mauss, M. Monteiro, T. Neidhold, D. Neumerkel, H. Olsson, J.-V. Peetz, and S. Wolf. The functional mockup interface for tool independent exchange of simulation models. In *Proceedings of the 2011 Modelica Conference*, 2011.
- [Bor10] Wolfgang Borutzky. *Bond Graph Methodology. Development and Analysis of Multidisciplinary Dynamic System Models*. Springer, 2010.
- [BS03] Thomas G. Burke and David A. Schiller. Using pspice for electrical heat analysis. *IEEE Potentials Magazine*, 22:35–38, 2003.

- [BT99] Claus Bendtsen and Per Grove Thomsen. Numerical solution of differential algebraic equations. Technical report, Department of mathematical modelling. Technical University of Denmark, 1999.
- [Cad16] Cadence. Cadence orcad solutions. [link:28], Visitado en 2013-08-16.
- [Cam12] Guido Camargo. Modelamiento de la dinámica del dengue en colombia. Master's thesis, Universidad Nacional de Colombia. Facultad de Ingeniería, 2012.
- [Cap03] Luiz Fernando Capretz. A brief history of the object-oriented approach. *Software Engineering Notes*, 28, 2003.
- [CDK87] Leon O. Chua, Charles A. Desoer, and Ernest S. Kuh. *Linear and nonlinear circuits*. McGraw-Hill, 1987.
- [Cel91] François E. Cellier. *Continuous System Modeling*, chapter System Dynamics, pages 453–503. Springer Science & Business Media, 1991.
- [Cel08] François E. Cellier. World3 in modelica: Creating system dynamics models in the modelica framework. In *Proceedings of the 6th Modelica conference*, pages 393–400. Modelica Association, 2008.
- [Cen08] World Technology Evaluation Center. International assessment of research and development in simulation-based engineering and science. [link:29], Visitado en 2013-07-08.
- [CEOL95] François E. Cellier, Hilding Elmpqvist, Martin Otter, and William S. Levine. *The Control Handbook*, chapter Determining models, pages 99–112. CRC Press and IEEE Press, 1995.
- [Cha05] Stephen Chapman. *Máquinas eléctricas*. McGrawHill, 2005.
- [Che93] Chi-Tsong Chen. *Analog and Digital Control System Design*. Saunders College Pubkishing, 1993.
- [Che95] Chi-Tsong Chen. *Linear System Theory and Design*. Oxford Press, 1995.

- [CK06] François E. Cellier and Ernesto Kofman. *Continuous System Simulation*. Springer, 2006.
- [CLL12] Chun Yang Chong, Sai Peck Lee, and Teck Chaw Ling. Development of virtual lab system through application of fuzzy analytic hierarchy process. In *Information Science and Digital Content Technology (ICIDT), 2012 8th International Conference on*, 2012.
- [CoICME08] National Research Council Committee on Integrated Computational Materials Engineering. *Integrated Computational Materials Engineering: A Transformational Discipline for Improved Competitiveness and National Security*. The National Academies Press, 2008.
- [Con10] Modelisar Consortium. Functional mock-up interface for model exchange and co-simulation, 2010.
- [Con08a] Open Source Modelica Consortium. Openmodelica. [link:30], Visitado en 2013-07-08.
- [Con08b] Open Source Modelica Consortium. Openmodelica user documentation. [link:31], Visitado en 2013-07-08.
- [Con07] Open Source Modelica Consortium. Library testing. [link:32], Visitado en 2014-08-07.
- [Cor22] Oracle Corporation. Mysql. the world's most popular open source database. [link:33], Visitada en 2013-08-22.
- [Cra01] Edward F. Crawley. The cdio syllabus a statement of goals for undergraduate engineering education. Technical report, Department of Aeronautics and Astronautics. Massachusetts Institute of Technology, 2001.
- [Dav08] Clara Davies. Learning and teaching in laboratories. Technical report, Engineering Subject Centre. The Higher Education Academy, 2008.
- [Dep17] UC Berkeley EECS Department. The ptolemy project. [link:34], Visitado en 2014-07-17.

