Dinámica de los Sistemas Físicos.
Notas de Clase – Año 2023

Cátedra de DSF

Departamento de Control
Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Universidad Nacional de Rosario

Índice general

Capítulo 1

Sistemas Dinámicos y Modelos Matemáticos

En este curso trabajaremos con modelos matemáticos de sistemas dinámicos. En este primer capítulo introductorio presentaremos los principales conceptos relacionados tanto a los sistemas dinámicos como a los modelos matemáticos.

El contenido de este capítulo está principalmente basado en el libro de Peter Fritzson *Introducción al Modelado y Simulación de Sistemas Técnicos y Físicos con Modelica* [Fri15] y en el apunte *Sistemas Dinámicos y Modelos Matemáticos* [Jun15] de la cátedra de Dinámica de los Sistemas Físicos de nuestra Facultad.

1.1. Sistemas Dinámicos

1.1.1. El Concepto de Sistema

La palabra *sistema* es utilizada de manera permanente en nuestro lenguaje cotidiano para referirnos a diversas cosas tanto en el lenguaje coloquial como en el técnico-científico. En el contexto de este curso, sin embargo, intentaremos utilizar una definición más precisa de este concepto.

La Teoría General de Sistemas brinda múltiples definiciones para el concepto. Una definición bastante habitual en la literatura [Sky05] es la siguiente: "Un sistema es un conjunto de entidades interactuantes que conforman una unidad".

Otra definición muy similar proveniente de normas DIN Alemanas, establece en tanto que:

Definición 1.1. *Sistema*
Un Sistema es una disposición delimitada de entidades interactuantes.

En esta última definición podemos destacar las siguientes palabras:

Delimitación: Los sistemas tienen una *frontera* (espacial o conceptual) que los separa del resto del Universo. Esto no quiere decir que no pueda haber interacción entre el sistema y el resto, sino que los elementos que están fuera del sistema y que actúan sobre este se reemplazan por sus acciones. Estas acciones son vistas como *entradas* externas por el sistema.

Entidades Interactuantes: Son los *Componentes* del sistema, que pueden ser elementos, procesos o sistemas más simples (sub-sistemas).

Disposición (de las Entidades): Los sistemas tienen una *Estructura* de manera que el funcionamiento de los mismos no depende sólo de los componentes que tiene sino también de la manera en que dichos componentes están dispuestos y conectados.

De este análisis, podemos concluir de manera equivalente que un *Sistema* es una entidad formada por un conjunto de componentes y una estructura.

1.1.2. Clases de Sistemas

Según los procesos que involucran, los sistemas suelen clasificarse como *Físicos* (o *Concretos*) y *Abstractos* (o Conceptuales). Los Sistemas Físicos se caracterizan por su existencia en la realidad física espacio-temporal y en los mismos la interacción entre los componentes involucra el intercambio de materia, y/o energía, y/o información.

Los sistemas físicos, en tanto, se pueden clasificar como *dinámicos* o estáticos, siendo los primeros los que tienen capacidad de almacenar energía, materia o información.

Este curso está dedicado al estudio de los *Sistemas Físicos Dinámicos*, si bien en el transcurso del mismo trabajaremos eventualmente con sistemas más generales.

1.2. Modelos Matemáticos

En las disciplinas científicas y técnicas se trabaja con sistemas sobre los que habitualmente debemos aplicar métodos experimentales. Esto es, debemos realizar ensayos para observar la reacción del sistema a los mismos y establecer leyes (generalmente matemáticas) que expliquen el comportamiento de dichos sistemas.

Estos métodos experimentales, sin embargo, no son siempre viables. Hay distintos factores por los cuáles muchas veces no es conveniente (o es imposible) experimentar directamente sobre un sistema. Entre estos factores, encontramos:

Costos: Los experimentos sobre un sistema pueden tener costos que impidan su realización. Por ejemplo, realizar un experimento en una planta industrial podría significar detener un proceso de producción durante varias horas, con las consiguientes pérdidas económicas.

Riesgos: Las consecuencias del experimento pueden ser inadmisibles como por ejemplo un experimento sobre una planta nuclear para estudiar el escape de vapor radioactivo a la atmósfera.

Experimentos Irrealizables: En ocasiones un experimento no puede ser realizado porque las acciones involucradas son físicamente imposibles (por ejemplo, si nos quisiéramos preguntar cuál sería el efecto de un desplazamiento del eje magnético de la Tierra) o porque el sistema aún no existe (en la etapa de diseño de dicho sistema, típicamente). Este último caso es muy habitual en aplicaciones de Ingeniería.

Frente a la inconveniencia o a la imposibilidad de realizar experimentos directamente sobre los sistemas, se recurre a la experimentación sobre *modelos* de dichos sistemas.

1.2.1. El Concepto de Modelo

Definición 1.2. *Modelo*

Un modelo de un sistema es una herramienta que permite responder interrogantes sobre este último sin tener que recurrir a la experimentación sobre el mismo.

En la práctica, un modelo es una representación simplificada del sistema que representa donde la simplificación se realiza en función de la o las preguntas que se quieren responder. De esta manera, un modelo siempre involucra un sistema y uno o más interrogantes.

1.2.2. Clases de Modelos

Más allá de las disciplinas científicas y técnicas, los seres humanos utilizamos cotidianamente modelos (es decir, simplificaciones) de la realidad para poder comprender lo que nos rodea y ocurre. Estos modelos pueden ser de distinto tipo:

Modelos mentales: Hay incontables situaciones cotidianas en las cuales utilizamos modelos mentales para predecir lo que ocurriría ante potenciales acciones que se realicen. Por ejemplo, si sostenemos un objeto y lo soltamos desde cierta altura, no es necesario conocer las leyes de la Física para saber que va a caer al piso.

Modelos verbales/textuales: También utilizamos cotidianamente modelos muy simplificados que expresan de manera verbal o textual el comportamiento de ciertos sistemas ante diversas situaciones. Por ejemplo, la frase "Viento del este, agua como peste".

Modelos físicos: Son objetos físicos que reproducen algunas propiedades de sistemas reales para ayudarnos a responder preguntas sobre los mismos. Por ejemplo, un túnel de viento para el estudio de fenómenos aerodinámicos, una reproducción a escala del lecho de un río para estudios hidrológicos, etc.

Modelos matemáticos: Son modelos que describen mediante expresiones matemáticas (ecuaciones, desigualdades, expresiones lógicas) las relaciones entre las distintas magnitudes que caracterizan un sistema.

1.2.3. Clasificación de Modelos Matemáticos

Hay diversas formas de clasificar los modelos matemáticos, entre ellas encontramos:

Tiempo Continuo vs. Tiempo Discreto: La manera en la que se representa la evolución del tiempo en un modelo establece una pauta para su clasificación. Decimos que un modelo es de tiempo continuo cuando las relaciones entre las variables están definidas para todo instante de tiempo (dentro del intervalo de validez del modelo). Por el contrario, diremos que un modelo es de tiempo discreto cuando las relaciones entre las variables estén definidas sólo en determinados instantes discretos de tiempo.

Estáticos vs. Dinámicos: Un sistema se dice *dinámico* cuando hay relaciones entre variables que no dependen únicamente de sus valores instantáneos sino también de valores pasados de las mismas. Es decir, un sistema es dinámico cuando hay *memoria*. Por el contrario, en un sistema *estático* todas las relaciones entre variables dependen exclusivamente de los valores instantáneos de dichas variables. A modo de ejemplo, el modelo de un circuito eléctrico que contenga algún capacitor y/o inductor resultará dinámico, mientras que el modelo de un circuito que contenga únicamente elementos resistivos será estático.

Determinísticos vs. Estocásticos: Un modelo es determinístico si expresa matemáticamente sin incertidumbre las relaciones entre las variables. El modelo determinístico asigna de esta forma unívocamente valores y/o funciones ciertas y determinadas a la información que procesa. Por el contrario, un modelo es estocástico si expresa las relaciones con incertidumbre entre las variables mediante conceptos probabilísticos usando variables aleatorias. Dichas relaciones son descriptas usando variables o procesos estocásticos.

Parámetros Distribuidos vs. Parámetros Concentrados: Las magnitudes que caracterizan a los fenómenos físicos toman valores en el tiempo y en el espacio. Si un modelo matemático conserva la dependencia espacio-temporal en la representación de las magnitudos, se dice que el mismo es de *parámetros distribuidos*, ya que en general los coeficientes o parámetros del sistema están distribuídos en el espacio (por ejemplo, la densidad del fluido compresible en un gasoducto o la resistividad, inductancia y capacidad por unidad de longitud en una línea de transmisión). Los modelos a parámetros distribuidos son, típicamente, ecuaciones en derivadas parciales ya que involucran variación en el tiempo y espacio que resultan en derivadas respecto a la variable temporal y respecto a las coordenadas espaciales.

Cuando las variables de un modelo no dependen de las coordenadas espaciales o bien cuando se reemplaza la dependencia espacial de las variables por su promedio en la región del espacio donde están definidas se dice que el modelo es de *parámetros concentrados*. Estos modelos dinámicos son, típicamente, ecuaciones diferenciales ordinarias o más generalmente ecuaciones algebraico diferenciales.

Paramétricos vs. No Paramétricos: Un modelo *paramétrico* está caracterizado por un número finito y determinado de parámetros (por ejemplo, una función transferencia o un sistema de ecuaciones

diferenciales). Los modelos no paramétricos, en cambio, la caracterización se basa típicamente en series de datos cuya longitud no es fija (por ejemplo, la curva de la respuesta a un escalón de un sistema dinámico, la curva de respuesta en frecuencia de un amplificador).

Lineales vs. No Lineales: Se dice que un modelo es *lineal* cuando vale el principio de superposición, es decir que la acción de causas superpuestas provoca un efecto equivalente a la superposición de los correspondientes efectos.

Estacionarios vs. Inestacionarios: Un modelo se dice *estacionario* si estando en idénticas condiciones en dos instantes diferentes, responde de igual manera ante idénticas acciones.

Causales vs. Acausales: Un modelo se dice *causal* cuando además de describir relaciones matemáticas entre las magnitudes establece relaciones de causa-efecto entre las mismas. Los Diagramas de Bloques son la herramienta típica de modelado causal.

1.2.4. Elementos de los Modelos Matemáticos

Como ya mencionamos, los modelos matemáticos formulan mediante expresiones matemáticas (ecuaciones principalmente) las relaciones entre las magnitudes que caracterizan al mismo. Dependiendo de la naturaleza del modelo en cuestión, estas expresiones matemáticas pueden estar dadas de diversas maneras. Sin embargo, independientemente de la representación que utilicemos, las expresiones matemáticas involucradas siempre relacionarán magnitudes que se clasifican de la siguiente manera:

Variables Fundamentales: Son las variables que representan el tiempo y las coordenadas espaciales.

Variables Descriptivas: Son las variables que representan las magnitudes físicas asociadas al sistema. Entre las mismas encontramos:

Parámetros: Son magnitudes constantes generalmente o pueden ser variables pero con una ley predeterminada independiente de los procesos que puedan ocurrir en el sistema.

Entradas (Variables Independientes): Son variables descriptivas cuyas señales son independientes de otras señales en el sistema, y no están prefijadas. Representan acciones externas del resto del universo sobre el sistema. En aplicaciones de control se suelen dividir en dos categorías:

Entradas manipuladas: Entradas cuya trayectoria se puede manipular.

Perturbaciones: Entradas que son el resultado de fenómenos que están fuera del modelo y por lo tanto no se pueden ni manipular ni conocer a priori la evolución de sus trayectorias.

Variables Dependientes: Son variables descriptivas cuyas señales dependen de la evolución del sistema. Dentro de estas encontramos a las *salidas*, que son simplemente variables dependientes de interés.

Un concepto fundamental asociado a las variables que forman parte del modelo matemático de un sistema dinámico es el de *estado*, que puede definirse como sigue:

Definición 1.3. *Vector de Estados*
Un vector de estados *es un conjunto de variables dependientes cuyo conocimiento en un instante de tiempo permite calcular el valor de cualquier otra variable dependiente en dicho instante, asumiendo conocidos los valores de las entradas en ese instante de tiempo.*

Por ejemplo, en un circuito RLC (serie o paralelo) conociendo la tensión del capacitor y la corriente del inductor (además del valor de alguna entrada, si la hubiera) podemos calcular cualquier otra corriente o tensión. De esta forma, el vector formado por ambas variables sería un vector de estados. Alternativamente, podríamos elegir la carga en el capacitor y el flujo en el inductor y también podríamos calcular cualquier otra corriente o tensión del circuito, lo que muestra que hay múltiples manera de elegir el vector de estados.

Más aún, de acuerdo a la definición anterior, podríamos elegir como vector de estados las cuatro variables (tensión y carga del capacitor, corriente y flujo del inductor), lo que conduce a un vector con

estados redundantes (ya que conociendo la tensión del capacitor podemos saber la carga y viceversa, y lo mismo ocurre con el inductor). Efectivamente, el conjunto de cuatro variables es un vector de estados pero el mismo no es un *vector de estados minimal*:

Definición 1.4. *Vector de Estados Minimal*
Un vector de estados minimal *es un vector de estados que contiene la mínima cantidad posible de variables dependientes.*

El número de variables que constituyen un vector de estados minimal determina así el *orden* del modelo matemático, definido como sigue:

Definición 1.5. *Orden de un Modelo Matemático Es el número mínimo de variables dependientes con cuyo valor quedan estáticamente determinadas todas las otras variables dependientes del modelo.*

Comparando las últimas dos definiciones, queda claro que el orden de un modelo es la cardinalidad de cualquier vector de estados minimal.

1.3. Representación de Modelos Matemáticos

Hay diversas representaciones para los modelos matemáticos, las que resultan más o menos adecuadas según la naturaleza del sistema a representar y según el problema que se quiera resolver.

En este curso trabajaremos en gran medida con modelos de parámetros concentrados de sistemas físicos dinámicos, los que constituirán desde el punto de vista matemático sistemas de ecuaciones diferenciales ordinarias (ODEs, por sus siglas en inglés) o, más generalmente, sistemas de ecuaciones algebraico diferenciales (DAEs). Para las ODEs utilizaremos principalmente su representación en forma de *Ecuaciones de Estado*.

1.3.1. Ecuaciones de Estado

Dado el circuito de la Figura 1.1, podemos plantear las siguientes ecuaciones:

$$
\begin{aligned}
u_R(t) - R\ i_R(t) &= 0 && \text{(1.1a)}\\
\dot{q}(t) - i_C(t) &= 0 && \text{(1.1b)}\\
q(t) - C\ u_C(t) &= 0 && \text{(1.1c)}\\
\dot{\phi}(t) - u_L(t) &= 0 && \text{(1.1d)}\\
\phi(t) - L\ i_L(t) &= 0 && \text{(1.1e)}\\
u_S(t) - v(t) &= 0 && \text{(1.1f)}\\
u_L(t) + u_R(t) + u_C(t) - u_S(t) &= 0 && \text{(1.1g)}\\
i_L(t) - i_R(t) &= 0 && \text{(1.1h)}\\
i_C(t) - i_L(t) &= 0 && \text{(1.1i)}\\
i_S(t) - i_R(t) &= 0 && \text{(1.1j)}
\end{aligned}
$$

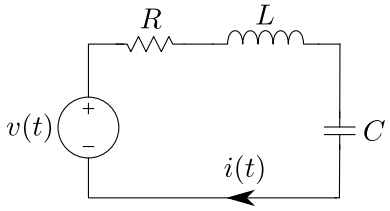

Figura 1.1: Circuito RLC Serie

Si trabajamos algebraicamente sobre el sistema de ecuaciones (1.1) podemos llegar fácilmente a la siguiente representación compacta

$$
\begin{aligned}
\dot{q}(t) &= \frac{1}{L}\ \phi(t)\\
\dot{\phi}(t) &= -\frac{1}{C}\ q(t) - \frac{R}{L}\ \phi(t) + u_S(t),
\end{aligned}
\tag{1.2}
$$

que tiene la forma de un sistema de *Ecuaciones de Estado*, en el cual las *variables de estado* son la carga $q(t)$ y el flujo $\phi(t)$ y la variable de entrada es la tensión de la fuente $u_S(t)$.

Puede verse fácilmente que, en concordancia con el concepto de *vector de estados* dado por la Definición 1.3, conociendo $q(t)$ y $\phi(t)$ además de la entrada $u_S(t)$, podemos calcular cualquier otra variable del modelo. Más aún, repasando la Definición 1.4, el vector $[q(t), \phi(t)]$ constituye un *Vector de Estados Minimal* por lo que el modelo matemático es de segundo orden.

Las Ecuaciones de Estado suelen complementarse con ecuaciones de salida estáticas que calculan ciertas variables dependientes de interés. Por ejemplo, si necesitáramos conocer las tensiones sobre la resistencia y sobre la inductancia ($u_R(t)$ y $u_L(t)$), podemos complementar el sistema de la Ec.(1.2) con las siguientes ecuaciones de salida

$$\begin{aligned} u_R(t) &= \frac{R}{L}\ \phi(t) \\ u_L(t) &= -\frac{1}{C}\ q(t) - \frac{R}{L}\ \phi(t) + u_S(t). \end{aligned} \tag{1.3}$$

En casos generales, la representación en ecuaciones de estado tendrá la forma

$$\begin{aligned} \dot{x}_1(t) &= f_1(x_1(t), \ldots, x_n(t), u_1(t), \ldots, u_m(t), t) \\ \dot{x}_2(t) &= f_2(x_1(t), \ldots, x_n(t), u_1(t), \ldots, u_m(t), t) \\ &\vdots \\ \dot{x}_n(t) &= f_n(x_1(t), \ldots, x_n(t), u_1(t), \ldots, u_m(t), t) \end{aligned} \tag{1.4}$$

donde $x_i(t)$ serán las *variables de estado* y las variables $u_i(t)$ serán las *variables de entrada*. Observar que esta notación permite describir relaciones *no lineales* a través de las funciones $f_i(\cdot)$ y representar también sistemas *inestacionarios* a través de la dependencia directa de las f_i con el tiempo t.

La Ecuación (1.4) suele utilizarse de manera vectorial, definiendo los vectores de estado $\mathbf{x} \triangleq [x_1, \ldots, x_n]^T$ y de entradas $\mathbf{u} \triangleq [u_1, \ldots, u_m]^T$ y reescribiendo entonces

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t). \tag{1.5}$$

Ocasionalmente complementaremos las Ecuaciones de Estados con Ecuaciones de Salida como la de la Eq.(1.3), que en el caso general tomarán la siguiente forma:

$$\begin{aligned} y_1(t) &= g_1(x_1(t), \ldots, x_n(t), u_1(t), \ldots, u_m(t), t) \\ y_2(t) &= g_2(x_1(t), \ldots, x_n(t), u_1(t), \ldots, u_m(t), t) \\ &\vdots \\ y_n(t) &= g_n(x_1(t), \ldots, x_n(t), u_1(t), \ldots, u_m(t), t) \end{aligned} \tag{1.6}$$

Las Ecuaciones de salida también admiten la representación vectorial

$$\mathbf{y}(t) = \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t), t), \tag{1.7}$$

donde $\mathbf{y} \triangleq [y_1, \ldots, y_p]^T$ es el vector de salidas.

Un caso particular de las Ecuaciones de Estado ocurre cuando la función $\mathbf{f}$ es *lineal* y por lo tanto las Ecs.(1.5) y (1.7) puede reescribirse como

$$\begin{aligned} \dot{\mathbf{x}}(t) &= A(t)\ \mathbf{x}(t) + B(t)\ u(t) \\ \mathbf{y}(t) &= C(t)\ \mathbf{x}(t) + D(t)\ u(t) \end{aligned} \tag{1.8}$$

para ciertas matrices $A(t)$, $B(t)$, $C(t)$ y $D(t)$. En tal caso diremos que el modelo matemático es *lineal*. Si además estas matrices fueran constantes (es decir, no dependieran de t), el modelo resultaría *lineal y estacionario*.