- [DHH02] David Duce, Ivan Herman, and Bob Hopgood. Svg tutorial. [link:35] charla en *The Eleventh International World Wide Web Conference* [link:36], 2002.
- [dI31] Comunidad de Inkscape. Inkscape. draw freely. [link:37], Visitada en 2013-12-31.
- [dllM12] Alfredo de la Lama and Aline Magaña. La distopia de la teoría del desarrollo. *Denarius Revista de Economía y Administración*, (24):15–56, 2012.
- [DMN70] Ole-Lohan Dahl, Bjorn Myhrhaug, and Kristen Nygaard. *SIMULA67 Common Base Language*. Norwegian Computing Center, 1970.
- [dPRRFH12] Juan Carlos Rodríguez del Pino, Enrique Rubio-Royo, and Zenon Hernandez-Figueroa. A virtual programming lab for moodle with automatic assessment and anti-plagiarism features. In *Proceedings of the 2012 International Conference on E-learning, E-bussines, Enterprise information systems and E-government*, 2012.
- [DS10] David A. Damassa and Toby D. Sitko. Simulation technologies in higher education: Uses, trends, and implications. Technical report, EDUCAUSE Center for Analysis and Research, 2010.
- [Dua05] Oscar Duarte. *Análisis de sistemas dinámicos lineales*. Universidad Nacional de Colombia, 2005.
- [Dua09] Oscar Duarte. La web de nueva generación en la enseñanza y el aprendizaje (de la ingeniería). Technical report, Universidad Nacional de Colombia., 2009.
- [Dua11] Oscar Duarte. Un-virtuallab : A web simulation environment of openmodelica models for educational purposes. In *8th International Modelica Conference*, 2011.
- [Dua13] Oscar Duarte. Un modelo del tránsito de los estudiantes a través del plan de estudios. Technical report, Universidad Nacional de Colombia. Facultad de Ingeniería. Vicedecanatura Académica, 2013.

- [Elm78] Hilding Elmquist. *A Structured Model Language for Large Continuous Systems*. PhD thesis, Department of Automatic Control, Lund University, Sweden, 1978.
- [Ent16] Scilab Enterprises. Scilab. [link:38], Vistada en 2013-08-16.
- [Eul68] Leonhard Euler. *Institutionum calculi integralis. VolumenPrimum*. Impensis Academiae Imperialis Scientiarum, 1768.
- [FB13] Donald G. Fink and H. Wayne Beaty. *Standard Handbook for Electrical Engineers, Sixteenth Edition*. McGraw-Hill Profesional, 2013.
- [FCDS08] Hongqing Fang, Long Chen, Nkosinathi Dlakavu, and Zuyi Shen. Basic modeling and simulation tool for analysis of hydraulic transients in hydroelectric power plants. *IEEE Transactions on energy conversion*, 23:834–841, 2008.
- [FDS<sup>+</sup>12] M. O. Faruque, V. Dinavahi, M. Steurer, A. Monti, K. Strunz, J. A. Martinez, G. W. Chang, J. Jatskevich, R. Iravani, and A. Davoudi. Interfacing issues in multi-domain simulation tools. *IEEE Transactions on power delivery*, 27:439–448, 2012.
- [FE98] Peter Fritzson and Vadim Engelson. An integrated modelica environment for modeling, documentation and simulation. In *Proceedings of the 1998 Summer Computer Simulation Conference*, 1998.
- [FE10] Mohammad Reza Farrokhnia and Ayoub Esmailpour. A study on the impact of real, virtual and comprehensive experimenting on students' conceptual understanding of dc electric circuits and their skills in undergraduate electricity laboratory. *Procedia Social and Behavioral Sciences*, 2: 5474–5482, 2010.
- [For71] J. W. Forrester. *World Dynamics*. Pegasus Communications, 1971.
- [For02] Mikael Forsgren. Generate white noise in modelica (system-modeler). [link:39], Visitado en 2014-07-02.
- [Fou05] The Apache Software Foundation. Apache http server project. [link:40], Visitado en 2013-10-05.

- [Fou18] The Eclipse Foundation. *eclipse*. [link:41], Visitado en 2014-07-18.
- [Fou22] Free Software Foundation. Gnu general public license. [link:42], Visitada en 2013-08-22.
- [FPA<sup>+</sup>13a] Peter Fritzson, Adrian Pop, Adeel Asghar, Willi Braun, Jens Frenkel, Lennart Ochel, Martin Sjölund, Per Östlund, Ole-na Rogovchenko, Peter Aronsson, Mikael Axin, Bernhard Bachmann, Vasile Baluta, Robert Braun, David Broman, Stefan Brus, Francesco Casella, Filippo Donida, Anand Gane-son, Mahder Gebremedhin, Pavel Grozman, Daniel Hedberg, Michael Hanke, Alf Isaksson, Kim Jansson, Daniel Kanth, Tommi Karhela, Juha Kortelainen, Abhinn Kothari, Petter Krus, Alexey Lebedev, Oliver Lenord, Ariel Liebman, Rickard Lindberg, Håkan Lundvall, Abhi Raj Metkar, Eric Meyers, Tuomas Miettinen, Afshin Moghadam, Maroun Nemer, Hannu Niemistö, Peter Nordin, Kristoffer Norling, Arun-kumar Palanisamy, Karl Pettersson, Pavol Privitzer, Jhansi Reddy, Reino Ruusu, Per Sahlin, Wladimir Schamai, Gerhard Schmitz, Alachew Shitahun, Anton Sodja, Ingo Staack, Kris-tian Stavåker, Sonia Tariq, Mohsen Torabzadeh-Tari, Parham Vasaiely, Niklas Worschech, Robert Wotzlaw, Björn Zackris-son, and Azam Zia. Openmodelica users guide for open-modelica 1.9.0 beta3. version 2013-01-28. Technical report, Open Source Modelica Consortium, 2013.
- [FPA<sup>+</sup>13b] Peter Fritzson, Adrian Pop, Adeel Asghar, Willi Braun, Jens Frenkel, Lennart Ochel, Martin Sjölund, Per Östlund, Ole-na Rogovchenko, Peter Aronsson, Mikael Axin, Bernhard Bachmann, Vasile Baluta, Robert Braun, David Broman, Stefan Brus, Francesco Casella, Filippo Donida, Anand Gane-son, Mahder Gebremedhin, Pavel Grozman, Daniel Hedberg, Michael Hanke, Alf Isaksson, Kim Jansson, Daniel Kanth, Tommi Karhela, Juha Kortelainen, Abhinn Kothari, Petter Krus, Alexey Lebedev, Oliver Lenord, Ariel Liebman, Rickard Lindberg, Håkan Lundvall, Abhi Raj Metkar, Eric Meyers, Tuomas Miettinen, Afshin Moghadam, Maroun Nemer, Hannu Niemistö, Peter Nordin, Kristoffer Norling, Arun-kumar Palanisamy, Karl Pettersson, Pavol Privitzer, Jhansi

- Reddy, Reino Ruusu, Per Sahlin, Vladimir Schamai, Gerhard Schmitz, Alachew Shitahun, Anton Sodja, Ingo Staack, Kristian Stavåker, Sonia Tariq, Mohsen Torabzadeh-Tari, Parham Vasaiely, Niklas Worschech, Robert Wotzlaw, Björn Zackrisson, and Azam Zia. Openmodelica system documentation for openmodelica 1.9.0 beta3. version 2013-01-28. Technical report, Open Source Modelica Consortium, 2013.
- [Fri04] Peter Fritzson. *Principles of Object Oriented Modeling and Simulation with Modelica 2.1*. Wiley-IEEE Press, 2004.
- [Fri14] Peter Fritzson. *Principles of Object Oriented Modeling and Simulation with Modelica 3.3*. Wiley-IEEE Press, 2014.
- [Gal03] Gonzalo Galiano. Introducción al cálculo variacional. Technical report, Universidad de Oviedo, 2003.
- [GGM97] G. Gibbs, R. Gregory, and I. Moore. *Labs and practicals with more students and fewer resources*. Oxford: Oxford Centre for Staff Development, 1997.
- [GH83] John Guckenheimer and Philip Holmes. *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*. Springer Verlag, 1983.
- [GKC<sup>+</sup>09] Sharon C. Glotzer, Sangtae Kim, Peter T. Cummings, Abhijit Deshmukh, Martin Head-Gordon, George Karniadakis, Linda Petzold, Celeste Sagui, and Masanobu Shinozuka. International assessment of research and development in simulation-based engineering and science. Technical report, World Technology Evaluation Center, 2009.
- [GR09] Domenico Grimaldi and Sergio Rapuano. Hardware and software to design virtual laboratory for education in instrumentation and measurement. *Measurement*, 42:485–493, 2009.
- [Gro17] FMI Development Group. Functional mock-up interface. [link:43], Visitado en 2014-07-17.

- [Han06] Andrew Hanson. *Electric Power Engineering Handbook*, chapter Basic Electric Power Utilization - Loads, Load Characterization and Load Modeling, page 26.1 a 26.7. CRC Press, 2006.
- [HDK07] William H. Hayt, Steven M. Durbin, and Jack E. Kemmerly. *Analisis de Circuitos en Ingenieria*. McGraw Hill, 2007.
- [Hel11] Ola Høydal Helle. Electric hydraulic interaction. Master's thesis, Norwegian University of Science and Technology, Department of Electric Power Engineering, 2011.
- [HNG00] E. Hairer, S.P. Norsett, and G.Wanner. *Solving Ordinary Differential Equations I. Nonstiff Problems*. Springer, 2000.
- [HW02] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems*. Springer, 2002.
- [IEE07] IEEE. Ieee standard for calculating the current-temperature of bare overhead conductors. *IEEE Std 738-2006 (Revision of IEEE Std 738-1993)*, Jan:c1–59, 2007.
- [Int06] SIL International. Iso 639 code tables. [link:44], Visitada en 2013-09-06.
- [JK09] Simy Joy and David A. Kolb. Are there cultural differences in learning style? *International Journal of Intercultural Relations*, 33:69–85, 2009.
- [JPR08] S. Jeschke, O. Pfeiffer, and Th. Richter. User adaptive interactive courses in scorm compliant learning management systems. In *IMCL2008 Conference*, 2008.
- [JWJH12] Wenbin Jiang, Shuguang Wang, Hai Jin, and Yong Huang. A novel task management system for modelica-based multi-discipline virtual experiment platform. In *2012 IEEE Asia-Pacific Services Computing Conference*, pages 157–164, 2012.
- [Kat05] Tohru Katayama. *Subspace Methods for System Identification*. Springer, 2005.