1.3.2. Ecuaciones Diferenciales Algebraicas

Las ecuaciones diferenciales algebraicas, como lo indica su nombre, son una combinación de ecuaciones diferenciales y ecuaciones algebraicas y pueden verse como una generalización de las ODEs. Este tipo de modelo matemático surge naturalmente al listar las relaciones matemáticas que aparecen entre las distintas variables de un sistema y por lo tanto suele ser la representación más simple de obtener en el proceso de modelado.

Por ejemplo, el sistema dado por las Ecs.(1.1a)-(1.1j) constituye un sistema de ecuaciones diferenciales algebraicas. En el mismo, podemos distinguir dos ecuaciones diferenciales, la Ec.(1.1b) y la Ec.(1.1d), y ocho ecuaciones puramente algebraicas.

Podemos ver también que el modelo contiene relaciones que son intrínsecas a los elementos del circuito y otras que se deben a la manera en la que dichos elementos están conectados. Específicamente, las Ecuaciones (1.1a)-(1.1f) se corresponden a las leyes asociadas a una resistencia, a un capacitor, a un inductor y a una fuente de tensión. Por el contrario, las Ecuaciones (1.1g)-(1.1j) se corresponden a la manera en que dichos elementos se encuentran conectados.

Las relaciones de un modelo matemático que son intrínsecas a los elementos se denominan *Relaciones Constitutivas*, mientras que las que expresan la manera en que los elementos se conectan se denominan *Relaciones Estructurales*. Como veremos más adelante, distinguir claramente estas relaciones va a ser fundamental a la hora de construir un modelo matemático.

De hecho, si a partir de un esquema gráfico como el de la Figura 1.1 quedaran unívocamente definidas cuáles son las relaciones constitutivas y estructurales del modelo matemático, podríamos utilizar directamente el esquema de dicha figura como representación equivalente al sistema de la DAE (1.1). En realidad, esto es lo que haremos más adelante cuando veamos los principios del *Modelado Orientado a Objetos*.

En un caso general las DAEs tendrán la forma:

$$\begin{aligned} f_1(x_1(t),\dots,x_n(t),\dot{x}_1(t),\dots,\dot{x}_n(t),a_1(t),\dots,a_r(t),u_1(t),\dots,u_m(t),t) &= 0 \\ f_2(x_1(t),\dots,x_n(t),\dot{x}_1(t),\dots,\dot{x}_n(t),a_1(t),\dots,a_r(t),u_1(t),\dots,u_m(t),t) &= 0 \\ \vdots & \\ f_{n+r}(x_1(t),\dots,x_n(t),\dot{x}_1(t),\dots,\dot{x}_n(t),a_1(t),\dots,a_r(t),u_1(t),\dots,u_m(t),t) &= 0 \end{aligned} \tag{1.9}$$

donde las variables $x_i(t)$ serán en principio los *estados* (más adelante veremos que algunas de estas componentes pueden no constituir verdaderamente estados). Las variables $a_i(t)$ serán las *variables algebraicas* y las variables $u_i(t)$ serán las *variables de entrada*. Observar que, al igual que en el caso de las Ecuaciones de Estado, esta notación permite describir relaciones *no lineales* a través de las funciones $f_i(\cdot)$ y representar también sistemas *inestacionarios* a través de la dependencia directa de las f_i con el tiempo t.

Al igual que las ODEs, la Ecuación (1.9) suele presentarse de manera vectorial, definiendo los vectores de estado $\mathbf{x} \triangleq [x_1,\dots,x_n]^T$, de entradas $\mathbf{u} \triangleq [u_1,\dots,u_m]^T$ y de variables algebraicas $\mathbf{a} \triangleq [a_1,\dots,a_r]^T$ y reescribiendo entonces

$$\mathbf{f}(\mathbf{x}(t),\dot{\mathbf{x}}(t),\mathbf{a}(t),\mathbf{u}(t),t) = 0. \tag{1.10}$$

Como ya mencionamos, las DAEs son generalmente la representación más simple de obtener cuando buscamos construir un modelo matemático. De hecho, basta con listar las relaciones constitutivas y estructurales de un modelo para tener una DAE sin ser necesario ningún tipo de trabajo algebraico. Sin embargo, muchas veces no resulta simple simular o realizar análisis de propiedades directamente sobre las DAEs. Para estos propósitos será conveniente convertirlas (cuando sea posible) en sistemas de ecuaciones diferenciales ordinarias que generalmente representaremos en su forma de *Ecuaciones de Estado*.

Este proceso de conversión de DAEs a ODEs se suele denominar *causalización* y tiene la ventaja de ser completamente algoritmizable y, por lo tanto, lo puede resolver una computadora. Más adelante en el curso veremos como es este proceso de causalización y qué herramientas informáticas pueden utilizarse para el mismo.

1.4. Utilización de Modelos Matemáticos

Cuando definimos el concepto de Modelo, dijimos que los mismos servían para responder ciertas preguntas sobre un sistema evitando realizar experimentos sobre dicho sistema. En el caso más específico de los modelos matemáticos, estas preguntas generalmente implican predecir el comportamiento del sistema ante distintas situaciones, ya sea a través de resultados cuantitativos o bien estableciendo ciertas propiedades cualitativas.

Pueden entonces establecerse a grandes rasgos dos maneras de utilizar un modelo matemático:

Análisis Teórico: consistente en el estudio analítico de los modelos, determinando propiedades tanto de carácter cualitativo (estabilidad, controlabilidad, observabilidad, etc.) como cuantitativo (a través de la resolución analítica de las ecuaciones de estado, por ejemplo).

Análisis Experimental: consistente en el estudio mediante *simulaciones* de los modelos para también establecer propiedades cualitativas y cuantitativas.

1.4.1. Análisis Teórico de los Modelos Matemáticos

Supongamos que queremos determinar si el transitorio que se produce al alimentar el circuito de la Figura 1.1 con un escalón de tensión en la fuente tiene o no oscilaciones en función de los parámetros R, L y C. Una manera relativamente simple de hacerlo sería obteniendo un sistema de Ecuaciones de Estados como la Ec.(1.8) (donde las matrices resultarán constantes) y calculando analíticamente los autovalores de la matriz A en función de los valores de los parámetros. Un procedimiento equivalente sería obtener la Función Transferencia entre la entrada y una salida y calcular los polos.

En los sistemas lineales y estacionarios (como lo es el caso del circuito de la Figura 1.1) suele ser relativamente sencillo analizar propiedades mediante análisis teórico, si bien para esto en general será necesario transformar primero el modelo dado por una DAE como la de la Ec. (1.1) en una ODE como la Ec.(1.2). Más adelante veremos como se puede realizar esta transformación de manera sistemática y luego, en el Capítulo 7, repasaremos algunas de las principales técnicas para realizar análisis sobre los modelos lineales y estacionarios.

En los sistemas no lineales, en cambio, es bastante más complejo establecer propiedades de carácter teórico. De hecho, las herramientas que vimos en cursos anteriores para analizar modelos matemáticos están restringidas a los modelos lineales y estacionarios. Más adelante, en el Capítulo 8, veremos sin embargo que en ocasiones podremos aproximar un modelo no lineal como el de la Ec.(1.5) por uno lineal como el de la Ec.(1.8) y obtener ciertas propiedades del modelo no lineal (como la estabilidad y la forma de las trayectorias) a partir de las propiedades del modelo *linealizado*.

Sin embargo, en sistemas más complejos o ante preguntas de carácter más cuantitativo las técnicas analíticas dejan de ser convenientes y la simulación digital resulta la herramienta más adecuada.

1.4.2. Simulación

El camino más simple y directo para estudiar la evolución de las variables de un modelo matemático ante condiciones iniciales y entradas determinadas es la *simulación digital*, que consiste en resolver las ecuaciones del modelo utilizando algún tipo de aproximación numérica (en el caso de los sistemas de tiempo continuo). Al contrario que en los métodos teóricos, veremos que podemos simular directamente sobre las DAEs ya que todo el proceso posterior a la formulación de las ecuaciones del modelo es totalmente algoritmizable.

La desventaja de la simulación frente al estudio analítico es que los resultados que se obtienen son particulares para un determinado juego de parámetros, para una determinada trayectoria de entradas y para un determinado conjunto de condiciones iniciales. Sin embargo, dado que muchas veces las simulaciones son muy rápidas, es posible realizarlas *barriendo* juegos de parámetros, entradas y condiciones iniciales, pudiendo en muchos casos establecerse resultados cualitativos a partir de múltiples experimentos.

En el Capítulo 2 veremos las principales técnicas para simulación de sistemas continuos.

1.4.3. Limitaciones al usar Modelos Matemáticos

Cuando formulamos un modelo matemático a partir de un sistema, introducimos ciertas simplificaciones acordes al problema que queremos resolver y a las preguntas que queremos responder con dicho modelo. Sin embargo, una vez obtenido el modelo matemático, las hipótesis que se utilizaron para realizar las simplificaciones pueden no quedar explicitadas.

Por ejemplo, en el esquema del circuito de la Figura 1.1 no aparecen las dimensiones físicas del mismo. Por lo tanto, no podemos determinar cuál será la máxima frecuencia a la cual las leyes de la teoría de circuitos constituyen una aproximación válida. Si alguien quisiera simular la respuesta de este circuito (o analizar las propiedades teóricas) a una entrada senoidal de muy alta frecuencia, el modelo matemático en principio lo permite. Sin embargo, dependiendo de las dimensiones físicas del sistema real, los resultados obtenidos podrán tener o no sentido para dicho sistema.

Es un error muy común analizar o simular el modelo de un sistema sin tener en cuenta que las trayectorias obtenidas estén dentro del rango en el cual la aproximación que brinda dicho modelo tiene sentido. Por eso es fundamental en todo momento relacionar los resultados obtenidos al usar un modelo con su correspondencia en el sistema real.

1.5. Construcción de Modelos Matemáticos

El proceso que se realiza para obtener un modelo matemático que permita responder ciertas preguntas sobre un sistema se denomina *Modelado*. Este proceso suele realizarse de diversas maneras, habiendo dos grandes grupos de técnicas conceptualmente diferentes pero de uso complementario en la práctica, como luego veremos:

Modelado por Primeros Principios : También denominado modelado analítico, físico, o de *caja blanca* se basa en la obtención de las ecuaciones del modelo a partir de leyes conocidas que rigen el comportamiento de las componentes del sistema y de la forma en que se relacionan las mismas.

Modelado por Identificación : Llamado también modelado por experimentación o modelado de *caja negra*, se basa en proponer un modelo matemático con una estructura determinada y luego ajustar sus parámetros en base a datos obtenidos mediante diversos experimentos.

Más allá de la diferencia conceptual entre ambos procedimientos, como ya dijimos, ambos son complementarios. En el modelado por primeros principios se suele recurrir a la experimentación para determinar los valores de ciertos parámetros. Asimismo, en el modelado por identificación la estructura del modelo propuesto se suele basar en cierto conocimiento de los primeros principios que rigen la dinámica del sistema en cuestión.

1.5.1. Modelado por Primeros Principios

El proceso de construcción de un modelo matemático por primeros principios puede dividirse generalmente en las siguientes etapas:

Formulación del problema : El primer paso para construir un modelo matemático es definir claramente el problema que se quiere resolver con dicho modelo. Recordemos que un modelo se define siempre a partir de un sistema y de un problema a resolver.

Determinación de los fenómenos relevantes : En función del problema a resolver se realizan diversas *hipótesis simplificatorias*, se definen las magnitudes que cuantifican los fenómenos relevantes y se delimita de esta manera el sistema físico.

Si el sistema en cuestión es un circuito eléctrico, por ejemplo, y el problema es estudiar el tiempo de respuesta de los transitorios, muy posiblemente podremos utilizar la teoría de circuitos clásica. Si en cambio el problema es el diseño de un disipador para dicho circuito, deberemos incluir fenómenos térmicos además. Si el problema en cambio fuera estudiar las emisiones electromagnéticas del circuito, tendremos que incluir en el modelo las ecuaciones de Maxwell y la teoría de líneas de transmisión.

Descomposición en estructura y componentes : Para simplificar la obtención del modelo matemático, habitualmente buscaremos reconocer distintos subsistemas simples (componentes) determinando a su vez la manera en que los mismos interactúan (estructura). El resultado de esta etapa es generalmente un esquema funcional del estilo de la Figura 1.1.

Obtención de las Relaciones Matemáticas : Una vez reconocidas las componentes y la estructura, se deberán formular las *Relaciones Constitutivas* y las *Relaciones Estructurales*. Para este paso bastará con hacer un esquema con las componentes y sus conexiones si se utiliza un lenguaje de modelado orientado a objetos como Modelica. En cualquier caso, el resultado de esta etapa será un sistema de ecuaciones diferenciales algebraicas como el de la Ec.(1.1) y a partir de este momento podemos decir que contamos con un Modelo Matemático. Sin embargo, es posible que el mismo no esté en la forma que lo necesitamos para resolver el problema en cuestión o incluso que aún no conozcamos los valores de algunos parámetros y sea necesario continuar con el proceso de modelado.

Manipulación del Modelo : Hay diversos procedimientos para convertir un sistema de DAEs en otro tipo de representación (ecuaciones de estado, función transferencia, diagramas de bloques, etc.). Estos procedimientos son completamente algoritmizables y pueden realizarse aprovechando diversas herramientas de software.

Validación y Ajuste de Parámetros : Una vez construido el modelo matemático, se suele corroborar mediante simulaciones que el mismo efectivamente replique el comportamiento del sistema original bajo ciertas condiciones en las que es posible observar lo que hace dicho sistema. En esta etapa, también se suelen obtener y ajustar los valores de ciertos parámetros del modelo para lograr que el comportamiento del mismo sea lo más similar posible al del sistema original. Hay ocasiones en las cuales esta etapa falla y es necesario revisar las hipótesis simplificatorias retomando el procedimiento de modelado desde dicha etapa.

1.5.2. Modelado por Identificación

El modelado por identificación se suele utilizar cuando no se conocen bien la estructura y los componentes que conforman el sistema real o bien cuando estos son demasiado complejos y por lo tanto el modelo matemático resultante sería excesivamente complicado para el problema que se quiere resolver.

Al igual que en el caso del modelado por primeros principios, el proceso de modelado por identificación puede plantearse a través de una secuencia de etapas que en este caso puede resumirse como sigue:

Obtención de Datos : En función del problema a resolver y de las características del sistema, se suelen realizar diversos experimentos que muestren el comportamiento del sistema ante distintas condiciones y/o secuencia de entradas. En ciertas aplicaciones no es posible experimentar, pero se cuenta con series de datos históricos permiten realizar este paso. Estos datos suelen dividirse en dos subconjuntos: los *datos de ajuste* y los *datos de validación*.

Elección de la Estructura del Modelo : De acuerdo al conocimiento previo sobre el sistema, a lo observado en los datos, y siempre en concordancia con el problema a resolver, se elije un conjunto de modelos que generalmente comparten cierta estructura pero no los valores de los parámetros y que podrían representar al sistema. Por ejemplo, este conjunto podría consistir en todas las funciones transferencias de segundo orden con respuesta subamortiguada.

Selección del Modelo : Del conjunto de modelos elegido en el punto anterior, se selecciona el que mejor replica los datos de ajuste. Este punto suele involucrar un proceso de *optimización* para la obtención de los parámetros del modelo.

Verificación del Modelo : Finalmente, se suele verificar que el modelo seleccionado en el paso anterior tenga un ajuste aceptable respecto de los datos de validación. Si esto falla, es probable que haya que modificar el conjunto de modelos y/o ampliar el conjunto de datos de ajuste realizando eventualmente más experimentos.

1.5.3. Modelado de Caja Gris

En la práctica generalmente el modelado no se realiza siguiendo estrictamente los primeros principios ni siguiendo estrictamente las técnicas de identificación, sino mediante una combinación de ambos procedimientos. Es muy común comenzar el modelado aplicando hasta donde sea posible los primeros principios pero dejando como incógnita los valores de ciertos parámetros. Luego, se suelen diseñar experimentos para obtener datos (o se pueden utilizar series de datos existentes) y a partir de estos datos, se aplican técnicas de identificación que permiten ajustar los valores de los parámetros desconocidos.

Esta forma de proceder que combina el modelado por primeros principios con la identificación se denomina *Modelado de Caja Gris*.

1.6. Herramientas de Software para Modelado y Simulación

El modelado de sistemas de cierta complejidad no puede concebirse hoy en día sin el apoyo de herramientas informáticas específicas, que simplifican enormemente el trabajo de construcción de los modelos matemáticos y su posterior manipulación para obtener resultados de simulación y analizar diversas propiedades de los mismos.

Hay diversas herramientas para modelar sistemas específicos de cada disciplina (como el PSpice y sus derivados para los circuitos electrónicos, por ejemplo) y también herramientas para sistemas generales como Simulink.

Dada la diversidad de herramientas y la falta de compatibilidad entre las mismas, a fines del siglo pasado la comunidad de Modelado y Simulación decidió desarrollar un lenguaje abierto para describir modelos que sea independiente de la disciplina y que tenga una definición abierta que permita su utilización en cualquier herramienta de software que desee implementarlo. Este lenguaje, llamado *Modelica*, es actualmente el estándar más aceptado para representar modelos matemáticos de sistemas dinámicos.

Por ejemplo, el sistema de DAEs presentado en la Ec.(1.1) para el caso con entrada $v(t) = \sin(t)$ puede representarse en Modelica utilizando el Código 1.1. La traducción totalmente directa y el modelo puede simularse utilizando cualquier herramienta de software que utilice el lenguaje Modelica.

Código 1.1: Modelo de la Ec.(1.1) en Modelica

```
model Circuit
    Real uR, iR, q, iC, uC, phi, uL, iL, uS, iS;
    parameter Real R=1, C=1, L=1;
equation
    uR - R * iR = 0;
    der(q) - iC = 0;
    q - C * uC = 0;
    der(phi) - uL = 0;
    phi - L * iL = 0;
    uS - sin(time) = 0;
    uL + uR + uC - uS = 0;
    iL - iR = 0;
    iC - iL = 0;
    iS - iR = 0;
end Circuit;
```

En particular, en este curso utilizaremos *OpenModelica*, una herramienta de código abierta que tiene un gran grado de desarrollo. Hay además herramientas comerciales como Dymola (la más eficiente y desarrollada) y Wolfram SystemModeler entre otras.

Más adelante veremos que utilizando el lenguaje Modelica no sólo podremos describir los modelos en forma de DAEs, sino también directamente a partir de sus componentes y estructuras brindando una representación textual o directamente gráfica.

1.7. Bibliografía Complementaria

En este capítulo presentamos sintéticamente varios temas, cuyas fuentes bibliográficas principales son el libro de Peter Fritzson [Fri15] y un apunte anterior de la cátedra [Jun15]. Quienes quieran profundizar sobre temas relacionados a la Teoría de Sistemas pueden recurrir a libros dedicados al tema como [Sky05] mientras que sobre Teoría de Modelado y Simulación se puede recomendar el libro *Theory of Modeling and Simulation* [ZMK18].

En este curso luego profundizaremos sobre otros temas (simulación de sistemas continuos, técnicas de modelado y técnicas de análisis). Respecto al modelado, trabajaremos casi exclusivamente utilizando primeros principios y en los capítulos posteriores profundizaremos al respecto. Quien quiera profundizar sobre modelado por identificación, la referencia bibliográfica más clásica es sin dudas el libro de Ljung *System identification* [Lju98].