- [KERS<sup>+</sup>12] Tiina M. Komulainen, Rasmus Enemark-Rasmussen, Gürkan Sin, John P. Fletcher, and David Cameron. Experiences on dynamic simulation software in chemical engineering education. *Education for chemical engineers*, 7:e153–e162, 2012.
- [Kim16] Mark Kimura. World3 simulator. [link:45], Visitado en 2014-08-16.
- [KK05] Alica Y. Kolb and David A. Kolb. Learning styles and learning spaces: Enhancing experiential learning in higher education. *Academy of Management Learning & Education*, 4:193–212, 2005.
- [Knu84] Donald E. Knuth. Literate programming. *The Computer Journal*, 27:97–111, 1984.
- [Kol81] David Kolb. *The Modern American College*, chapter Learning styles and disciplinary differences, pages 233–255. Jossey-Bass, 1981.
- [Kol84] David A. Kolb. *Experiential Learning - experience as a source of learning and development*. Prentice Hall, 1984.
- [Lan07] David C. Lane. The power of the bond between cause and effect (full version): Jay wright forrester and the field of system dynamics. *System Dynamics Review*, 23:95–118, 2007.
- [LG94] Lennart Ljung and Torkel Glad. *Modeling of dynamic systems*. Prentice Hall, 1994.
- [Lju87] Lennart Ljung. *System identification theory for the user*. Prentice Hall, 1987.
- [Mar07] Carla Martín. *Modelado Orientado a Objetos de Laboratorios Virtuales para la Educación en Control Automático*. PhD thesis, Universidad Nacional de Educación a Distancia, 2007.
- [Mat14] Mathworks. *Matlab. MAT-File Format*. Mathworks, versión 8.3. edition, Marzo 2014.
- [Mat16] Mathworks. Matlab. [link:46], Vistada en 2013-08-16.

- [MB12] Ion Matei and Conrad Bock. Modeling methodologies and simulation for dynamical systems. Technical report, National Institute of Standards and Technology. U.S. Department of Commerce, 2012.
- [MIM<sup>+</sup>74] D.L. Meadows, W.W. Behrens III, D.H. Meadows, R.F. Naill, and J. Randersand E.K.O Zah. *Dynamics of Growth in a Finite World*. Wright Allen Press, 1974.
- [MN11] Tassos A. Mikropoulos and Antonis Natsis. Educational virtual environments: A ten-year review of empirical research (1999-2009). *Computers & Education*, 56:769–780, 2011.
- [Mod16] Modelon. Optimica studio for physical modeling and design. [link:47], Visitado en 2013-07-16.
- [Mod16] Modelon. Fmi toolbox for matlab.[link:48], Visitado en 2013-08-16.
- [MoL06] Michael Magee and Canadian Council on Learning. State of the field review: Simulation in education: Final report. Technical report, Canadian Council on Learning, 2006.
- [MRM05] Donella Meadows, Jorgen Randers, and Dennis Meadows. *Limits to Growth. The 30 year update*. Earthscan, 2005.
- [MS11] Daniel Mueller and Stefan Strohmeier. Design characteristics of virtual learning environments: state of research. *Computers & Education*, 57:2505–2516, 2011.
- [MUD06] Carla Martín, Alfonso Urquía, and Sebastian Dormido. An approach to virtual-lab implementation using modelica. In *Proceedings of the 20th Annual European Simulation and Modelling Conference*, pages 137–141, 2006.
- [MUD08] Carla Martín, Alfonso Urquía, and Sebastián Dormido. An approach to virtual-lab implementation using modelica. *Mathematical and Computer Modelling of Dynamical Systems*, 14:341–360, 2008.
- [Naj05] Masoud Najafi. *Solveur Numérique pour les systèmes Algébro-Différentiels Hybrides*. PhD thesis, Université Paris XII, Val de Marne, 2005.

- [NFN05] M. Najafi, S. Furic, and R. Nikoukhah. Scicos: a general purpose modeling and simulation environment. In *Proceedings of the 4th International Modelica Conference*, 2005.
- [NVJ12] R. A. Naghizadeh, S. Jazebi, and B. Vahidi. Modeling hydro power plants and tuning hydro governors as an educational guideline. *International Review on Modelling and Simulations*, 5:1780–1790, 2012.
- [Nor73] William D. Nordhaus. World dynamics: Measurement without data. *The Economic Journal*, 83(332):1156–1183, 1973.
- [OC95] Martin Otter and Francois E. Cellier. *The Control Handbook*, chapter Software for modeling and simulating control systems, pages 415–428. CRC Press and IEEE Press, 1995.
- [oCaLA10] University of California at Los Angeles. Ucla lab videos. [link:49], Visitado en 2013-07-10.
- [oCaSB14] University of California at Santa Barbara. Computational science and engineering. [link:50], Visitado en 2014-09-14.
- [OEC15] OECD. *Frascati Manual 2015. Guidelines for collecting and reporting data on research and experimental development. The Measurement of Scientific, Technological and Innovation Activities*. OECD, 2015.
- [OLNSR98] Romeo Ortega, Antonio Loría, Per Johan Nickalsson, and Hebertt Sira-Ramírez. *Passivity-based control of Euler-Lagrange systems. Mechanical, electrical and electromechanical applications*. Springer, 1998.
- [OM96] P. Van Overschee and B. De Moor. *Subspace identification for linear systems*. Kluwer Academic Publishers, 1996.
- [OR94] Babatunde A. Ogunnaike and W. Harmon Ray. *Process dynamics, modeling and control*. Oxford University Press, 1994.
- [oST14] National Institutue of Standards and Technology. Lapack++. [link:51], Visitado en 2014-07-14.

- [Pet82] Linda R. Petzold. A description of dassl: a differential algebraic system solver. In *Conference: 10. international mathematics and computers simulation congress on systems simulation and scientific computation*, 1982.
- [pG16] El proyecto Gnu. Gnu octave. [link:52], Vistada en 2013-08-16.
- [PM12] Olga Proncheva and Sergey Makhov. *Artificial Intelligence Methods and Techniques for Business and Engineering Applications*, chapter J. Forrester's model of world dynamics and its development (Review), pages 191–201. ITHEA Science and Publishing, 2012.
- [PMM12] Bibudhendu Pati, Sudip Misr, and Atasi Mohanty. A model for evaluating the effectiveness of software engineering virtual labs. In *Technology Enhanced Education (ICTEE), 2012 IEEE International Conference on*, 2012.
- [Pri06] William W. Price. *Power system stability and control*, chapter Power System Dynamic Modeling. CRC Press, 2006.
- [RFT<sup>+</sup>09] M.G. Rasteiro, L. Ferreira, J. Teixeira, F.P. Bernardo, M.G. Carvalho, A. Ferreira, R.Q. Ferreira, F. Garcia, C.M.S.G. Baptista, N. Oliveira, M. Quina, L. Santos, P.A. Saraiva, A. Mendes, F. Magalhães, A.S. Almeida, J. Granjo, M. Ascenso, R.M. Bastos, and R. Borges. Labvirtual-a virtual platform to teach chemical processes. *education for chemical engineers*, 4:e9–e19, 2009.
- [RGLC12] Yois Smith Pascuas Rengifo, José Joaquín Bocanegra García, Edson Johann Ortiz Lozada, and José Nelson Pérez Castillo. Desarrollo dirigido por modelos para la creación de laboratorios virtuales. *Scientia et Technica*, 17:119–125, 2012.
- [Rod10] Sandy Rodger. A critique of the “world3” model used in “the limits to growth”. Technical report, University College London, 2010.
- [San03] M. Santander. Introducción al cálculo variacional. Technical report, Universidad de Valladolid, 2003.