1.8. Problemas Propuestos

[P1.1] Clasificación de Modelos Matemáticos

Dados los siguientes modelos matemáticos, clasificarlos según sean (a) Estáticos o Dinámicos. (b) Parámetros Distribuidos o Concentrados. (c) Lineales o No Lineales. (d) Estacionarios o Inestacionarios.

1. El modelo de la Ec.(1.1) suponiendo que $v(t)$ es una entrada externa.
2. El modelo de la Ec.(1.1) suponiendo que $v(t) = \sin(t)$ no es una entrada externa, sino que es parte del modelo.
3. El modelo de un péndulo dado por las EE/ES:

$$\begin{aligned} \dot{\theta}(t) &= \omega(t) \\ \dot{\omega}(t) &= -\frac{g}{l}\sin(\theta(t)) \\ x(t) &= l\ \cos(\theta(t)) \\ y(t) &= l\ \sin(\theta(t)) \end{aligned} \tag{P1.1a}$$

   donde $\theta(t)$ y $\omega(t)$ son la posición y velocidad angular del péndulo, g es la aceleración de la gravedad y l es la longitud del cable.

4. El modelo del calentamiento de una barra cilíndrica de cobre debido a la circulación de corriente eléctrica, dado por la ecuación:

$$\frac{\partial T(r,t)}{\partial t} = \sigma\left(\frac{\partial^2 T(r,t)}{\partial r^2} + \frac{1}{r}\cdot\frac{\partial T(r,t)}{\partial r} + \frac{P_{\mathrm{e}}(t)}{\lambda\cdot V}\right) \tag{P1.1b}$$

   donde $T(r,t)$ es la temperatura en el tiempo t a una distancia r del centro de la barra, $P_e(t)$ es la potencia eléctrica (entrada externa) que se disipa en la barra, mientras que σ (coeficiente de difusión), λ (conductividad) y V (volumen de la barra) son parámetros constantes.

   El modelo se completa con las *condiciones de borde*:

$$\begin{aligned} \frac{\partial T(0,t)}{\partial r} &= 0.0 \\ \frac{\partial T(R,t)}{\partial r} &= -k_1\left(T(R,t)^4 - T_{\mathrm{r}}(t)^4\right) - k_2\left(T(R,t) - T_{\mathrm{r}}(t)\right) \end{aligned} \tag{P1.1c}$$

   donde R es el radio de la barra, $T_r(t)$ es la temperatura ambiente y los coeficientes k_1 y k_2 son constantes.

5. El modelo de las Ecs.(P1.1b)–(P1.1c) suponiendo ahora que $k_1 = 0$.

6. El modelo dado por la Ecuación en Derivadas Parciales

$$\frac{\partial^2 \phi(x,y)}{\partial x^2} + \frac{\partial^2 \phi(x,y)}{\partial y^2} = \rho(x,y) \tag{P1.1d}$$

que representa el potencial eléctrico en un espacio de dos dimensiones (x,y) y donde $\rho(x,y)$ es la densidad de carga (que no cambia con el tiempo).

[P1.2] Orden de Modelos Matemáticos

Determinar el orden de los modelos matemáticos dinámicos del Problema P1.1. En cada caso, además, proponer un vector de estados minimal.

[P1.3] Orden de Modelos Matemáticos II

Determinar el orden de los siguientes modelos matemáticos, proponiendo además en cada caso un vector de estados minimal.

1. El modelo del circuito de La Fig.1.2, dado por la Ec.(P1.3a):

$$\begin{aligned} C_1\ \dot{u}_{C_1}(t) - i_{C_1}(t) &= 0 \\ C_2\ \dot{u}_{C_2}(t) - i_{C_2}(t) &= 0 \\ u_R(t) - R\ i_R(t) &= 0 \\ i_R(t) + i_{C_1}(t) + i_{C_2}(t) &= 0 \\ u_{C_1}(t) - u_R(t) &= 0 \\ u_{C_1}(t) - u_{C_2}(t) &= 0 \end{aligned} \tag{P1.3a}$$

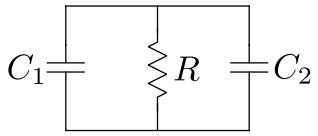

Figura 1.2: Circuito RC Paralelo

2. El modelo del péndulo de la Fig.1.3, dado por la Ec.(P1.3b).

$$\begin{aligned} \dot{x}(t) - v_x(t) &= 0 \\ \dot{y}(t) - v_y(t) &= 0 \\ m\ \dot{v}_x(t) + \frac{x(t)}{L}\ F(t) &= 0 \\ m\ \dot{v}_y(t) + \frac{y(t)}{L}\ F(t) - m\ g &= 0 \\ x(t)^2 + y(t)^2 &= 0 \end{aligned} \tag{P1.3b}$$

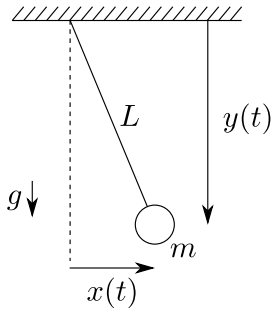

Figura 1.3: Péndulo

[P1.4] Relaciones Constitutivas y Estructurales

1. Clasificar las ecuaciones del modelo matemático de la Ec.(P1.3a) según sean constitutivas o estructurales.

2. Repetir el punto anterior para el modelo de la Ec.(P1.3b).

Capítulo 2

Simulación de Sistemas Continuos

En el Capítulo 1 mencionamos que una de las principales razones por las que se utilizan modelos matemáticos es por la posibilidad de experimentar sobre los mismos sin recurrir al sistema real. Estos experimentos que se realizan sobre los modelos matemáticos se denominan *simulaciones* y cuando los modelos son de tiempo continuo y toman la forma de ecuaciones diferenciales, dichas simulaciones involucran distintas estrategias de aproximación numérica ya que, en general, las ecuaciones diferenciales no pueden resolverse de manera analítica.

Este capítulo, basado principalmente en el libro *Continuous System Simulation* [CK06], tiene el objetivo de introducir de manera breve las bases de la integración numérica de ecuaciones diferenciales ordinarias y su uso en el contexto de la simulación de sistemas de tiempo continuo.

2.1. Principios de la Integración Numérica de ODEs

Consideremos a modo de ejemplo un sistema-masa–resorte–amortiguamiento muy simple como el de la Figura 2.1, que puede representarse por el siguiente sistema de ecuaciones:

$$\begin{aligned} \dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= \frac{1}{m}[-k\,x_1(t) - b\,x_2(t) + F(t)] \end{aligned} \tag{2.1}$$

donde la variable $x_1(t)$ representa la posición, $x_2(t)$ representa la velocidad y $F(t)$ es la fuerza de entrada aplicada al sistema. Los parámetros son la masa m, el coeficiente de roce b y la constante del resorte k.

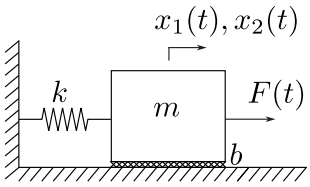

Figura 2.1: Sistema Masa Resorte

Si queremos determinar como evolucionarán las distintas variables de este sistema a medida que pasa el tiempo, deberemos resolver la Ecuación (2.1). En este caso, por ejemplo, tomando parámetros $m = b = k = 1$, suponiendo que la entrada es constante $F(t) = 1$, y asumiendo que $x_1(0) = x_2(0) = 0$, obtenemos la siguiente solución:

$$\begin{aligned} x_1(t) &= 1 - \frac{\sqrt{3}}{3}e^{-t/2}\sin\frac{\sqrt{3}}{2}t - e^{-t/2}\cos\frac{\sqrt{3}}{2}t \\ x_2(t) &= \frac{\sqrt{12}}{3}e^{-t/2}\sin\frac{\sqrt{3}}{2}t \end{aligned} \tag{2.2}$$

para todo $t \geq 0$. La gráfica de esta solución puede verse en la Figura 2.2

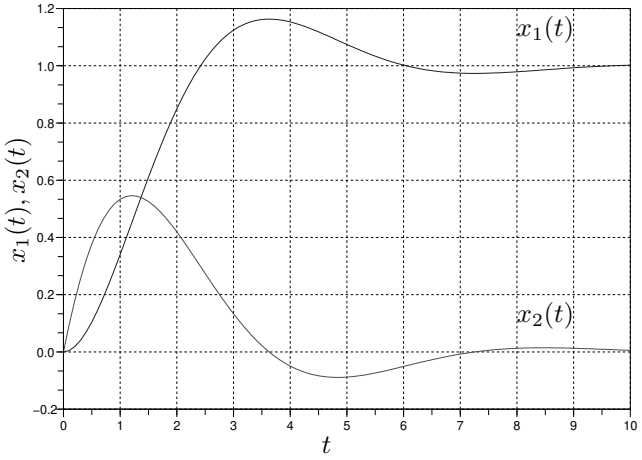

Figura 2.2: Solución de la Ecuación (2.1) del Sistema Masa Resorte

Si bien en este caso fue posible resolver de manera *analítica* la ecuación diferencial y obtener así la evolución de las variables en el tiempo, en general esto no será posible. En presencia de no linealidades, excepto casos muy simples y triviales, no es posible obtener una expresión analítica para la solución de una Ecuación Diferencial Ordinaria.

En los casos lineales, aún siendo posible resolver las ecuaciones (usando la Transformada de Laplace, por ejemplo), es muchas veces engorroso y poco práctico debido a la complejidad de los cálculos y de las expresiones resultantes.

Por todos estos motivos, generalmente se recurre a ditintos métodos que bajo ciertas condiciones permiten obtener soluciones aproximadas, normalmente mediante el uso de computadoras. Estos algoritmos, denominados *métodos de integración numérica* para ecuaciones diferenciales ordinarias constituyen la herramienta básica fundamental para la *Simulación de Sistemas Continuos*.

En general, nos interesará obtener las trayectorias de un sistema de ecuaciones de la forma

$$\begin{aligned}
\dot{x}_1 &= f_1(x_1, x_2, \ldots, x_n, u_1, \cdots, u_m, t) \\
\dot{x}_2 &= f_2(x_1, x_2, \ldots, x_n, u_1, \cdots, u_m, t) \\
&\vdots \\
\dot{x}_n &= f_n(x_1, x_2, \ldots, x_n, u_1, \cdots, u_m, t)
\end{aligned}$$

donde las x_i son las variables de estado y $u_i(t)$ son las entradas. Este sistema de ecuaciones puede reescribirse en forma vectorial como sigue:

$$\dot{\mathbf{x}}(t) = \hat{\mathbf{f}}(\mathbf{x}(t), \mathbf{u}(t), t)$$

donde definimos $\mathbf{x}(t) \triangleq [x_1, x_2, \ldots, x_n]^T$, $\mathbf{u}(t) \triangleq [u_1, \cdots, u_m]^T$ y $\hat{\mathbf{f}}(\cdot) \triangleq [f_1, \cdots, f_n]^T$.

Para poder calcular una solución numérica, será necesario siempre conocer completamente el vector de trayectorias de entrada $\mathbf{u}(t)$. Asumiendo entonces que este vector es conocido, no tiene sentido expresarlo como una variable. Por este motivo, en la literatura sobre métodos de integración numérica se define

$$\mathbf{f}(\mathbf{x}(t), t) \triangleq \hat{\mathbf{f}}(\mathbf{x}(t), \mathbf{u}(t), t)$$

de manera que el efecto de las entradas queda contenido en la propia definición de la función $\mathbf{f}$.

En virtud de esto, el objetivo de los métodos de integración numérica será encontrar una solución aproximada del sistema de ecuaciones diferenciales:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t) \tag{2.3}$$

a partir de la condición inicial conocida

$$\mathbf{x}(t_0) = \mathbf{x}_0 \tag{2.4}$$

2.1.1. Métodos de Euler

El método más sencillo para resolver la Ec.(2.3) fue propuesto por Leonhard Euler en 1768. Consiste básicamente en aproximar la derivada por el cociente incremental en la ecuación mencionada:

$$\dot{\mathbf{x}}(t) \approx \frac{\mathbf{x}(t+h) - \mathbf{x}(t)}{h} \approx \mathbf{f}(\mathbf{x}(t), t)$$

de donde despejamos

$$\mathbf{x}(t+h) \approx \mathbf{x}(t) + h\ \mathbf{f}(\mathbf{x}(t), t)$$

Llamando entonces $t_k = t_0 + k\ h$ para $k = 0, 1, \cdots$ y definiendo $\mathbf{x}_k \triangleq \mathbf{x}(t_k)$, resulta la fórmula de *Forward Euler*:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + h\ \mathbf{f}(\mathbf{x}_k, t_k) \tag{2.5}$$

El parámetro h, que define la distancia entre dos instantes de tiempo donde calculamos la solución, se denomina *paso de integración*. Más adelante veremos que h podrá ser constante o variable según el caso, y estudiaremos formas de elegir su valor adecuado.

El algoritmo de Forward Euler es entonces trivial: dado $\mathbf{x}_0$ y elegido un valor de h, hacemos $k = 0$ y utilizamos la fórmula anterior para calcular $\mathbf{x}_1$. Luego, a partir de este valor de $\mathbf{x}_1$ calculamos $\mathbf{x}_2$ y así sucesivamente.

El algoritmo terminará cuando lleguemos al paso en que $t_k = t_f$, donde el parámetro t_f se denomina *tiempo final* de la simulación. También veremos más adelante como elegir el tiempo final.

Para el sistema masa resorte de la Ec.(2.1), el método de Forward Euler con paso $h = 0.1$ comenzaría con los siguientes cálculos:

$$\mathbf{x}_0 = \begin{bmatrix}0\\0\end{bmatrix} \Rightarrow \mathbf{f}(\mathbf{x}_0, t_0) = \begin{bmatrix}x_2 = 0\\1 - x_1 - x_2 = 1\end{bmatrix} \Rightarrow \mathbf{x}_1 = \mathbf{x}_0 + h\ \mathbf{f}(\mathbf{x}_0, t_0) = \begin{bmatrix}0 + 0.1 \cdot 0\\0 + 0.1 \cdot 1\end{bmatrix} = \begin{bmatrix}0\\0.1\end{bmatrix}$$

$$\mathbf{x}_1 = \begin{bmatrix}0\\1\end{bmatrix} \Rightarrow \mathbf{f}(\mathbf{x}_1, t_1) = \begin{bmatrix}x_2 = 0.1\\1 - x_1 - x_2 = 1 - 0 - 0.1 = 0.9\end{bmatrix} \Rightarrow \mathbf{x}_2 = \mathbf{x}_1 + h\ \mathbf{f}(\mathbf{x}_1, t_1) = \begin{bmatrix}0 + 0.1 \cdot 0.1\\0.1 + 0.1 \cdot 0.9\end{bmatrix} =$$
$$= \begin{bmatrix}0.01\\0.19\end{bmatrix}$$

Esto es, de acuerdo al método de Forward Euler, $\mathbf{x}(t_1 = 0.1) \approx [0\ 0.1]^T$ y $\mathbf{x}(t_2 = 0.2) \approx [0.01\ 0.19]^T$.

La Figura 2.3 muestra el resultado de continuar calculando la solución de la Ec.(2.1) con este método numérico usando un paso de integración $h = 0.1$ y compara dicha solución con la analítica.

Como bien puede apreciarse en la gráfica, la solución numérica brindada por el método de Euler se aproxima bastante a la solución analítica. Más adelante analizaremos las causas del error y como se puede mejorar.

Otro método de integración numérica muy conocido, inspirado en el anterior y que recibe el nombre de *Método de Backward Euler*, reemplaza la fórmula de la Ec.(2.5) por la siguiente:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + h\ \mathbf{f}(\mathbf{x}_{k+1}, t_k) \tag{2.6}$$

El método de Backward Euler tiene un pequeño inconveniente: de acuerdo a la Ec.(2.6) para calcular $\mathbf{x}_{k+1}$ necesitamos conocer $\mathbf{x}_{k+1}$.

Naturalmente, conociendo $\mathbf{x}_k$ podemos resolver de alguna forma la Ec.(2.6) y obtener de allí el valor de $\mathbf{x}_{k+1}$. Por este motivo, se dice que el método de Backward Euler es un *Método Implícito*, ya que para encontrar cada valor debemos resolver una ecuación. En contraposición, el método de Forward Euler es un *Método Explícito* ya que cada valor de la solución numérica queda determinado sin necesidad de resolver ecuaciones algebraicas.

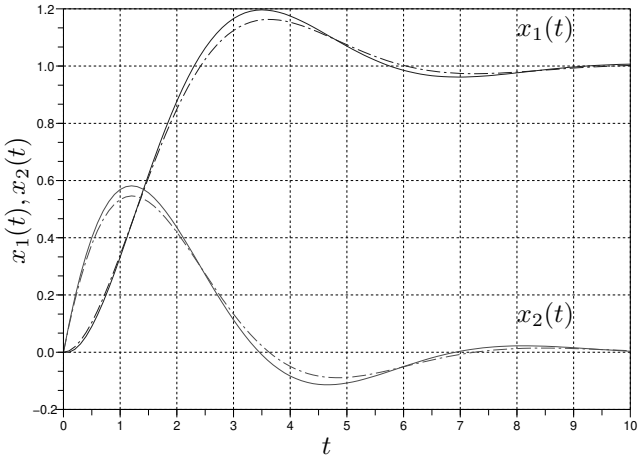

Figura 2.3: Solución de la Ecuación (2.1) del Sistema Masa Resorte con Forward Euler (sólida) con $h = 0.1$ y Solución Analítica (punteada)

Cuando la función $\mathbf{f}$ es lineal, la resolución de la ecuación implícita (2.6) es muy sencilla (si bien requiere invertir una matriz). Por el contrario, en el caso no lineal, en general necesitaremos utilizar algún algoritmo iterativo que encuentre la solución para cada instante de tiempo. Normalmente, se utiliza la *iteración de Newton*.

Por lo tanto, la implementación práctica del método de Backward Euler es mucho más compleja y computacionalmente costosa que la de Forward Euler. Más adelante, veremos que ventajas traerá que harán que valga la pena su uso.

La Figura 2.4 muestra el resultado de calcular la solución con Backward Euler usando un paso de integración $h = 0.1$ y compara dicha solución con la analítica. Como puede observarse, no hay una diferencia perceptible en cuanto a la precisión si se compara con lo que obtuvimos usando Forward Euler.

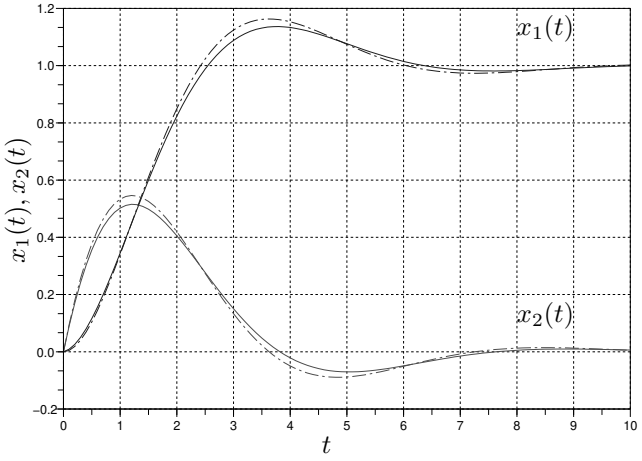

Figura 2.4: Solución de la Ecuación (2.1) del Sistema Masa Resorte con Backward Euler (sólida) con $h = 0.1$ y Solución Analítica (punteada)

2.1.2. La Iteración de Newton

Si queremos implementar de manera práctica el método de Backward Euler para simular sistemas no lineales, deberemos programar algún algoritmo que pueda *despejar* $\mathbf{x}_{k+1)}$ de la Ec.(2.6). El más utilizado para este propósito, tanto para Backward Euler como para cualquier otro método *implícito* es la iteración de Newton.