- [Sch31] Jeff Schiller. Scour - an svg scrubber. [link:53], Visitada en 2013-12-31.
- [Sch00] Diana Estévez Schwarz. *Consistent initialization for index-2 differential algebraic equations and its application to circuit simulation*. PhD thesis, Humboldt Universität von Berlin, 2000.
- [Sch03] Steffen Schulz. Four lectures on differential-algebraic equations. Technical report, Humboldt Universität zu Berlin, 2003.
- [SDGL10] Hu Shaobin, Yin Daiyin, Cao Guangsheng, and Guo Lingling. Construction and application of autonomous learning based remote downhole tool virtual lab. In *2010 2nd International Conference on Education Technology and Computer (ICETC)*, pages VI343 – VI345, 2010.
- [SJK03] Jacquelin M.A. Scherpen, Dimitri Jeltsema, and J. Ben Klaassens. Lagrangian modeling of switching electrical networks. *Systems & Control Letters*, 48:365 – 374, 2003.
- [SOHA12] Shatha Abu Shanab, Salaheddin Odeh, Rami Hodrob, and Mahasen Anabtawi. Augmented reality internet labs versus hands-on and virtual labs: A comparative study. In *2012 International Conference on Interactive Mobile and Computer Aided Learning*, pages 17–21, 2012.
- [Sol16] SolidWorks. Solidworks. [link:54], Visitado en 2013-08-16.
- [SRGRCRLJ14] Hebertt Sira-Ramírez, Carlos García-Rodríguez, John Cortés-Romero, and Alberto Luviano-Juárez. *Algebraic identification and estimation methods in feedback control systems*. Wiley, 2014.
- [SS11] Richard O. Sinnott and Anthony J. Stell. Towards a virtual research environment for international adrenal cancer research. *Procedia Computer Science*, 4:1109–1118, 2011.
- [SYLL02] Dongil Shin, En Sup Yoon, Kyung Yong Lee, and Euy Soo Lee. A web-based, interactive virtual laboratory system for unit operations and process systems engineering education:

- issues, design and implementation. *Computers and Chemical Engineering*, 26:319– 330, 2002.
- [sys07] ISEE systems. Stella. systems thinking for education and research. [link:55], Visitado en 2014-07-07.
- [SZaZ11] Zhengyin Shi, Shenglin Zhao, and Shan an Zhu. An internet-based electrical engineering virtual lab: Using modelica for unified modeling. In *Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on*, 2011.
- [Til01] Micheal M. Tiller. *Introduction to physical modeling with Modelica*. Kluwer Academic Publisher, 2001.
- [Til23] Michael M. Tiller. Modelica by example. [link:56], Visitado en 2014-07-23.
- [THF11] Mohsen Torabzadeh-Tari, Zoheb Muhammed Hossain, Peter Fritzson, and Thomas Richter. Omweb - virtual web - based remote laboratory for modelica in engineering courses. In *Proceedings 8th Modelica Conference*, 2011.
- [UD03] Alfonso Urquia and Sebastian Dormido. Object-oriented description of hybrid dynamic systems of variable structure. *Simulation*, 79:485–493, 2003.
- [UMD05] Alfonso Urquía, Carla Martín, and Sebastián Dormido. Design of spicelib: A modelica library for modeling and analysis of electric circuits. *Mathematical and Computer Modelling of Dynamical Systems*, 11:43–60, 2005.
- [Uni18] Linköping University. Programming environments laboratory. [link:57], Visitado en 2014-07-18.
- [Vil09] Antonio Cañada Villar. Cálculo de variaciones. Technical report, Universidad de Granada, 2009.
- [Wel79] Peter E. Wellstead. *Introduction to Physical System Modelling*. Academic Press, 1979.
- [Wel00] Peter E. Wellstead. *Introduction to Physical System Modelling*. Control Systems Principles. [link:58], 2000.

- [WOB<sup>+</sup>99] Mathis Wackernagel, Larry Onisto, Patricia Bello, Alejandro Callejas Linares, Ina Susana López Falfán, Jesus Méndez García, Ana Isabel Suárez Guerrero, and Ma. Guadalupe Suárez Guerrero. National natural capital accounting with the ecological footprint concept. *Ecological Economic*, 29:375–390, 1999.
- [Wol16] Wolfram. Mathematica. [link:59], Vistada en 2013-08-16.
- [Zim06] Dirk Zimmer. A modelica library for multibond graphs and its application in 3d-mechanics. Master’s thesis, ETH Zürich. Department of Computer Science. Institute of Computational Science, 2006.
- [ZPK00] Bernard P. Zeigler, Herbert Praehofer, and Tag Gon Kim. *Theory of Modeling and Simulation*. Academic Press, 2000.

## LISTA DE ENLACES

Link	URL
1	<a href="https://github.com/modelica/PowerSystems">https://github.com/modelica/PowerSystems</a>
2	<a href="http://commons.wikimedia.org/wiki/File:Laptop.svg">http://commons.wikimedia.org/wiki/File:Laptop.svg</a>
3	<a href="http://commons.wikimedia.org/wiki/File:Unisex_user_icon.svg">http://commons.wikimedia.org/wiki/File:Unisex_user_icon.svg</a>
4	<a href="http://commons.wikimedia.org/wiki/File:Cartoon_cloud.svg">http://commons.wikimedia.org/wiki/File:Cartoon_cloud.svg</a>
5	<a href="http://www.phpclasses.org/browse/file/40373.html">http://www.phpclasses.org/browse/file/40373.html</a>
6	<a href="http://www.berthou.com">http://www.berthou.com</a>
7	<a href="http://www.berthou.com/fr/2008/03/19/phptliste-un-treeview-opensource-en-php/">http://www.berthou.com/fr/2008/03/19/phptliste-un-treeview-opensource-en-php/</a>
8	<a href="http://www.php.net/manual/en/function.xml-parse-into-struct.php">#84261</a>
9	<a href="http://odvarko.cz">http://odvarko.cz</a>
10	<a href="http://jscolor.com">http://jscolor.com</a>
11	<a href="https://openmodelica.org/download/download-linux">https://openmodelica.org/download/download-linux</a>
12	<a href="http://commons.wikimedia.org/wiki/File:Small_DC_Motor_Rotor.JPG">http://commons.wikimedia.org/wiki/File:Small_DC_Motor_Rotor.JPG</a>
13	<a href="http://commons.wikimedia.org/wiki/File:View_of_transmission_line_tower_on_LABPL_2_southwest_of_Boulder_City_Substation,_view_southwest_-_Hoover_Dam,_Los_Angeles_Bureau_of_Power_and_Light_Lines_1-3,_U.S._Highway_93,_Boulder_HAER_NV-27-M-13.tif">http://commons.wikimedia.org/wiki/File:View_of_transmission_line_tower_on_LABPL_2_southwest_of_Boulder_City_Substation,_view_southwest_-_Hoover_Dam,_Los_Angeles_Bureau_of_Power_and_Light_Lines_1-3,_U.S._Highway_93,_Boulder_HAER_NV-27-M-13.tif</a>
14	<a href="https://www.facebook.com/ArchivoGeneral">https://www.facebook.com/ArchivoGeneral</a>
15	<a href="http://commons.wikimedia.org/wiki/File:Cheoah_Hydroelectric_Dam_Graham_Co_NC.jpg">http://commons.wikimedia.org/wiki/File:Cheoah_Hydroelectric_Dam_Graham_Co_NC.jpg</a>
16	<a href="http://commons.wikimedia.org/wiki/File:Hydroelectric_dam-es.svg">http://commons.wikimedia.org/wiki/File:Hydroelectric_dam-es.svg</a>
17	<a href="http://commons.wikimedia.org/wiki/File:Hoover_Dam's_generators2.jpg">http://commons.wikimedia.org/wiki/File:Hoover_Dam's_generators2.jpg</a>
18	<a href="http://commons.wikimedia.org/wiki/File:Hoover_Dam_Generator.jpg">http://commons.wikimedia.org/wiki/File:Hoover_Dam_Generator.jpg</a>