Supongamos que tenemos una ecuación algebraica

$$\mathbf{F}(\mathbf{x}) = 0 \tag{2.7}$$

y queremos encontrar una raíz de la misma, es decir, un valor del vector $\mathbf{x}$ que hace que dicha ecuación se cumpla. La iteración de Newton brinda un procedimiento que, partiendo de un valor inicial $\mathbf{x}^0$, permite obtener una sucesión de vectores $\mathbf{x}^1, \mathbf{x}^2, \ldots, x^N$, que bajo ciertas condiciones se aproxima cada vez más a la solución $\mathbf{x}$. La iteración está dada por la fórmula:

$$\mathbf{x}^{l+1} = \mathbf{x}^l - \left(\left. \frac{\partial \mathbf{F}}{\partial \mathbf{x}} \right|_{\mathbf{x}^l} \right)^{-1} \mathbf{F}(\mathbf{x}^l) \tag{2.8}$$

En la práctica, las iteraciones se realizan hasta que la diferencia entre $\mathbf{x}^{l+1}$ y $\mathbf{x}^l$ se torna más pequeña que cierta tolerancia pre-establecida.

Para el caso particular de Backward Euler, la Ec.(2.6) puede reescribirse como

$$\mathbf{x}_{k+1} - \mathbf{x}_k - h\ \mathbf{f}(\mathbf{x}_{k+1}, t_k) = \mathbf{F}(\mathbf{x}_{k+1}) = 0 \tag{2.9}$$

donde la incógnita es $\mathbf{x}_{k+1}$ por lo que la iteración resulta

$$\mathbf{x}_{k+1}^{l+1} = \mathbf{x}_{k+1}^l - \left(I - h\ \left. \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right|_{\mathbf{x}_{k+1}^l} \right)^{-1} \left(\mathbf{x}_{k+1}^l - \mathbf{x}_k - h\ \mathbf{f}(\mathbf{x}_{k+1}^l, t_k) \right) \tag{2.10}$$

Sobre este procedimiento, valen los siguientes comentarios:

- Para que esta iteración converja, es importante que la misma comience desde un punto inicial $\mathbf{x}_{k+1}^0$ no muy lejano a la solución. Por esto suele utilizarse $\mathbf{x}_{k+1}^0 = \mathbf{x}_k$.
- En cada iteración debemos calcular la matriz *Jacobiana* $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ evaluada en $\mathbf{x}_{k+1}^l$ y luego invertir la matriz $(I - h\ \frac{\partial \mathbf{f}}{\partial \mathbf{x}})$. Esto suele tener un costo computacional muy elevado, sobre todo cuando el sistema es de orden alto.
- Para evitar este costo, se suele calcular la matriz Jacobiana sólo en la primer iteración.
- Para las precisiones que se utilizan habitualmente en aplicaciones de ingeniería, el procedimiento suele converger tras 4 o 5 iteraciones.
- Si el sistema fuera lineal, puede verse que la iteración converge al valor exacto tras una sola iteración.

2.1.3. Precisión de las Aproximaciones

La Figura 2.5 muestra el resultado de calcular la solución del sistema Masa Resorte con distintos pasos de integración usando el método de Forward Euler. Como puede observarse,el error de la aproximación crece a medida que aumentamos el paso de integración h.

Trataremos entonces de justificar teóricamente esta observación.

Conocida la solución analítica en un punto $\mathbf{x}(t_k)$, podemos calcula la solución analítica tras un paso de integración de valor h haciendo uso de la expansión en serie de Taylor en torno a t_k:

$$\begin{aligned}
\mathbf{x}(t_{k+1}) &= \mathbf{x}(t_k) + h\ \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}(t_k) + \frac{h^2}{2!}\ \frac{\mathrm{d}^2\mathbf{x}}{\mathrm{d}t^2}(t_k) + \frac{h^3}{3!}\ \frac{\mathrm{d}^3\mathbf{x}}{\mathrm{d}t^3}(t_k) + \cdots \\
&= \mathbf{x}(t_k) + h\ \mathbf{f}(\mathbf{x}(t_k), t_k) + \frac{h^2}{2!}\ \frac{\mathrm{d}^2\mathbf{x}}{\mathrm{d}t^2}(t_k) + \frac{h^3}{3!}\ \frac{\mathrm{d}^3\mathbf{x}}{\mathrm{d}t^3}(t_k) + \cdots \\
&= \mathbf{x}(t_k) + h\ \mathbf{f}(\mathbf{x}(t_k), t_k) + o(h^2)
\end{aligned}$$

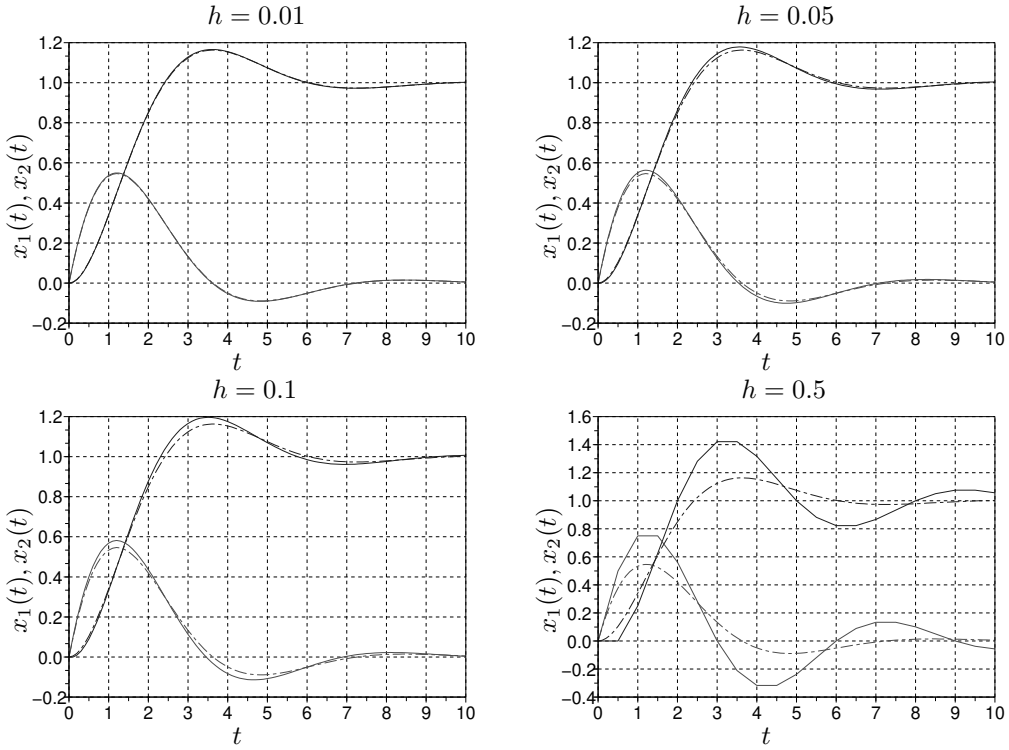

Figura 2.5: Solución de la Ecuación (2.1) del Sistema Masa Resorte con Forward Euler (sólida) para distintos pasos de integración y Solución Analítica (punteada)

Como podemos ver en esta expresión, el método de Forward Euler utiliza la expansión de Taylor sólo hasta el término de h, ignorando los términos de h^2, h^3, etc. Dado que el método iguala la serie hasta el término de h^1, se dice que es un algoritmo de primer orden.

El error que comete el método de Euler tras el primer paso es del orden de h^2. Esto es, asumiendo que se conoce la solución analítica en t_k el método de Euler cometerá un error proporcional a h^2 en t_{k+1}, es decir, en un paso del algoritmo.

Este error (cometido tras un paso) se denomina *Error Local por Truncamiento*[1], y en general aumentará a medida que se aumente el paso de integración. Este error siempre será proporcional a h^{N+1} donde N es el orden del método de integración.

En realidad, salvo en el primer paso, no conoceremos el valor de la solución analítica en t_k. Por lo tanto, en t_{k+1} habrá un error adicional ya que en cada caso estaremos partiendo de un valor inexacto en t_k. Por lo tanto, la diferencia que observamos entre la solución analítica y la solución numérica en un instante arbitrario de tiempo no es el error local, sino lo que se denomina *Error Global*.

Puede demostrarse que el máximo valor que alcanza el error global tiene siempre un orden menos que el error local. Esto es, en un método de orden N el error local es proporcional a h^{N+1} mientras que el máximo error global es proporcional a h^N.

En los métodos de Forward y Backward Euler, al ser ambos de primer orden, el máximo error global resulta proporcional al paso h. Por lo tanto, si disminuimos 10000 veces el paso de integración el error máximo disminuirá aproximadamente 10000 veces. Esto es, para realizar una simulación 10000 veces más precisa necesitamos usar un paso 10000 veces más chico, lo que implica realizar 10000 veces más cálculos.

En un método de cuarto orden, en cambio, disminuir 10 veces el paso de integración implica disminuir 10000 veces el error. Entonces, para realizar una simulación 10000 veces más precisa con un método de cuarto orden necesitamos usar un paso 10 veces más chico, lo que sólo implica realizar 10 veces más cálculos.

[1]Se habla de truncamiento ya que se puede pensar que el método *trunca* la serie de Taylor a partir de h^2.

Esta última observación tiene una implicancia importante: cuando queremos simular con mucha precisión debemos necesariamente utilizar métodos de orden alto. De otra manera, estaremos obligados a disminuir mucho el paso de integración lo que implica realizar muchos pasos y por lo tanto muchos cálculos. Si bien como veremos más adelante los métodos de orden mayor tienen mayor costo computacional por paso, este costo se compensa al utilizar pasos más largos obteniendo mejor precisión.

Si bien los errores de truncamiento son generalmente los más importantes, hay aplicaciones en las que no se pueden ignorar los *Errores de Redondeo*. Estos errores se deben a la representación numérica con un número finito de bits en las computadoras.

Para evitar estos errores siempre se intenta trabajar con representaciones de punto flotante de 64 bits (doble precisión), lo que permite en general despreciar los efectos del redondeo. De todas maneras, en algunas aplicaciones que requieren gran precisión se utilizan métodos de orden muy alto (para dinámica de cuerpos celestes, por ejemplo) y los efectos del redondeo no pueden ignorarse por completo.

Una característica interesante de los errores de redondeo es que, a diferencia de los errores por truncamiento, tienden a aumentar a medida que el paso de integración disminuye. Una explicación simplificada de este fenómeno puede deducirse de observar que ocurre en la Fórmula de Euler (2.5) al disminuir h. Cuando sumamos en punto flotante un número grande con uno muy pequeño, debido al redondeo, obtenemos nuevamente el número grande. Si repetimos indefinidamente la operación, seguiremos por lo tanto obteniendo siempre el primer número grande como resultado.

2.1.4. Estabilidad Numérica

Una pregunta que surge naturalmente tras analizar el error global de un método es si el mismo se sigue acumulando indefinidamente o no a medida que avanza el tiempo de la simulación. Observando nuevamente la Figura 2.5, vemos que a medida que la solución analítica se acerca a su valor final, la solución numérica tiende al mismo valor.

En otras palabras, en la Figura 2.5 vemos que la solución numérica preserva la estabilidad y el punto de equilibrio de la solución analítica.

Ahora bien, ¿ocurre esto para cualquier paso de integración?. La Figura 2.6 muestra que ocurre si seguimos aumentando el paso h.

Evidentemente no siempre se preserva la estabilidad. Para $h = 1$ perdemos la *estabilidad asintótica* y para $h = 1.5$ la solución numérica es claramente inestable.

Veamos que ocurre si, en cambio, utilizamos el método de Backward Euler. La Figura 2.7 muestra los resultados de este método implícito y como puede observarse, la estabilidad se preserva en todos los casos.

¿Como podemos justificar teóricamente estos resultados?

Si aplicamos el método de Forward Euler a un sistema lineal y estacionario,

$$\dot{\mathbf{x}}(t) = A\ \mathbf{x}(t) + B\ \mathbf{u}(t) \tag{2.11}$$

resulta,

$$\mathbf{x}_{k+1} = \mathbf{x}_k + h\ (A\ \mathbf{x}_k + B\ \mathbf{u}_k) = (I + h\ A)\ \mathbf{x}_k + h\ B\ \mathbf{u}_k$$

Es decir, la ecuación en diferencias resultante tiene matriz de evolución

$$A_d = (I + h\ A)$$

Los autovalores λ_d de A_d serán las soluciones de

$$\det(\lambda_d\ I - A_d) = \det(\lambda_d\ I - I - h\ A) = 0$$

Si el determinante de una matriz es nulo, también lo es el determinante del producto por h^{-1}, por lo que,

$$\det(h^{-1}\ (\lambda_d - 1)\ I - A) = 0$$

de donde resulta que los autovalores de A_d y los de A se relacionan según:

$$h^{-1}\ (\lambda_d - 1) = \lambda$$

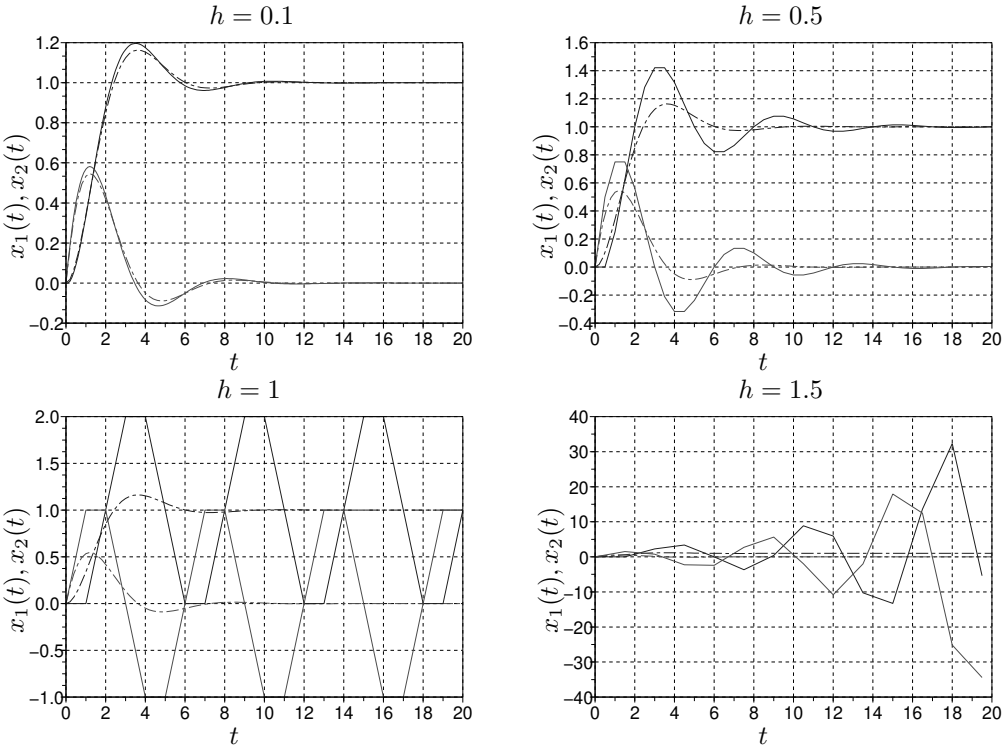

Figura 2.6: Solución de la Ecuación (2.1) del Sistema Masa Resorte con Forward Euler (sólida) para distintos pasos de integración y Solución Analítica (punteada)

de donde,

$$\lambda_d = \lambda\ h + 1 \tag{2.12}$$

Es decir, los autovalores de la matriz del sistema discreto aproximado pueden calcularse directamente en función de los autovalores del sistema continuo original.

Si aproximamos la solución de un sistema estable, será deseable que la aproximación resulte también estable. Para esto, deberá cumplirse para todos los autovalores de A que

$$|\lambda\ h + 1| < 1 \tag{2.13}$$

lo que sólo ocurrirá para valores pequeños de h. Esta condición de estabilidad del método deberá cumplirse siempre y es fundamental a la hora de elegir el paso de integración.

El sistema masa resorte del ejemplo tiene autovalores $\lambda_{1,2} = -0.5 \pm j\frac{\sqrt{3}}{2}$. Puede verse entonces que la condición de estabilidad de la Ec. (2.13) se cumple sólo para $h < 1$, lo que corrobora los resultados de la Fig.2.6.

Si aplicamos en cambio Backward Euler al sistema lineal y estacionario de la Ec.(2.11), resulta

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{x}_k + h\ (A\ \mathbf{x}_{k+1} + B\ \mathbf{u}_{k+1}) \Rightarrow \\ (I - h\ A)\ \mathbf{x}_{k+1} &= \mathbf{x}_k + h\ B\ \mathbf{u}_{k+1} \\ \mathbf{x}_{k+1} &= (I - h\ A)^{-1}\ [\mathbf{x}_k + h\ B\ \mathbf{u}_{k+1}] \end{aligned}$$

de donde la ecuación en diferencias resultante tiene matriz de evolución

$$A_d = (I - h\ A)^{-1}$$

Los autovalores λ_d de A_d serán las soluciones de

$$\det(\lambda_d\ I - A_d) = \det(\lambda_d\ I - (I - h\ A)^{-1}) = 0$$

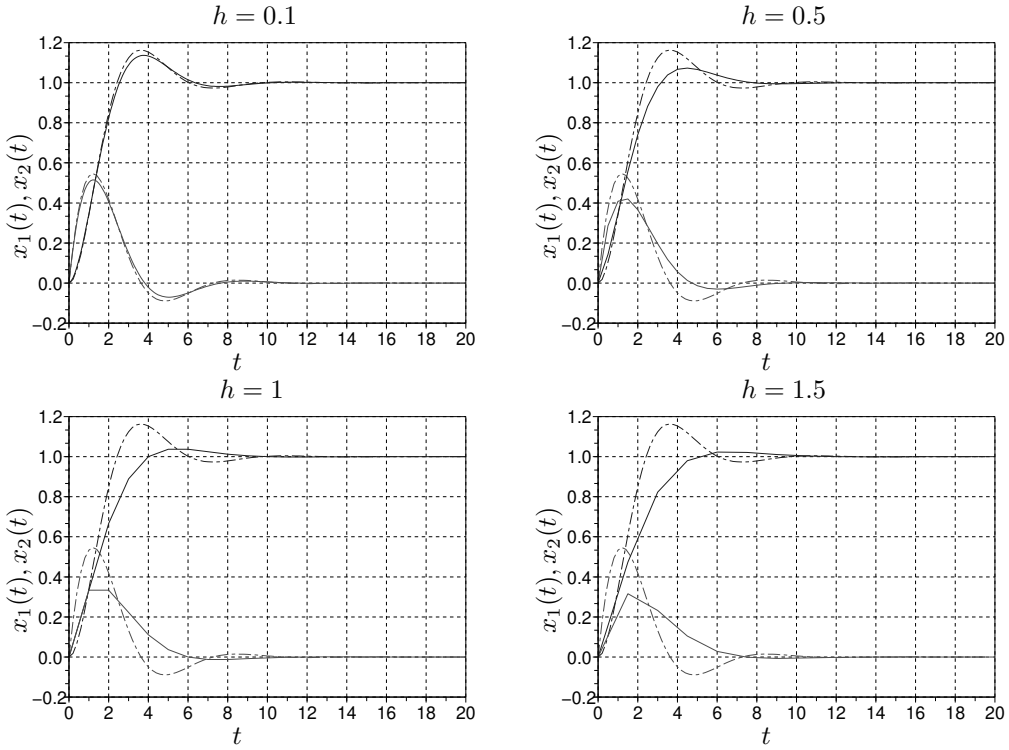

Figura 2.7: Solución de la Ecuación (2.1) del Sistema Masa Resorte con Backward Euler (sólida) para distintos pasos de integración y Solución Analítica (punteada)

Si el determinante de una matriz es nulo, también lo es el determinante del producto por $\frac{(I-h\ A)}{h\ \lambda_d}$ de donde,

$$\det(\lambda_d\ \frac{(I-h\ A)}{h\ \lambda_d} - \frac{I}{h\ \lambda_d}) = \det(I\ (\frac{1}{h} - \frac{1}{h\ \lambda_d}) - A) = 0$$

de donde resulta que los autovalores de A_d y los de A se relacionan según:

$$\frac{1}{h} - \frac{1}{h\ \lambda_d} = \lambda$$

o bien

$$\lambda_d = \frac{1}{1-\lambda\ h}$$

Por lo tanto, la condición para que el resultado numérico sea estable es que

$$|\frac{1}{1-\lambda\ h}| < 1 \tag{2.14}$$

Puede verse fácilmente que cuando $\mathbb{R}\mathrm{e}(\lambda) < 0$, la condición se cumplirá siempre. Esto es, al simular un sistema estable con el método de Backward Euler, la solución numérica siempre resultará estable (hecho que corrobora lo observado en la Fig.2.7).