19	<a href="http://commons.wikimedia.org/wiki/File:Innenpolmaschine.svg">http://commons.wikimedia.org/wiki/File:Innenpolmaschine.svg</a>
20	<a href="https://github.com/modelica/PowerSystems">https://github.com/modelica/PowerSystems</a>
21	<a href="http://commons.wikimedia.org/wiki/File:National_Museum_in_Poznan_-_painting_2.JPG">http://commons.wikimedia.org/wiki/File:National_Museum_in_Poznan_-_painting_2.JPG</a>
22	<a href="http://hjmacho.github.io/translation_GPLv3_to_spanish/">http://hjmacho.github.io/translation_GPLv3_to_spanish/</a>
23	<a href="http://www.aspentechnology.com/hysys/">http://www.aspentechnology.com/hysys/</a>
24	<a href="https://www.modelica.org/tools">https://www.modelica.org/tools</a>
25	<a href="https://www.modelica.org/libraries">https://www.modelica.org/libraries</a>
26	<a href="https://modelica.org/association">https://modelica.org/association</a>
27	<a href="https://modelica.org/licenses/ModelicaLicense2">https://modelica.org/licenses/ModelicaLicense2</a>
28	<a href="http://www.cadence.com/products/orcad/pages/default.aspx">http://www.cadence.com/products/orcad/pages/default.aspx</a>
29	<a href="http://www.wtec.org/sbes">http://www.wtec.org/sbes</a>
30	<a href="https://www.openmodelica.org/">https://www.openmodelica.org/</a>
31	<a href="https://www.openmodelica.org/index.php/userresources/userDocumentation">https://www.openmodelica.org/index.php/userresources/userDocumentation</a>
32	<a href="https://test.openmodelica.org/hudson/view/LibraryTesting/">https://test.openmodelica.org/hudson/view/LibraryTesting/</a>
33	<a href="http://www.mysql.com/">http://www.mysql.com/</a>
34	<a href="http://ptolemy.eecs.berkeley.edu/">http://ptolemy.eecs.berkeley.edu/</a>
35	<a href="http://www.w3.org/2002/Talks/www2002-svgtut-ih/hwtut.pdf">http://www.w3.org/2002/Talks/www2002-svgtut-ih/hwtut.pdf</a>
36	<a href="http://www2002.org">http://www2002.org</a>
37	<a href="http://www.inkscape.org">http://www.inkscape.org</a>
38	<a href="https://www.scilab.org/">https://www.scilab.org/</a>
39	<a href="http://stackoverflow.com/questions/14943950/generate-white-noise-in-modelica-systemmodeler">http://stackoverflow.com/questions/14943950/generate-white-noise-in-modelica-systemmodeler</a>
40	<a href="http://httpd.apache.org/">http://httpd.apache.org/</a>
41	<a href="https://www.eclipse.org/">https://www.eclipse.org/</a>
42	<a href="http://www.gnu.org/licenses/gpl-3.0.en.html">http://www.gnu.org/licenses/gpl-3.0.en.html</a>
43	<a href="https://www.fmi-standard.org/">https://www.fmi-standard.org/</a>
44	<a href="http://www-01.sil.org/iso639-3/codes.asp">http://www-01.sil.org/iso639-3/codes.asp</a>
45	<a href="http://www.world3simulator.org/">http://www.world3simulator.org/</a>
46	<a href="http://www.mathworks.com/products/matlab/">http://www.mathworks.com/products/matlab/</a>
47	<a href="http://www.modelon.com/products/optimica-studio-for-physical-modeling/">http://www.modelon.com/products/optimica-studio-for-physical-modeling/</a>
48	<a href="http://www.modelon.com/products/fmi-toolbox-for-matlab/">http://www.modelon.com/products/fmi-toolbox-for-matlab/</a>
49	<a href="http://lslab.lscore.ucla.edu/LAB/Video">http://lslab.lscore.ucla.edu/LAB/Video</a>
50	<a href="https://cse.cs.ucsb.edu/">https://cse.cs.ucsb.edu/</a>
51	<a href="http://math.nist.gov/lapack++/">http://math.nist.gov/lapack++/</a>

52	<a href="http://www.gnu.org/software/octave/">http://www.gnu.org/software/octave/</a>
53	<a href="http://www.codedread.com/scour/">http://www.codedread.com/scour/</a>
54	<a href="http://www.solidworks.com/">http://www.solidworks.com/</a>
55	<a href="http://www.iseesystems.com/softwares/Education/StellaSoftware.aspx">http://www.iseesystems.com/softwares/Education/StellaSoftware.aspx</a>
56	<a href="http://book.xogeny.com/">http://book.xogeny.com/</a>
57	<a href="http://www.ida.liu.se/labs/pelab/">http://www.ida.liu.se/labs/pelab/</a>
58	<a href="http://www.control-systems-principles.co.uk/ebooks/Introduction-to-Physical-System-Modelling.pdf">http://www.control-systems-principles.co.uk/ebooks/Introduction-to-Physical-System-Modelling.pdf</a>
59	<a href="http://www.wolfram.com/mathematica">http://www.wolfram.com/mathematica</a>



## UNVIRTUALLAB. UN LABORATORIO VIRTUAL BASADO EN OPENMODELICA

Hace parte de la Colección Ingenio Propio de la Facultad de Ingeniería.  
Este libro digital fue preparado en  $\text{\LaTeX}$ .  
Se publicó en septiembre de 2018 en Bogotá, D. C., Colombia.

En su composición se utilizaron caracteres  
Minion Pro 11/13.5 puntos,  
formato de 16.5 x 24 centímetros.

El libro presenta el diseño y desarrollo de *UNVirtualLab*, un ambiente web para alojar laboratorios virtuales de simulación de fenómenos dinámicos. En él se explica qué es *UNVirtualLab*, para qué puede utilizarse, cómo está diseñado, cómo se utiliza y cuáles son los fundamentos que le subyacen. El libro está conformado por tres partes y 14 capítulos; en la primera parte se explican los principios teóricos – matemáticos y de software – aplicados en la construcción de modelos orientados a objetos y su ulterior simulación en la suite *OpenModelica*; los capítulos que la componen se han redactado a manera de texto guía, y pueden ser utilizados en un módulo de algún curso de posgrado dedicado al modelado orientado a objetos. La segunda parte presenta en detalle las características del laboratorio virtual *UNVirtualLab* y en la tercera parte se ilustra su potencial con un conjunto de casos de ejemplo.