Esta propiedad de preservar la estabilidad numérica es la principal ventaja de Backward Euler sobre Forward Euler. Más adelante veremos que esta característica será imprescindible para poder simular adecuadamente ciertos sistemas.

Entre tanto, una desventaja del algoritmo de Backward Euler es que para autovalores inestables $\mathbb{R}\mathrm{e}(\lambda) > 0$ puede resultar (si h es muy grande) $|\lambda_d| < 1$ (estable). Es decir, la simulación de un sistema inestable puede resultar estable, lo cual es indeseado.

Por ejemplo, el sistema inestable

$$\dot{x}(t) = x(t)$$

tiene un autovalor $\lambda = 1$. Por lo tanto, usando cualquier paso $h > 2$, vemos que la condición de la Ec.(2.14) se cumple lo que indica que la solución numérica será estable.

2.2. Métodos Monopaso

Como mencionamos antes, los métodos de Forward y Backward Euler que vimos hasta aquí realizan sólo aproximaciones de primer orden. Debido a esto, si se quiere obtener una buena precisión en la simulación, se debe reducir excesivamente el paso de integración lo que implica una cantidad de pasos y de cálculos en general inaceptable.

Para obtener aproximaciones de orden mayor, será necesario utilizar más de una evaluación de la función $\mathbf{f}(\mathbf{x}, t)$ en cada paso. Cuando dichas evaluaciones se realizan de manera tal que para calcular $\mathbf{x}_{k+1}$ sólo se utiliza el valor de $\mathbf{x}_k$, se dice que el algoritmo es *monopaso*. Por el contrario, cuando se utilizan además valores anteriores de la solución ($\mathbf{x}_{k-1}$, $\mathbf{x}_{k-2}$, etc.), se dice que el algoritmo es *multipaso*.

Los métodos monopaso se suelen denominar también *Métodos de Runge–Kutta*, ya que el primero de estos métodos de orden alto fue formulado por Runge y Kutta a finales del siglo XIX.

2.2.1. Métodos Explícitos de Runge Kutta

Los métodos explícitos de Runge–Kutta realizan varias evaluaciones de la función $\mathbf{f}(\mathbf{x}, t)$ en cercanías del punto $(\mathbf{x}_k, t_k)$ y luego calculan $\mathbf{x}_{k+1}$ realizando una suma pesada de dichas evaluaciones.

Uno de los algoritmos más simple es el denominado *Método de Heun*, inventado por Heun en el año 1900. Dicho método utiliza la fórmula

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \frac{h}{2}\,(\mathbf{k}_1 + \mathbf{k}_2) \tag{2.15}$$

donde

$$\begin{aligned} \mathbf{k}_1 &= \mathbf{f}(\mathbf{x}_k, t_k) \\ \mathbf{k}_2 &= \mathbf{f}(\mathbf{x}_k + h\ \mathbf{k}_1, t_k + h) \end{aligned} \tag{2.16}$$

Comparado con Euler, este método realiza una evaluación adicional de la función $\mathbf{f}$ para calcular $\mathbf{k}_2$. El beneficio que se obtiene es que el método de Heun realiza una aproximación de segundo orden y resulta mucho más preciso que Euler.

Para el sistema masa resorte de la Ec.(2.1), el método de Heun con paso $h = 0.1$ comenzaría con los siguientes cálculos:

$$\begin{aligned} \mathbf{x}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Rightarrow \\ \mathbf{k}_1 = \mathbf{f}(\mathbf{x}_0, t_0) = \begin{bmatrix} x_2 \\ 1 - x_1 - x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\ \mathbf{x}_0 + h\ \mathbf{k}_1 = \begin{bmatrix} 0 + 0.1 \cdot 0 \\ 0 + 0.1 \cdot 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} \Rightarrow \\ \mathbf{k}_2 = \mathbf{f}(\mathbf{x}_0 + h\ \mathbf{k}_1, t_0 + h) = \begin{bmatrix} 0.1 \\ 1 - 0 - 0.1 \end{bmatrix} = \begin{bmatrix} 0.1 \\ 0.9 \end{bmatrix} \\ \mathbf{x}_1 = \mathbf{x}_0 + \frac{h}{2}\,(\mathbf{k}_1 + \mathbf{k}_2) = \begin{bmatrix} 0 + 0.05 \cdot (0 + 0.1) \\ 0 + 0.05 \cdot (1 + 0.9) \end{bmatrix} = \begin{bmatrix} 0.005 \\ 0.095 \end{bmatrix} \end{aligned}$$

Otro método, uno de los más utilizados, es el de Runge–Kutta de orden 4 (RK4), desarrollado por Runge y Kutta en 1895:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \frac{h}{6}\,(\mathbf{k}_1 + 2\ \mathbf{k}_2 + 2\ \mathbf{k}_3 + \mathbf{k}_4)$$

donde

$$\begin{aligned}
\mathbf{k}_1 &= \mathbf{f}(\mathbf{x}_k, t_k) \\
\mathbf{k}_2 &= \mathbf{f}(\mathbf{x}_k + h\ \frac{\mathbf{k}_1}{2}, t_k + \frac{h}{2}) \\
\mathbf{k}_3 &= \mathbf{f}(\mathbf{x}_k + h\ \frac{\mathbf{k}_2}{2}, t_k + \frac{h}{2}) \\
\mathbf{k}_4 &= \mathbf{f}(\mathbf{x}_k + h\ \mathbf{k}_3, t_k + h)
\end{aligned}$$

Este algoritmo, como vemos, utiliza cuatro evaluaciones de la función $\mathbf{f}$ en cada paso. En este caso, el beneficio es que resulta una aproximación de orden 4.

La Tabla 2.1 resume los resultados de simular el sistema Masa Resorte de la Ec.(2.1) con los métodos de Euler, Heun y Runge–Kutta de orden 4. En cada caso, se reporta el máximo error obtenido en toda la simulación (esto es, el máximo error global). Puede notarse como en Euler el error disminuye linealmente con el paso, mientras que en Heun lo hace de forma cuadrática y en RK4 lo hace con la cuarta potencia del paso.

Paso	Error Euler	Error Heun	Error RK4
$h = 0.5$	0.298	0.0406	4.8×10^{-4}
$h = 0.1$	0.042	1.47×10^{-3}	6.72×10^{-7}
$h = 0.05$	0.0203	3.6×10^{-4}	4.14×10^{-8}
$h = 0.01$	3.94×10^{-3}	1.42×10^{-5}	6.54×10^{-11}

Tabla 2.1: Máximo Error Cometido por los Métodos de Euler, Heun y RK4 para Distintos Pasos de Integración con el Sistema Masa Resorte de le Ec.(2.1)

Hay infinidad de métodos de Runge Kutta explícitos para diversos órdenes (incluyendo métodos de órdenes mayores a 10). De todas maneras, los de orden 1 a 5 son los más utilizados en la práctica.

En lo referente a estabilidad, los métodos de RK tienen características similares a las de Forward Euler. Esto es, preservan la estabilidad para valores no muy grandes del paso de integración h. Por lo tanto, aunque la precisión sea buena, no podremos utilizar pasos muy grandes debido a los límites que impone la estabilidad.

Por ejemplo, para el método de Heun, podemos analizar la estabilidad sobre un sistema escalar de primer orden:

$$\dot{x} = \lambda\ x(t)$$

cuya solución analítica será estable si $\lambda < 0$. Aplicando Heun, tenemos,

$$\begin{aligned}
x_{k+1} &= x_k + \frac{h}{2}(k_1 + k_2) \\
&= x_k + \frac{h}{2}(\lambda x_k + \lambda\ (x_k + h\ \lambda x_k)) \\
&= x_k + h\ \lambda x_k + \frac{h^2\lambda^2}{2}\ x_k \\
&= (1 + h\ \lambda + \frac{h^2\lambda^2}{2})\ x_k \\
&= \lambda_d\ x_k
\end{aligned}$$

es decir, el autovalor discreto resulta $\lambda_d = 1 + h\ \lambda + \frac{h^2\lambda^2}{2}$, y la condición de estabilidad numérica es entonces

$$|1 + h\ \lambda + \frac{h^2\lambda^2}{2}| < 1$$

que se traduce en dos desigualdades:

$$\begin{cases} 1 + h\ \lambda + \frac{h^2\lambda^2}{2} & < 1 \\ 1 + h\ \lambda + \frac{h^2\lambda^2}{2} & > -1 \end{cases}$$

La primera desigualdad conduce a la condición

$$h < 2|\lambda| \tag{2.17}$$

y la segunda se cumple siempre que $\lambda < 0$. En conclusión, para un autovalor real negativo, el método de Heun dará un resultado estable cuando el paso sea suficientemente chico de manera que se cumpla la Ec.(2.17). Este resultado es idéntico al de Forward Euler para autovalores reales negativos.

2.2.2. Métodos Monopaso Implícitos

Vimos que el método de Backward Euler era un algoritmo implícito cuya principal ventaja es preservar la estabilidad de la solución numérica para cualquier paso de integración. Sin embargo, al igual que Forward Euler, realiza sólo una aproximación de primer orden.

Hay diversos métodos implícitos monopaso de orden mayor que, al igual que Backward Euler, preservan la estabilidad.

Uno de los más utilizados es la *Regla Trapezoidal*, cuya definición es la siguiente:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \frac{h}{2}\left[\mathbf{f}(\mathbf{x}_k, t_k) + \mathbf{f}(\mathbf{x}_{k+1}, t_{k+1})\right] \tag{2.18}$$

Este método implícito realiza una aproximación de segundo orden y tiene la propiedad (al menos en sistemas lineales y estacionarios) de que la solución numérica es estable si y sólo si la solución analítica es estable.

Por este motivo, la regla trapezoidal se suele utilizar a menudo para simular eficazmente sistemas muy oscilantes.

Si bien existen numerosos métodos implícitos monopaso, en la práctica no son tan utilizados ya que en general los métodos multipaso implícitos suelen ser más eficientes.

2.2.3. Algoritmos de Control de Paso

Hasta aquí consideramos siempre el paso h como un parámetro fijo que debe elegirse previo a la simulación. Sin embargo, en muchos casos es posible implementar algoritmos que cambien el paso de integración de forma automática a medida que avanza la simulación.

Los algoritmos de *control de paso* tienen por propósito mantener el error de simulación acotado para lo cual ajustan automáticamente el paso en función del error estimado. La idea es muy simple, en cada paso se hace lo siguiente:

1. Se da un paso con el método de integración elegido calculando $\mathbf{x}_{k+1}$ y cierto paso h.
2. Se estima el error cometido.
3. Si el error es mayor que la tolerancia, se disminuye el paso de integración h y se recalcula $\mathbf{x}_{k+1}$ volviendo al punto 1.
4. Si el error es menor que la tolerancia, se acepta el valor de $\mathbf{x}_{k+1}$ calculado, se incrementa el paso h y se vuelve al punto 1 para calcular $\mathbf{x}_{k+2}$.

Si bien la idea es muy sencilla, de la descripción anterior surgen varias cuestiones:

- ¿Cómo estimamos el error?
- ¿Qué ley utilizamos para recalcular el paso de integración h en cada caso?
- ¿Qué valor de paso de integración usamos inicialmente?

A continuación analizaremos en detalle dichos interrogantes.

Estima del Error

La estima del error de integración se hace utilizando dos métodos de orden diferente. Dado $\mathbf{x}_k$, calculamos, por ejemplo, $\mathbf{x}_{k+1}$ usando un método de orden 4 y luego $\tilde{\mathbf{x}}_{k+1}$ con un método de orden 5. Dado que un método de orden 5 es mucho más preciso que uno de orden 4, podemos suponer que el error en el método de orden 4 es directamente la diferencia entre ambos.

Para ahorrar cálculos, se suelen buscar métodos que compartan evaluaciones comunes en las funciones, de manera que para evaluar $\tilde{\mathbf{x}}_{k+1}$ puedan reutilizarse algunos de los cálculos hechos para calcular $\mathbf{x}_{k+1}$.

Por ejemplo, las siguientes fórmulas calculan el método de Heun (2do orden) y un método de RK de tercer orden:

$$\begin{aligned}\mathbf{k}_1 &= \mathbf{f}(\mathbf{x}_k, t_k)\\ \mathbf{k}_2 &= \mathbf{f}(\mathbf{x}_k + h\ \mathbf{k}_1, t_k + h)\\ \mathbf{k}_3 &= \mathbf{f}(\mathbf{x}_k + \frac{h}{4}\ \mathbf{k}_1 + \frac{h}{4}\ \mathbf{k}_2, t_k + \frac{h}{2})\end{aligned}$$

y luego

$$\mathbf{x}_{k+1} = x_k + \frac{h}{2}(\mathbf{k}_1 + \mathbf{k}_2) \tag{2.19}$$

$$\tilde{\mathbf{x}}_{k+1} = x_k + \frac{h}{6}(\mathbf{k}_1 + \mathbf{k}_2 + 4\mathbf{k}_3) \tag{2.20}$$

de donde puede estimarse el error como

$$\mathbf{e}_{k+1} \approx \mathbf{x}_{k+1} - \tilde{\mathbf{x}}_{k+1} = \frac{\mathbf{k}_1 + \mathbf{k}_2 - 2\mathbf{k}_3}{3}$$

Este método se denominará RK23, ya que se trata de un algoritmo de Runge–Kutta de orden 2 con estima de error de orden 3. Como puede observarse la Ec.(2.19) (Heun) y la Ec.(2.20) (RK3) comparten las etapas de cálculo de $\mathbf{k}_1$ y $\mathbf{k}_2$, lo que ahorra bastante costo computacional.

Uno de los métodos más utilizados en la práctica es un RK45 denominado *Runge–Kutta–Fehlberg*, que utiliza 6 evaluaciones de la función $\mathbf{f}$ para calcular dos soluciones, una de orden 4 y otra de orden 5.

Ajuste del Paso

Como vimos anteriormente, el error local que comete un método de orden N es proporcional a h^{N+1}, esto es,

$$err \approx c\ h^{N+1} \tag{2.21}$$

donde c es una constante desconocida. Si usamos cierto paso h y obtenemos un error err, la pregunta es, ¿que paso h_0 deberíamos usar para obtener un error tol, donde tol es la tolerancia de error que admitimos?

La respuesta a esta pregunta es lo que nos permitirá ajustar el paso tanto para disminuirlo como para aumentarlo.

Naturalmente, podemos escribir

$$tol \approx c\ h_0^{N+1}$$

Dividiendo miembro a miembro con la Ec.(2.21) resulta,

$$\frac{tol}{err} \approx \frac{h_0^{N+1}}{h^{N+1}}$$

de donde,

$$h_0 = h\ (tol/err)^{\frac{1}{N+1}}$$

que nos brinda la fórmula para ajustar el paso de acuerdo a la tolerancia admitida, al error estimado y al paso utilizado. Esta última ecuación generalmente se modifica de manera tal que

$$h_0 = 0.8\ h\ (tol/err)^{\frac{1}{N+1}} \tag{2.22}$$

para asegurar que el paso utilizado no resulte demasiado cercano al límite teórico de precisión y evitar tener que recalcular pasos.

Si simulamos con esta idea el sistema masa resorte de la Ec.(2.1), utilizando el método de Runge–Kutta–Fehlberg (RK45) que mencionamos antes, obtenemos el resultado que se muestra en la Fig.2.8.

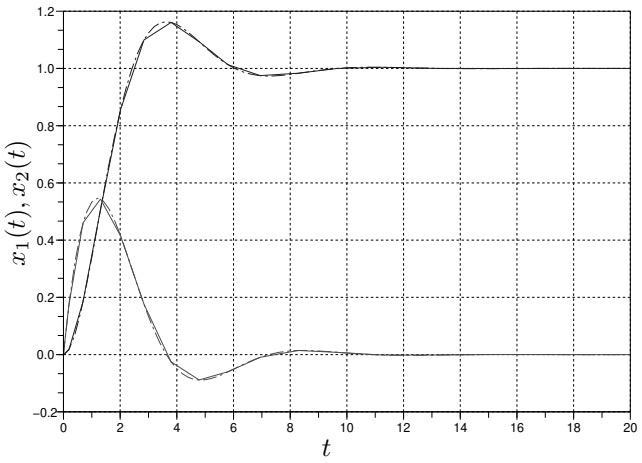

Figura 2.8: Solución de la Ecuación (2.1) del Sistema Masa Resorte con Runge–Kutta–Fehlberg (sólida) con $tol = 0.001$ y Solución Analítica (punteada)

El error es inapreciable aunque las curvas difieren ya que la solución numérica tiene muy pocos puntos (hay sólo 25 pasos) y al graficar se perciben las rectas que unen los puntos sucesivos.

La Figura 2.9 muestra la variación del paso de integración. A medida que el sistema se acerca al punto de equilibrio los pasos se hacen más grandes. Sin embargo, h no puede crecer demasiado debido a las limitaciones de estabilidad del método.

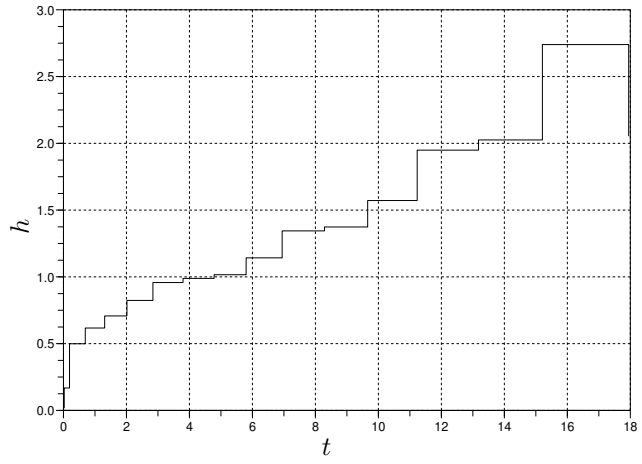

Figura 2.9: Variación del Paso de Integración en la Solución de la Fig.2.8

Algunos Parámetros de los Algoritmos de Control de Paso

Los parámetros que utilizan los algoritmos de control de paso son generalmente los siguientes:

Tolerancia relativa: Es el error que se admite, expresado como fracción de los valores de las variables. Por ejemplo, cuando se propone una tolerancia relativa de 0.001 (un valor típico), se está imponiendo que el error no debe superar el uno por mil del valor de la solución.

Tolerancia absoluta: Cuando las variables toman valores muy cercanos a cero, la tolerancia relativa impone un error demasiado pequeño que implicaría utilizar pasos exageradamente chicos. Por esto, se establece este parámetro adicional que es el mínimo valor de tolerancia a utilizar.

Máximo paso de integración: A veces, puede resultar que los dos métodos usados coincidan muy bien en el valor calculado y el error estimado sea cercano a cero. De acuerdo a la fórmula (2.22), en tal caso el valor de h resultará muy grande. Por eso, se suele limitar el paso a un valor máximo.

Mínimo paso de integración: En ciertas ocasiones, es posible que no se pueda alcanzar la tolerancia establecida sin utilizar pasos exageradamente pequeños. Esto traería como consecuencia que se trabe la simulación ya que deberían realizarse un número demasiado grande de pasos. Por esto, se suele limitar también por debajo el paso de integración. Este paso mínimo se suele también utilizar como valor inicial del paso.

Intervalo de Interpolación: Al utilizar control de paso, la solución se calcula a intervalos irregulares de tiempo. En muchas aplicaciones, es importante contar con la solución calculada en determinados instantes de tiempo generalmente equiespaciados. Para esto, se suelen utilizar algoritmos de interpolación que calculan la solución en los instantes deseados.

2.3. Métodos Multipaso

Los métodos monopaso obtienen aproximaciones de orden alto utilizando para esto varias evaluaciones de la función $\mathbf{f}$ en cada paso. Para evitar este costo computacional adicional, se han formulado diversos algoritmos que, en lugar de evaluar repetidamente la función $\mathbf{f}$ en cada paso, utilizan los valores evaluados en pasos anteriores.

Veremos a continuación algunos de estos algoritmos.

2.3.1. Métodos Multipaso Explícitos

Dentro de los métodos multipaso explícitos encontramos, entre los más utilizados, a los métodos de Adams–Bashforth (AB) y de Adams–Bashforth–Moulton (ABM).

El método de AB de 2do orden, por ejemplo, está dado por la fórmula:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \frac{h}{2}\left(3\mathbf{f}_k - \mathbf{f}_{k-1}\right) \tag{2.23}$$

donde definimos

$$\mathbf{f}_k \triangleq \mathbf{f}(\mathbf{x}_k, t_k)$$

El método de AB de 4to orden, en tanto, es el que sigue:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \frac{h}{24}\left(55\mathbf{f}_k - 59\mathbf{f}_{k-1} + 37\mathbf{f}_{k-2} - 9\mathbf{f}_{k-3}\right) \tag{2.24}$$

Como vemos, estos métodos requieren realizar una única evaluación de la función $\mathbf{f}$ en cada paso. El orden alto de aproximación se logra utilizando la información de las evaluaciones anteriores de dicha función.

Un problema de los métodos multipaso es el arranque. La expresión de la Ec.(2.23) sólo puede utilizarse para calcular la solución a partir de $k = 2$. Para el primer pasos no está definido el valor de $\mathbf{x}_{-1}$ requerido por la fórmula. Similarmente, el algoritmo de AB4 de la Ec.(2.24) sólo puede usarse a partir de $k = 4$ por la misma razón.

Por este motivo, los primeros pasos de los métodos multipaso deben realizarse con algún método monopaso de, al menos, el mismo orden de precisión. Por ejemplo, para el método de AB4 de la Ec.(2.24), deberían darse los tres primeros pasos con Runge–Kutta de orden 4.

Veamos por ejemplo, el resultado de utilizar AB2 para simular el sistema masa resorte del ejemplo. Usaremos como valores iniciales $\mathbf{x}_0 = [0\ 0]$ y $\mathbf{x}_1 = [0.005\ 0095]$, este último calculado previamente con el método de Heun. Luego,

$$\mathbf{x}_0 = \begin{bmatrix}0\\0\end{bmatrix} \Rightarrow \mathbf{f}(\mathbf{x}_0, t_0) = \begin{bmatrix}x_2\\1 - x_1 - x_2\end{bmatrix} = \begin{bmatrix}0\\1\end{bmatrix}$$
$$\mathbf{x}_1 = \begin{bmatrix}0.005\\0.095\end{bmatrix} \Rightarrow \mathbf{f}(\mathbf{x}_1, t_1) = \begin{bmatrix}x_2\\1 - x_1 - x_2\end{bmatrix} = \begin{bmatrix}0.095\\0.9\end{bmatrix}$$
$$\mathbf{x}_2 = \mathbf{x}_1 + \frac{h}{2}(3\ \mathbf{f}_1 - \mathbf{f}_0) = \begin{bmatrix}0.005\\0.095\end{bmatrix} + 0.05\ \left(3\ \begin{bmatrix}0.095\\0.9\end{bmatrix} - \begin{bmatrix}0\\1\end{bmatrix}\right) = \begin{bmatrix}0.01925\\0.18\end{bmatrix}$$

Otro problema de los métodos de Adams–Bashforth es que resultan estables sólo para valores muy pequeños del paso de integración. Esta característica se mejora utilizando los métodos de Adams–Bashforth–Moulton (ABM).

El método de ABM de tercer orden, por ejemplo, está dado por la siguiente fórmula

$$\mathbf{x}^P_{k+1} = \mathbf{x_k} + \frac{h}{12}(23\mathbf{f}_k - 16\mathbf{f}_{k-1} + 5\mathbf{f}_{k-2})$$
$$\mathbf{x}_{k+1} = \mathbf{x_k} + \frac{h}{12}(5\mathbf{f}^P_{k+1} + 8\mathbf{f}_k - \mathbf{f}_{k-1})$$

donde $\mathbf{f}^P_{k+1} = \mathbf{f}(x^P_{k+1}, t_{k+1})$.

Este método, denominado *predictor–corrector*, tiene mejor estabilidad que AB3 a costa de utilizar una evaluación extra de la función $\mathbf{f}$.

2.3.2. Métodos Multipaso Implícitos

Para obtener soluciones numéricas que preserven la estabilidad independientemente del paso de integración, al igual que en los métodos monopaso, hay que recurrir a los métodos implícitos.

Los métodos implícitos más utilizados en la práctica son los denominados *Backward Difference Formulae* (BDF), que son de tipo multipaso. Por ejemplo, el siguiente es el método de BDF de orden 3:

$$\mathbf{x}_{k+1} = \frac{18}{11}\mathbf{x}_k - \frac{9}{11}\mathbf{x}_{k-1} + \frac{2}{11}\mathbf{x}_{k-2} + \frac{6}{11} \cdot h \cdot \mathbf{f}_{k+1} \tag{2.25}$$

Este método tiene prácticamente el mismo costo computacional que Backward Euler, ya que la ecuación a resolver es muy similar. Sin embargo, BDF3 es de tercer orden.

Existen métodos de BDF de hasta orden 8, si bien los más utilizados son los de orden 1 (Backward Euler) hasta 5.

2.3.3. Control de Paso

En los métodos multipaso se puede también controlar el paso de integración de manera similar a la de los métodos monopaso. Sin embargo, las fórmulas de los métodos multipaso son sólo válidas asumiendo paso constante (ya que usan valores anteriores de la solución). Por lo tanto, para cambiar el paso en un método multipaso hay que interpolar los últimos valores de la solución a intervalos regulares correspondientes al nuevo paso de integración. En consecuencia, cambiar el paso tiene un costo adicional en este caso.

Por este motivo, los algoritmos de control de paso en métodos multipaso sólo cambian el paso cuando el cambio de paso es suficientemente grande.

2.4. Algunos Casos Problemáticos

Muchos sistemas dinámicos en la práctica tienen modelos en sistemas de ecuaciones diferenciales que resultan problemáticos para los distintos métodos de integración numérica.

Veremos a continuación distintos casos que nos obligarán a utilizar ciertas técnicas particulares.

2.4.1. Sistemas Stiff

Consideremos nuevamente el sistema masa resorte de la Ec.(2.1), pero ahora supongamos que el coeficiente de rozamiento es $b = 100$.

Con este nuevo valor para el parámetro, el sistema tiene matriz de evolución

$$A = \begin{bmatrix} 0 & 1 \\ -1 & -100 \end{bmatrix}$$

cuyos autovalores son $\lambda_1 \approx -0.01$ y $\lambda_2 \approx -100$.

La Figura 2.10 muestra la solución analítica (sólo se grafica x_2 en este caso). En la curva de la izquierda se ve la trayectoria completa, que se asemeja a una evolución de primer orden (correspondiente al autovalor lento $\lambda_1 \approx -0.01$. La curva de la derecha muestra en tanto el inicio de la trayectoria, que también se asemeja a una evolución de primer orden, pero mucho más rápida (correspondiente al autovalor $\lambda_2 \approx -100$).

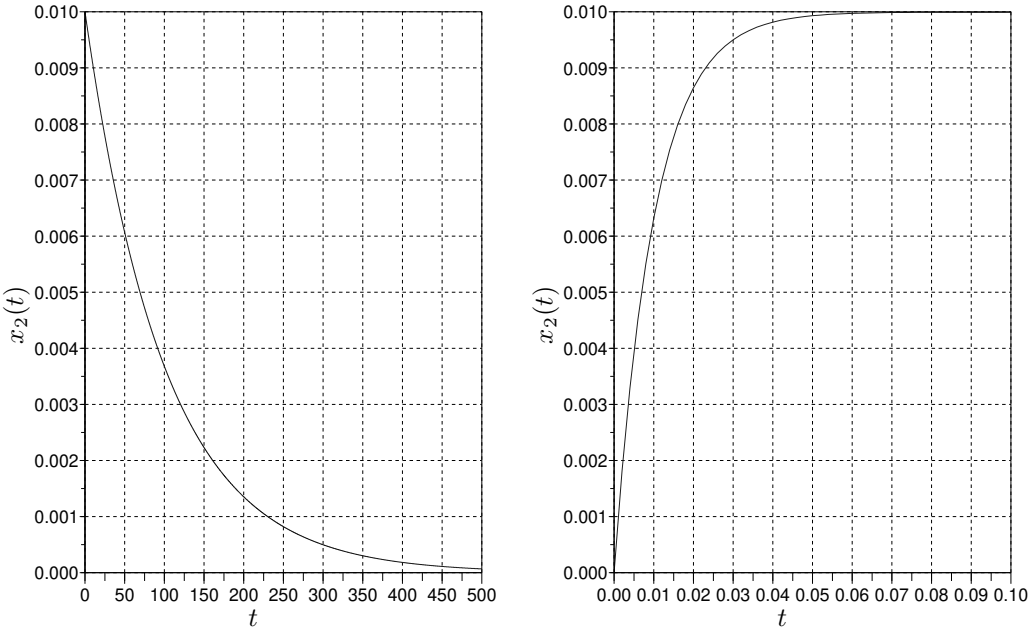

Figura 2.10: Solución Analítica para x_2 en el Sistema Masa Resorte con $b = 100$.

Evidentemente este sistema posee una dinámica lenta y una rápida. Los sistemas que tienen estas características se denominan *sistemas stiff* o rígidos. Como veremos, esto nos complicará a la hora de simularlo.

Es claro que si queremos simular este sistema usando algún método de integración, debemos hacerlo hasta un tiempo final de al menos $t_f = 500$. Además, si utilizamos el método de Euler, por ejemplo, deberemos usar un paso de integración $h < 0.02$ ya que de otra manera tendremos un resultado inestable según la Ec.(2.13). Esto implicará que haremos al menos $500/0.2 = 25000$ pasos para completar la simulación, lo que es absurdo dado lo simple del sistema.

La primer idea es entonces usar algún método de paso variable, como por ejemplo, el RK45 que vimos antes. El resultado en este caso no es mucho mejor, ya que el método necesita 13604 pasos para completar la simulación. La Figura 2.11 muestra la evolución del paso de integración en dicha simulación (se muestra sólo lo que ocurre hasta $t = 10$, luego sigue constante).

Aunque como puede verse en la solución analítica a la izquierda de la Fig.2.10 la dinámica rápida prácticamente desaparece a partir de $t = 0.1$, el método de RK45 nunca puede aumentar el paso de

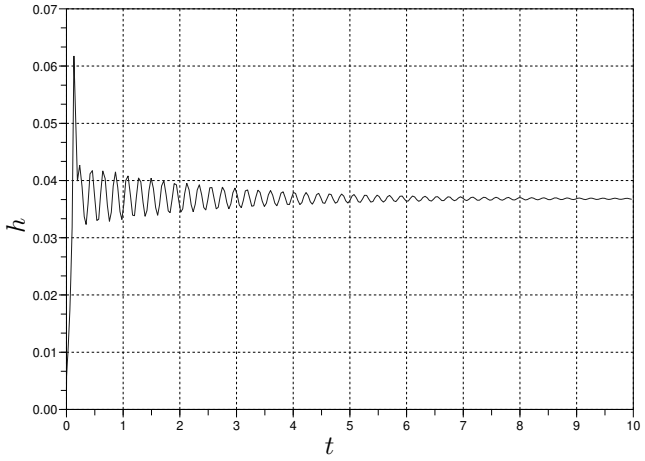

Figura 2.11: Paso de Integración h en la Simulación de un Sistema Stiff con RK45.

integración mucho más allá de $h \approx 0.04$. La explicación para esto es sencilla: debido al autovalor rápido $\lambda_2 \approx -100$ con pasos de integración mayores el método de RK4 se torna inestable por lo que el algoritmo obtiene un error grande que lo obliga a disminuir nuevamente el paso.

Este efecto sólo puede evitarse si usamos algoritmos que no se inestabilicen, esto es, si usamos alguno de los métodos implícitos que presentamos.

Utilizando por ejemplo, un método implícito de RK de cuarto orden con control de error de orden 5, con la misma tolerancia que en RK45 (0.001), la simulación se completa en 33 sólo pasos y se obtiene una precisión idéntica a la anterior. La Fig.2.12 muestra la evolución del paso de integración en este caso. El mismo está sólo limitado por la precisión y nunca por la estabilidad.

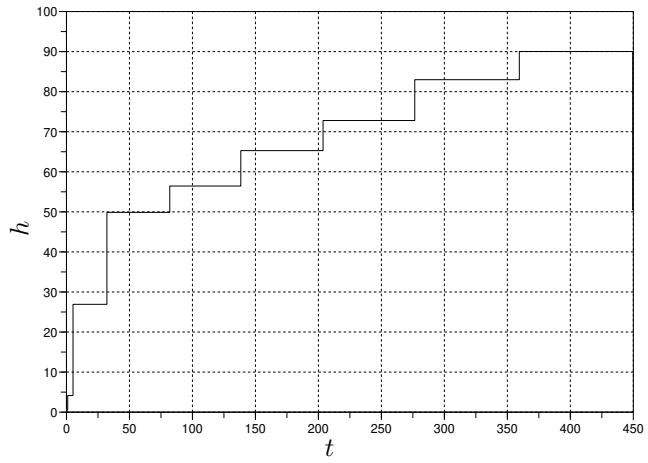

Figura 2.12: Paso de Integración h en la Simulación de un Sistema Stiff con un método implícito monopaso.

Si bien una computadora moderna puede realizar 13000 pasos en un sistema sencillo en un tiempo casi despreciable, en general los casos stiff son mucho más severos que lo visto aquí. En el simple ejemplo del masa resorte, si hubiéramos tenido $b = 10000$, RK45 necesitaría más de 13 millones de pasos para completar la simulación.

En la práctica, es casi imposible simular sistemas stiff sin recurrir a algoritmos implícitos.

2.4.2. Sistemas Marginalmente Estables

Muchos sistemas de la práctica tienen nulo o escaso amortiguamiento. En tal caso, las soluciones muestran oscilaciones sostenidas o bien que se amortiguan muy lentamente. Esta característica puede observarse en el ejemplo del sistema Masa Resorte haciendo $b = 0$ o tomando b muy peqeño.

Los sistemas con estas propiedades se denominan *marginalmente estables* y también revisten algunos problemas para la simulación numérica.

En el caso del sistema masa resorte con $b = 0$ tenemos un par de autovalores imaginarios puros conjugados $\lambda_{1,2} = \pm j$, por lo que las soluciones serán periódicas.

La Figura 2.13 muestra la solución analítica y el resultado de simular el sistema con Forward y Backward Euler usando $h = 0.1$. Como puede observarse en la misma, la solución numérica de Forward Euler es inestable, mientras que la de Backward Euler es asintóticamente estable.

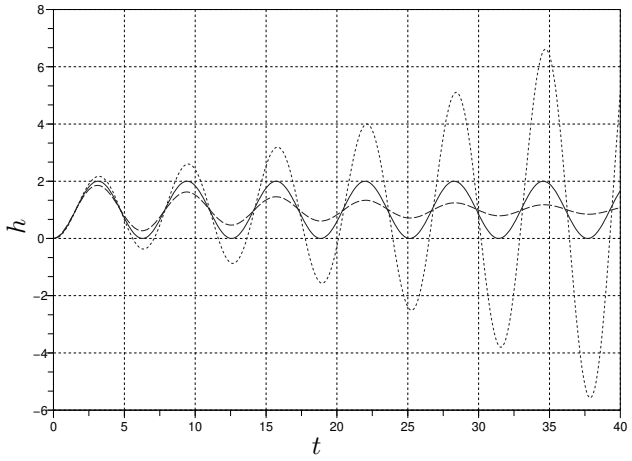

Figura 2.13: Sistema Masa Resorte con $b = 0$: Solución analítica (sólida), Forward Euler (punteada) y Backward Euler (línea discontinua).

Ninguna de las dos soluciones respeta la característica marginalmente estable original por lo que los resultados son cualitativamente incorrectos. Para obtener buenos resultados hay que utilizar métodos denominados *F–Estables* que preservan también la estabilidad marginal. Uno de estos métodos es la Regla Trapezoidal que presentamos en la Ec.(2.18).

2.4.3. Sistemas con Discontinuidades

La siguiente ecuación es un modelo muy simple de una pelotita que rebota contra el piso:

$$\begin{aligned} \dot{x}(t) &= v(t) \\ \dot{v}(t) &= -g - s_w(t) \cdot \frac{1}{m}(k \cdot x(t) + b \cdot v(t)) \end{aligned}$$

donde

$$s_w = \begin{cases} 0 & \text{si } x(t) > 0 \\ 1 & \text{en otro caso} \end{cases}$$

En este modelo, estamos considerando que cuando $x(t) > 0$ la pelotita está en el aire y responde a una ecuación de caída libre ($s_w = 0$). Cuando la pelotita entra en contacto con el piso, en cambio, sigue un modelo *masa–resorte–amortiguador*, lo que produce el rebote.

En este caso, consideramos parámetros $m = 1$, $b = 30$, $k = 1 \times 10^6$ y $g = 9.81$.

Decidimos simular este sistema utilizando el método de RK4, durante 5 segundos, a partir de la condición inicial $x(0) = 1$, $v(0) = 0$. Esta situación corresponde a soltar la pelotita desde 1 metro de distancia del piso. Comenzamos a tener resultados *decentes* con un paso de integración $h = 0.002$. En la Fig 2.14 se muestran los resultados de la simulación con pasos $h = 0.002$, $h = 0.001$ y $h = 0.0005$.

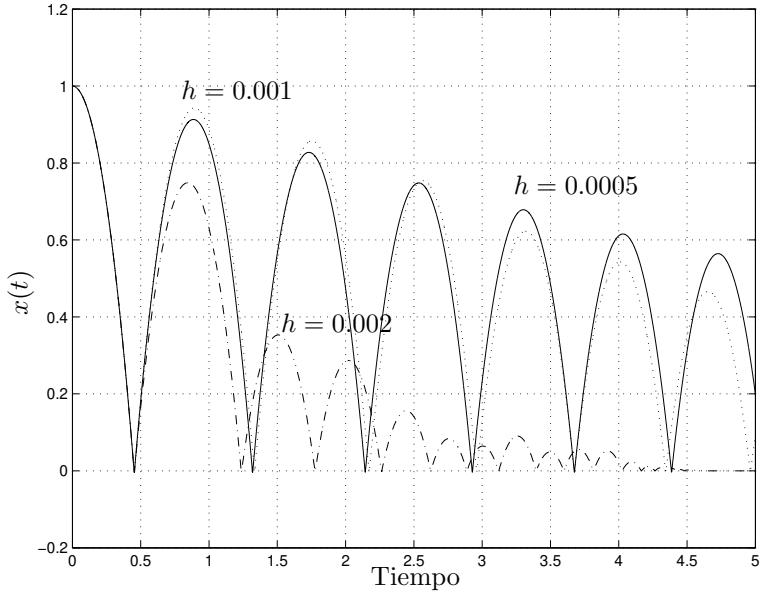

Figura 2.14: Simulación con RK4 de la pelotita rebotando.

Evidentemente, no podemos confiar en los resultados de esta simulación. La del paso más pequeño parecería más precisa, pero no hay manera de asegurarlo por el momento.

Mirando el comienzo de la simulación, no hay error apreciable hasta el primer pique. Evidentemente, el problema tiene que ver con la discontinuidad.

Veamos que ocurre en tanto si utilizamos un método de paso variable. En la Fig 2.15 se pueden apreciar los resultados con RK23 utilizando las tolerancias relativas 10^{-3}, 10^{-4} y 10^{-5}. De nuevo, los resultados son desalentadores. Si bien algunos piques se resuelven *bien* (al menos el método hace lo mismo con las tres tolerancias), en otros piques el error se torna inaceptable.

Tanto al usar paso fijo como al usar paso variable el problema tiene que ver con que se dieron pasos que atravesaron la discontinuidad. Es decir, en t_k teníamos $x(t_k) > 0$ y en t_{k+1} resultó $x(t_{k+1}) < 0$. Como la función $\mathbf{f}$ resulta discontinua entre t_k y t_{k+1}, se produce un error muy grande.

La manera de corregir estos errores es evitando la situación anterior. Para esto, si nos encontramos con que $x(t_{k+1}) < 0$ hay que volver atrás y buscar el punto donde se produce la discontinuidad, esto es, el valor de $\tilde{t}$ para el cual $x(\tilde{t}) = 0$. Una vez detectado este punto (para lo cual puede ser necesario iterar), se debe realizar un paso de valor $h = \tilde{t} - t_k$ (para llegar a dicho punto) y desde este nuevo punto reiniciar la simulación.

Los problemas de detección y tratamiento de discontinuidades son cruciales para simular correctamente sistemas de electrónica de potencia. En estos sistemas, generalmente, las discontinuidades ocurren a frecuencias muy grandes por lo que el uso de algoritmos eficientes para tratar con las discontinuides es imprescindible.

2.5. Simulación de DAEs

Como vimos en el Capítulo 1, lo primero que obtenemos durante el proceso de modelado por primeros principios es un conjunto de relaciones constitutivas y estructurales que conforman un sistema de ecuaciones diferenciales algebraicas que en su expresión general tendrán la forma de la Ec.(1.10). Recordando que

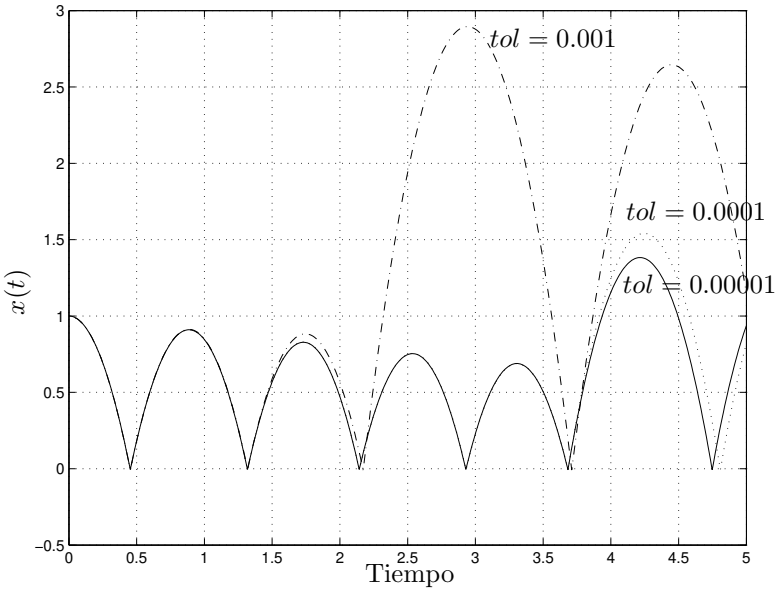

Figura 2.15: Simulación con RK23 de la pelotita rebotando.

para simular necesitamos conocer las trayectorias de entradas, esta ecuación tomará la siguiente forma:

$$\mathbf{f}(\mathbf{x}(t), \dot{\mathbf{x}}(t), \mathbf{a}(t), t) = 0. \tag{2.26}$$

Recordemos que $\mathbf{x} \triangleq [x_1, \ldots, x_n]^T$ es el vector de estados y $\mathbf{a} \triangleq [a_1, \ldots, a_r]^T$ es el vector de variables algebraicas.

Para poder simular un sistema como este utilizando alguno de los métodos numéricos que vimos, deberíamos implementar una iteración que a cada paso calcule el valor de $\dot{\mathbf{x}}(t)$, para lo que se deberá *despejar* de la Ec.(2.26) no sólo dicho vector sino también el vector de variables algebraicas $\mathbf{a}(t)$ (es un sistema de $n + r$ ecuaciones). Esto podría hacerse con la iteración de Newton.

Si además el método de integración a utilizar fuera implícito (como Backward Euler, por ejemplo) deberíamos agregar otra iteración de Newton por fuera para obtener $\mathbf{x}_{k+1}$. En definitiva, tendríamos un algoritmo de simulación del estilo

for $k = 0 : N - 1$ **do**
  Inicializar $\mathbf{x}_{k+1}^0$
  repeat
    Inicializar $\dot{\mathbf{x}}_{k+1}^0$
    repeat
      Calcular $\dot{\mathbf{x}}_{k+1}^{j+1}$
    until $|\dot{\mathbf{x}}_{k+1}^{j+1} - \dot{\mathbf{x}}_{k+1}^{j}| < \text{tol}$
    Calcular $\mathbf{x}_{k+1}^{l+1}$
  until $|\mathbf{x}_{k+1}^{l+1} - \mathbf{x}_{k+1}^{l}| < \text{tol}$
end for

Esto es, tenemos tres iteraciones anidadas, una para avanzar el tiempo (for k), y dos iteraciones de Newton en el interior, cada una con sus respectivos cálculos de Jacobiano e inversión de la matriz resultante. Por lo tanto, el costo asociado a este algoritmo podría ser prohibitivo si el tamaño del problema fuera muy grande.

Si bien en el capítulo siguiente veremos que en muchos casos podremos convertir la DAE en una ODE simplificando notablemente el problema, hay otra solución que puede seguirse.

De la fórmula de Backward Euler de la Ec.(2.6), podemos reescribir

$$\mathbf{x}_{k+1} = \mathbf{x}_k + h\ \mathbf{f}(\mathbf{x}_{k+1}, t_k) = \mathbf{x}_k + h\ \dot{\mathbf{x}}_{k+1}$$

y despejar

$$\dot{\mathbf{x}}_{k+1} = \frac{\mathbf{x}_{k+1} - \mathbf{x}_k}{h}$$

y finalmente reemplazar en la DAE (2.26):

$$\mathbf{f}(\mathbf{x}_k, \frac{\mathbf{x}_{k+1} - \mathbf{x}_k}{h}, \mathbf{a}_k, t_k) = 0. \tag{2.27}$$

En esta última ecuación la incógnita es directamente $\mathbf{x}_{k+1}$, que puede calcularse en cada paso con la iteración de Newton sin necesidad de tener dos iteraciones anidadas.

Esta misma idea se puede utilizar con otros métodos implícitos (particularmente con los métodos de BDF), y es una alternativa muy utilizada en la práctica.

2.6. Implementación de los Algoritmos en Software

Hay muchas herramientas de software que implementan los distintos métodos de integración numérica. Hay además versiones de los algoritmos ya implementadas y disponibles en distintos lenguajes de programación (sobre todo en lenguaje C), incluyendo métodos de todo tipo: explícitos, implícitos, con control de paso, para DAEs, etc. Entre los códigos abiertos disponibles más populares que implementas algoritmos de integración numérica de paso variable se encuentran DOPRI (métodos explícitos de Runge Kutta), DASSL (implícitos multipaso, para DAEs), y los CVODE, que incluyen métodos implícitos (BDF) y explícitos (AM) multipaso. Estos códigos se denominan *solvers* y son utilizados por diversas herramientas de modelado y simulación.

Pueden además implementarse fácilmente los distintos algoritmos que vimos en este capítulo, sobre todo los explícitos, ya que los métodos implícitos requieren implementar además la iteración de Newton que implica una dificultad extra. Por ejemplo, utilizando el lenguaje de Matlab u Octave, el Código 2.1 implementa el método de Forward Euler.

Código 2.1: Función de Matlab/Octave que implementa método de Forward Euler

```
function [t,x]=feuler(f,x0,h,t0,tf)
  t=[t0:h:tf];
  x=zeros(length(x0),length(t));
  x(:,1)=x0;
  for k=1:length(t)-1
    x(:,k+1)=x(:,k)+h*f(x(:,k),t(k));
  end
end
```

Notar que este algoritmo usa como entradas la función $\mathbf{f}$ que calcula la derivada del estado, la condición inicial $\mathbf{x}_0$, el paso de integración h, el tiempo inicial t_0 y el tiempo final de simulación t_f. Como resultado, brinda un vector fila t con los instantes de tiempo donde se realizaron los pasos y una matriz x que contiene en cada columna el valor de la solución en el instante t_k.

Si queremos simular utilizando este algoritmo un modelo Masa-Resorte como el de Ec.(2.1), deberemos construir la función $\mathbf{f}(\mathbf{x}, t)$ que calcula la derivada del estado $\dot{\mathbf{x}}$ según dicho modelo como se muestra en el Código 2.2.

Código 2.2: Función de Matlab/Octave con el modelo de la Ec.(2.1)

```
function dx=masares(x,t)
  m=1; b=1; k=1; %parametros
  F=1;           %entrada
  der_x1=x(2);
  der_x2=-k/m*x(1)-b/m*x(2)+F;
  dx=[der_x1; der_x2]; %vector de derivadas
end
```

Luego, podemos realizar la simulación y graficar los resultados invocando el Código 2.3.

Código 2.3: Comandos de Matlab/Octave para simular el modelo de la Ec.(2.1)

```
[t,x]=feuler(@masares,[0;0],0.1,0,10);
plot (t,x)
```

Matlab y Octave tienen también herramientas que resuelven ecuaciones algebraicas basadas en la iteración de Newton. Particularmente, el comando `fsolve` permite resolver un problema de tipo $\mathbf{F}(\mathbf{z}) = 0$. Utilizando este comando entonces, un método implícito como el de Backward Euler se puede implementar de manera trivial usando el Código 2.4:

Código 2.4: Función de Matlab/Octave que implementa método de Backward Euler

```
function [t,x]=beuler(f,x0,h,t0,tf)
  t=[t0:h:tf];
  x=zeros(length(x0),length(t));
  x(:,1)=x0;
  for k=1:length(t)-1
    F=@(z)(z-x(:,k)-h*f(z,t(k))); %funcion que debe ser 0
    x(:,k+1)=fsolve(F,x(:,k));
  end
end
```

Además, tanto Matlab como Octave tienen implementados distintos métodos numéricos de paso variable, incluyendo `ode23` y `ode45` (métodos explícitos de Runge-Kutta) y `ode15s` que es un método multipaso implícito similar a los algoritmos de BDF. En Octave también hay una implementación de DASSL que permite simular directamente DAEs.

Las herramientas de modelado y simulación basadas en Modelica, como Dymola, Wolfram SystemModeler y OpenModelica, cuentan con una amplia variedad de solvers y tienen además la ventaja de que realizan un preproceso sobre las ecuaciones (veremos parte de ellas en el Capítulo 3) de manera que, particularmente en las DAEs, los modelos se pueden escribir de manera más sencillas y las simulaciones resultan más eficientes.

2.7. Bibliografía Complementaria

Los contenidos de este apunte están principalmente basados en las notas de clases de la materia de doctorado *Simulación de Sistemas Continuos* [Kof06], que a su vez están basadas principalmente en el libro [CK06] (disponible en la biblioteca de la Facultad y de la Escuela de Ingeniería Electrónica).

Por otro lado, los libros [HNW00] y [HW91] son referencia obligada en la disciplina para quienes tengan interés en las propiedades más matemáticas de los algoritmos.

Finalmente, una colección muy importante de algoritmos y programas para implementar diversos métodos numéricos (no sólo para ecuaciones diferenciales, sino para problemas más generales) puede encontrarse en el libro clásico [PTVF07].

2.8. Problemas Propuestos

[P2.1] Forward y Backward Euler

Dada la Ecuación Diferencial Ordinaria

$$\dot{x}(t) = -3\ x(t) + 3 \tag{P2.1a}$$

con condición inicial $x(0) = 0$, se pide,

1. Obtener y graficar la solución analítica.

2. Resolverla mediante el método de Forward Euler utilizando pasos $h = 0.1$ y $h = 1$. (Calcular los 5 primeros valores en cada caso).

3. Resolverla ahora mediante Backward Euler utilizando los mismos pasos que en el punto anterior.

4. Comparar el error cometido tras el primer paso de integración utilizando los dos valores de h en cada método. Explicar los resultados.

5. Analizar la estabilidad numérica de las soluciones obtenidas con ambos valores de h y con ambos métodos.

[P2.2] Método de Heun

Utilizando el Método de Heun, definido en las Ecs.(2.15)–(2.16), se pide

1. Resolver la Ec.(P2.1a) a partir de la condición inicial $x(0) = 0$ utilizando pasos $h = 0.1$ y $h = 1$. (Calcular los 5 primeros valores en cada caso).

2. Comparar el error cometido tras el primer paso de integración utilizando los dos valores de h. Explicar las diferencias o coincidencias con los métodos de Euler del Problema P2.1.

3. Analizar la estabilidad numérica con ambos valores de h y comparar con lo que se obtiene usando Forward y Backward Euler.

4. Programar en Octave o Matlab una función que implemente este método, similar a la del Código 2.1 y verificar los resultados obtenidos en los puntos 1 y 3.

[P2.3] Regla del Punto Medio

La *Regla del Punto Medio* es un método de integración de segundo orden, que tiene versiones implícitas y explícitas. Una versión de este método aplicada al sistema

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t) \tag{P2.3a}$$

tiene la siguiente definición:

$$\begin{aligned} \mathbf{k}_1 &= \mathbf{f}(\mathbf{x}_k, t_k) \\ \mathbf{k}_2 &= \mathbf{f}(\mathbf{x}_k + \frac{h}{2}\,\mathbf{k}_1, t_k + \frac{h}{2}) \\ \mathbf{x}_{k+1} &= \mathbf{x}_k + h\;\mathbf{k}_2 \end{aligned} \tag{P2.3b}$$

Se pide entonces

1. Clasificar este método según sea

   a) Implícito/Explícito

   b) Monopaso/Multipaso

2. Utilizando este método con un paso $h = 0.1$, obtener la solución hasta $t_f = 0.5$ del sistema de la Ec.(P2.1a) a partir de la condición inicial $x(0) = 0$.

3. Determinar para que valores del paso de integración h la solución numérica del ejemplo de la Ec.(P2.1a) resulta estable.

4. Programar en Octave o Matlab una función que implemente este método, similar a la del Código 2.1 y verificar los resultados obtenidos en los puntos anteriores.

[P2.4] Regla Trapezoidal

Repetir el Problema P2.3 utilizando la Regla Trapezoidal, dada por la Ec.(2.18), en lugar de la Regla del Punto Medio.

[P2.5] Método de Adams-Bashforth 3

Dado el sistema de la Ec.(P2.3a), el método de AB3 lo aproxima con la fórmula

$$\mathbf{x}_{k+1} = x_k + \frac{h}{12}\left(23\ \mathbf{f}(\mathbf{x}_k, t_k) - 16\ \mathbf{f}(\mathbf{x}_{k-1}, t_{k-1}) + 5\ \mathbf{f}(\mathbf{x}_{k-2}, t_{k-2})\right) \tag{P2.5a}$$

Se pide entonces

1. Clasificar este método según sea
   - *a*) Implícito/Explícito
   - *b*) Monopaso/Multipaso
2. Dado el sistema
   $$\dot{x}(t) = -2\ x(t) + 4 \tag{P2.5b}$$
   con la condición inicial $x(0) = 0$, se pide
   - *a*) Obtener la solución numérica con el método de AB3 hasta $t_f = 0.6$ con un paso $h = 0.2$. Si necesita condiciones iniciales adicionales, utilice la solución analítica:
     $$x(t) = 2 \cdot (1 - e^{-2 \cdot t})$$
   - *b*) Comparar la solución numérica y la analítica.
   - *c*) Obtener la solución numérica con el método de AB2 hasta $t_f = 0.06$ con un paso $h = 0.02$. Si necesita condiciones iniciales adicionales, utilice nuevamente la solución analítica.
   - *d*) Comparar los errores cometidos tras el primer paso con los dos pasos de integración y concluya sobre el orden del método.

Capítulo 3

Causalización de Sistemas de DAEs

En el Capítulo 1 vimos que las Ecuaciones Diferenciales Algebraicas son la forma más simple en la que se puede obtener un modelo matemático. Sin embargo, este tipo de representación no es la más adecuada para simular porque en cada paso de integración debemos resolver un sistema de ecuaciones algebraicas que puede tener gran tamaño. Tampoco es la más adecuada para realizar análisis de propiedades, porque para esto necesitamos que el sistema esté en su forma de Ecuaciones de Estado o en forma de Función Transferencia si fuera lineal y quisiéramos analizar la respuesta en frecuencia.

En este capítulo, que está basado principalmente en el Capítulo 7 del libro *Continuous System Simulation* [CK06], vamos a ver versiones simplificadas de los procedimientos que permiten en ciertos casos transformar sistemas de Ecuaciones Diferenciales Algebraicas en sistemas de Ecuaciones Diferenciales Ordinarias y en otros casos permiten al menos reducir el número de variables que intervienen en las iteraciones para calcular la derivada del vector de estados $\dot{\mathbf{x}}(t)$.

Estos procedimientos constituyen una parte esencial de los *compiladores* de los lenguajes de modelado como Modelica.

3.1. Ordenamiento y Causalización de Ecuaciones

3.1.1. Modelos Implícitos y Explícitos

Consideremos nuevamente el circuito RLC del Capítulo 1, con el sistema de DAEs que definen sus relaciones constitutivas y estructurales, donde $v(t)$ representa una función del tiempo conocida.

$$u_R(t) - R\ i_R(t) = 0 \tag{3.1a}$$
$$\dot{q}(t) - i_C(t) = 0 \tag{3.1b}$$
$$q(t) - C\ u_C(t) = 0 \tag{3.1c}$$
$$\dot{\phi}(t) - u_L(t) = 0 \tag{3.1d}$$
$$\phi(t) - L\ i_L(t) = 0 \tag{3.1e}$$
$$u_S(t) - v(t) = 0 \tag{3.1f}$$
$$u_L(t) + u_R(t) + u_C(t) - u_S(t) = 0 \tag{3.1g}$$
$$i_L(t) - i_R(t) = 0 \tag{3.1h}$$
$$i_C(t) - i_L(t) = 0 \tag{3.1i}$$
$$i_S(t) - i_R(t) = 0 \tag{3.1j}$$

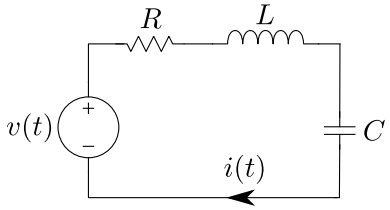

Figura 3.1: Circuito RLC Serie

Si quisiéramos simular este sistema utilizando el método de Forward Euler sin realizar ningún tipo de transformación sobre el sistema de DAEs, tendríamos que implementar un algoritmo iterativo que sea capaz de *despejar* las derivadas de los estados (y las variables algebraicas) de la Ec.(3.1), lo que constituye

un sistema de 10 ecuaciones con 10 incógnitas. Esto además debería realizarse cada vez que queremos obtener el valor de dichas derivadas (una vez en cada paso si usamos Forward Euler).

El Código 3.3 contiene dos funciones de Matlab/Octave que brindan una manera de realizar esto:

Código 3.1: Funciones de Matlab/Octave con el modelo de la Ec.(3.1)

```
function dx=circuit_ode(x,t)
  F = @(z) circuit_dae(x,z(1:2),z(3:10),t);
  z0=zeros(10,1);
  z=fsolve(F,z0);
  dx=z(1:2);
end

function res=circuit_dae(x,dx,a,t)
  R=1; C=1; L =1; %parametros
  q=x(1); phi=x(2); %estados
  der_q=dx(1); der_phi=dx(2); %derivadas de estados
  uR=a(1); iR=a(2); iC=a(3); uC=a(4); uL=a(5); iL=a(6); uS=a(7); iS=a
      (8);
  res=zeros(10,1);  %residuo (debera ser cero)
  res(1) = uR - R * iR;
  res(2) = der_q - iC;
  res(3) = q - C * uC;
  res(4) = der_phi - uL;
  res(5) = phi - L * iL;
  res(6) = uS - sin(t);
  res(7) = uL + uR + uC - uS;
  res(8) = iL - iR;
  res(9) = iC - iL;
  res(10) = iS - iR;
end
```

La función `circuit_dae` calcula el lado izquierdo de las ecuaciones diferenciales algebraicas, es decir, las expresiones que deben ser nulas. Estas expresiones dan como resultado un residuo `res`. Esta función es utilizada por la función `circuit_ode` que calcula la derivada del estado de manera que efectivamente dicho residuo sea cero. Esto último lo hace utilizando la función `fzero` que calcula las raíces de una ecuación algebraica.

Podemos entonces simular esta DAE directamente utilizando el método de Forward Euler del Código 2.1 invocando

Código 3.2: Comando de Matlab/Octave para simular la Ec.(3.1)

```
[t,x]=feuler(@circuit_ode,[0;0],0.1,0,10);
plot (t,x)
```

Si bien esto funciona, es extremadamente costoso desde el punto de vista computacional. Nuestro objetivo entonces será transformar este sistema de DAEs en algo equivalente al sistema de Ecuaciones de Estado dado por

$$\begin{aligned} \dot{q}(t) &= \frac{1}{L}\,\phi(t) \\ \dot{\phi}(t) &= -\frac{1}{C}\,q(t) - \frac{R}{L}\,\phi(t) + v(t), \end{aligned} \tag{3.2}$$

es decir, buscaremos tener expresiones explícitas para las derivadas de los estados en función de los propios estados y de las entradas.

Una opción para obtener expresiones explícitas está dada por el siguiente sistema de ecuaciones

$$u_C(t) := \frac{q(t)}{C} \tag{3.3a}$$
$$i_L(t) := \frac{\phi(t)}{L} \tag{3.3b}$$
$$u_S(t) := v(t) \tag{3.3c}$$
$$i_R(t) := i_L(t) \tag{3.3d}$$
$$i_C(t) := i_L(t) \tag{3.3e}$$
$$i_S(t) := i_R(t) \tag{3.3f}$$
$$u_R(t) := R\, i_R(t) \tag{3.3g}$$
$$u_L(t) := u_S(t) - u_R(t) - u_C(t) \tag{3.3h}$$
$$\dot{q}(t) := i_C(t) \tag{3.3i}$$
$$\dot{\phi}(t) := u_L(t) \tag{3.3j}$$

donde el símbolo := indica una *asignación* (no una igualdad). Puede verse fácilmente que la secuencia de asignaciones dada por las Ecs.(3.3a)-(3.3j) tiene las siguientes propiedades:

- En el lado izquierdo de cada asignación se calcula una variable que no fue calculada previamente.
- En el lado derecho de cada asignación se utilizan estados o bien variables algebraicas que ya fueron calculadas previamente.

Esto implica que la secuencia de asignaciones se puede programar en cualquier lenguaje de programación de manera muy simple y conduce a un código que permite evaluar efectivamente las derivadas de los estados en función de los estados y las entradas como en la Ec.(3.2). De hecho, si utilizamos Matlab u Octave, podemos traducir la secuencia de asignaciones en la siguiente función:

Código 3.3: Función de Matlab/Octave con el modelo de la Ec.(3.3)

```
function dx=circuit(x,t)
  q=x(1); phi=x(2); %estados
  R=1; L=1; C=1; %parametros
  v=sin(t); %entrada
  uC=q/c;
  iL=phi/L;
  uS=v;
  iR=iL;
  iC=iL;
  iS=iR;
  uR=R*iR;
  uL=uS-uR-uC;
  der_q=iC;
  der_phi=uL;
  dx=[der_q; der_phi]; %vector de derivadas
end
```

Esta función puede utilizarse junto con el método de Forward Euler del Código 2.1 para simular el sistema, y puede verse que es equivalente a la función que obtendríamos si implementamos directamente el código a partir del modelo en Ecuaciones de Estado dado por la Ec.(3.2):

Código 3.4: Función de Matlab/Octave con el modelo de la Ec.(3.2)

```
function dx=circuit2(x,t)
  q=x(1); phi=x(2); %estados
  R=1; L=1; C=1; %parametros
```

```
  v=sin(t); %entrada
  der_q=phi/L;
  der_phi=-q/C-R/L*phi+v;
  dx=[der_q; der_phi]; %vector de derivadas
end
```

Estos modelos explícitos son muchísimo más eficientes que el del Código 3.1 ya que no involucran iteraciones para resolver los sistemas de ecuaciones.

El pasaje de la Ec.(3.1) a la Ec.(3.3) implica ordenar las ecuaciones horizontalmente (para decidir que variable se despeja de cada ecuación) y verticalmente (para decidir en qué orden se realizan las asignaciones). Veremos entonces un procedimiento que en muchos casos permite resolver este problema de ordenamiento causalizando un sistema de DAEs y convirtiéndolo en una secuencia de asignaciones que equivale a un sistema de Ecuaciones de Estado.

3.1.2. Algoritmo de Ordenamiento Horizontal y Vertical

En un caso general, tendremos un sistema de ecuaciones diferenciales algebraicas de la forma:

$$\begin{aligned} f_1(x_1(t),\dots,x_n(t),\dot{x}_1(t),\dots,\dot{x}_n(t),a_1(t),\dots,a_r(t),v_1(t),\dots,v_m(t),t) &= 0 \\ f_2(x_1(t),\dots,x_n(t),\dot{x}_1(t),\dots,\dot{x}_n(t),a_1(t),\dots,a_r(t),v_1(t),\dots,v_m(t),t) &= 0 \\ &\vdots \\ f_{n+r}(x_1(t),\dots,x_n(t),\dot{x}_1(t),\dots,\dot{x}_n(t),a_1(t),\dots,a_r(t),v_1(t),\dots,v_m(t),t) &= 0 \end{aligned} \tag{3.4}$$

donde las variables x_i serán (en principio) los estados, las a_i son variables algebraicas y las entradas v_i se asumen conocidas.

El objetivo será entonces ordenar tanto vertical como horizontalmente estas ecuaciones de manera de poder calcular las derivadas de los estados $\dot{x}_i$. Para esto, tomaremos en cuenta lo siguiente:

- Tendremos un sistema de $n+r$ ecuaciones y $n+r$ incógnitas.
- Las incógnitas son las n derivadas de los estados y las r variables algebraicas desconocidas.

El algoritmo que utilizaremos se basará en las siguientes reglas:

Regla 1 Si una ecuación (E_i) contiene una *única* incógnita u_j (que puede ser una derivada o una variable algebraica), entonces:

- u_j debe despejarse de (E_i) (ordenamiento *horizontal*). Esto debe ser así porque de lo contrario no podríamos utilizar dicha ecuación para otra cosa.
- La ecuación (E_i) despejada debe colocarse al principio (ordenamiento *vertical*). Esto puede hacerse así ya que esta ecuación no necesita que ninguna otra incógnita se calcule previamente.

Regla 2 Si una incógnita u_j aparece en una *única* ecuación (E_i), entonces:

- u_j debe despejarse de (E_i) (ordenamiento *horizontal*). Esto debe ser así porque no hay otra ecuación para calcular u_j.
- La ecuación (E_i) despejada debe colocarse al final (ordenamiento *vertical*). Esto puede hacerse así ya que la incógnita u_j no será necesaria en ninguna otra ecuación.

Cada vez que apliquemos una de estas reglas, la incógnita que despejamos dejará de ser considerada incógnita y la ecuación utilizada se removerá del sistema. De esta manera, si todo funciona bien (luego veremos que esto no será siempre así), tras $n+r$ iteraciones del algoritmo tendríamos nuestro sistema ordenado tanto vertical como horizontalmente.

Si aplicamos entonces este procedimiento sobre el modelo del circuito dado por el sistema de DAEs de la Ec.(3.1), en el primer paso encontramos que podemos aplicar la Regla 1 a la tercer ecuación, ya que

esta contiene una única incógnita ($u_C(t)$) que deberemos despejar de la misma y colocar arriba de todo. Colocamos entonces a la derecha la ecuación causalizadas.

$$
\begin{aligned}
\boldsymbol{u_R(t)} - R\ \boldsymbol{i_R(t)} &= 0 \\
\boldsymbol{\dot{q}(t)} - \boldsymbol{i_C(t)} &= 0 \\
\boxed{q(t) - C\ \boldsymbol{u_C(t)} = 0} & \\
\boldsymbol{\dot{\phi}(t)} - \boldsymbol{u_L(t)} &= 0 \\
\phi(t) - L\ \boldsymbol{i_L(t)} &= 0 \\
\boldsymbol{u_S(t)} - v(t) &= 0 \\
\boldsymbol{u_L(t)} + \boldsymbol{u_R(t)} + \boldsymbol{u_C(t)} - \boldsymbol{u_S(t)} &= 0 \\
\boldsymbol{i_L(t)} - \boldsymbol{i_R(t)} &= 0 \\
\boldsymbol{i_C(t)} - \boldsymbol{i_L(t)} &= 0 \\
\boldsymbol{i_S(t)} - \boldsymbol{i_R(t)} &= 0
\end{aligned}
\qquad u_C(t) := \frac{q(t)}{C}
$$

A partir de este punto, u_C deja de ser una incógnita (y no la representamos más en negritas) removiendo además la ecuación correspondiente del sistema de DAEs.

En el siguiente paso, vemos que la incógnita $\dot{\phi}(t)$ sólo aparece en la cuarta ecuación y podemos por lo tanto aplicar la Regla 2, tras lo cual colocamos la ecuación despejada al final.

$$
\begin{aligned}
\boldsymbol{u_R(t)} - R\ \boldsymbol{i_R(t)} &= 0 \\
\boldsymbol{\dot{q}(t)} - \boldsymbol{i_C(t)} &= 0 \\
q(t) - C\ \boldsymbol{u_C(t)} &= 0 \\
\boxed{\boldsymbol{\dot{\phi}(t)} - \boldsymbol{u_L(t)} = 0} & \\
\phi(t) - L\ \boldsymbol{i_L(t)} &= 0 \\
\boldsymbol{u_S(t)} - v(t) &= 0 \\
\boldsymbol{u_L(t)} + \boldsymbol{u_R(t)} + u_C(t) - \boldsymbol{u_S(t)} &= 0 \\
\boldsymbol{i_L(t)} - \boldsymbol{i_R(t)} &= 0 \\
\boldsymbol{i_C(t)} - \boldsymbol{i_L(t)} &= 0 \\
\boldsymbol{i_S(t)} - \boldsymbol{i_R(t)} &= 0
\end{aligned}
\qquad
\begin{aligned}
u_C(t) &:= \frac{q(t)}{C} \\
\\ \\ \\ \\ \\ \\ \\ \\ \\
\dot{\phi}(t) &:= u_L(t)
\end{aligned}
$$

Tras eliminar la ecuación anterior, al avanzar a la ecuación siguiente vemos que la misma contiene una única incógnita y podemos entonces aplicar la Regla 1.

$$
\begin{aligned}
\boldsymbol{u_R(t)} - R\ \boldsymbol{i_R(t)} &= 0 && u_C(t) := \frac{q(t)}{C} \\
\boldsymbol{\dot{q}(t)} - \boldsymbol{i_C(t)} &= 0 && i_L(t) := \frac{\phi(t)}{L} \\
q(t) - C\ \boldsymbol{u_C(t)} &= 0 \\
\dot{\phi}(t) - u_L(t) &= 0 \\
\boxed{\phi(t) - L\ \boldsymbol{i_L(t)} = 0} & \\
\boldsymbol{u_S(t)} - v(t) &= 0 \\
\boldsymbol{u_L(t)} + \boldsymbol{u_R(t)} + u_C(t) - \boldsymbol{u_S(t)} &= 0 \\
\boldsymbol{i_L(t)} - \boldsymbol{i_R(t)} &= 0 \\
\boldsymbol{i_C(t)} - \boldsymbol{i_L(t)} &= 0 \\
\boldsymbol{i_S(t)} - \boldsymbol{i_R(t)} &= 0 && \dot{\phi}(t) := u_L(t)
\end{aligned}
$$

La siguiente ecuación también contiene una única incógnita (u_S), por lo que aplicamos nuevamente la Regla 1.

$$
\begin{aligned}
\boldsymbol{u_R(t)} - R\ \boldsymbol{i_R(t)} &= 0 && u_C(t) := \frac{q(t)}{C} \\
\boldsymbol{\dot{q}(t)} - \boldsymbol{i_C(t)} &= 0 && i_L(t) := \frac{\phi(t)}{L} \\
q(t) - C\ \boldsymbol{u_C(t)} &= 0 && u_S(t) := v(t) \\
\dot{\phi}(t) - u_L(t) &= 0 \\
\phi(t) - L\ \boldsymbol{i_L(t)} &= 0 \\
\boxed{\boldsymbol{u_S(t)} - v(t) = 0} & \\
\boldsymbol{u_L(t)} + \boldsymbol{u_R(t)} + u_C(t) - \boldsymbol{u_S(t)} &= 0 \\
i_L(t) - \boldsymbol{i_R(t)} &= 0 \\
\boldsymbol{i_C(t)} - i_L(t) &= 0 \\
\boldsymbol{i_S(t)} - \boldsymbol{i_R(t)} &= 0 && \dot{\phi}(t) := u_L(t)
\end{aligned}
$$

Luego de este paso, podemos ver por ejemplo que $\dot{q}$ aparece sólo en la segunda ecuación y aplicar la Regla 2.

$$
\begin{aligned}
\boldsymbol{u_R(t)} - R\ \boldsymbol{i_R(t)} &= 0 && u_C(t) := \frac{q(t)}{C} \\
\boxed{\boldsymbol{\dot{q}(t)} - \boldsymbol{i_C(t)} = 0} & && i_L(t) := \frac{\phi(t)}{L} \\
q(t) - C\ \boldsymbol{u_C(t)} &= 0 && u_S(t) := v(t) \\
\dot{\phi}(t) - u_L(t) &= 0 \\
\phi(t) - L\ \boldsymbol{i_L(t)} &= 0 \\
\boldsymbol{u_S(t)} - v(t) &= 0 \\
\boldsymbol{u_L(t)} + \boldsymbol{u_R(t)} + u_C(t) - u_S(t) &= 0 \\
i_L(t) - \boldsymbol{i_R(t)} &= 0 \\
\boldsymbol{i_C(t)} - i_L(t) &= 0 && \dot{q}(t) := i_C(t) \\
\boldsymbol{i_S(t)} - \boldsymbol{i_R(t)} &= 0 && \dot{\phi}(t) := u_L(t)
\end{aligned}
$$

Continuando con este procedimiento, tras cinco pasos más, llegamos finalmente al sistema de la Ec.(3.3) que contiene todas las ecuaciones ordenadas horizontal y verticalmente y que, como vimos, resulta equivalente a un sistema de Ecuaciones de Estado.

Las ecuaciones ordenadas de la Ec.(3.3) pueden utilizarse tanto para simular un sistema utilizando una función como la del Código 3.3 como para obtener las Ecuaciones de Estado de manera sistemática. Esto es útil para luego realizar análisis sobre el modelo, utilizando las EE o la Función Transferencia (asumiendo que el sistema es lineal) que se puede obtener directamente de las EE.

Dado que en cada paso del algoritmo ordenamos una ecuación, en un caso genérico como el de la Ec.(3.4) el procedimiento finalizará tras $n+r$ pasos. Tanto para acelerar el proceso de causalización como para generar un código más corto y eficiente para evaluar las derivadas del estado, se suelen eliminar las ecuaciones *triviales* del modelo, eliminando también las correspondientes variables algebraicas.

3.1.3. Eliminación de Ecuaciones Triviales

Las ecuaciones triviales son las que tienen la forma $a_i - a_j = 0$ (o sea, $a_i = a_j$) y en tal caso se dice que las variables algebraicas a_i y a_j son *alias* y se puede eliminar una de ellas. En la Ec.(3.1) podríamos eliminar las últimas tres ecuaciones que son triviales reemplazando las corrientes por una única corriente (i_L por ejemplo), lo que reduciría de 8 a 5 el número de variables algebraicas.

$$\begin{aligned} u_R(t) - R\ i_L(t) &= 0 \\ \dot{q}(t) - i_L(t) &= 0 \\ q(t) - C\ u_C(t) &= 0 \\ \dot{\phi}(t) - u_L(t) &= 0 \\ \phi(t) - L\ i_L(t) &= 0 \\ u_S(t) - v(t) &= 0 \\ u_L(t) + u_R(t) + u_C(t) - u_S(t) &= 0 \end{aligned}$$

En realidad, es posible también eliminar i_L y u_L ya que ambas son iguales a dos derivadas de los estados (podemos tratar las derivadas de los estados como variables algebraicas, pero que no se pueden eliminar). En tal caso, el sistema se reduce a

$$u_R(t) - R\ \dot{q}(t) = 0 \tag{3.6a}$$
$$q(t) - C\ u_C(t) = 0 \tag{3.6b}$$
$$\phi(t) - L\ \dot{q}(t) = 0 \tag{3.6c}$$
$$u_S(t) - v(t) = 0 \tag{3.6d}$$
$$\dot{\phi}(t) + u_R(t) + u_C(t) - u_S(t) = 0 \tag{3.6e}$$

que sólo contiene tres variables algebraicas.

3.1.4. Obtención de las Ecuaciones de Estado

Una vez ordenado horizontal y verticalmente un sistema de DAEs, la obtención de las ecuaciones de estado es directa. Se puede directamente comenzar desde la primera ecuación del sistema ordenado e ir reemplazando las variables algebraicas por sus expresiones en función de los estados.

Por ejemplo, si usamos el algoritmo de causalización en el sistema reducido (sin ecuaciones triviales) de la Ec.(3.6), obtenemos el siguiente sistema ordenado:

$$u_C(t) := \frac{q(t)}{C} \tag{3.7a}$$
$$\dot{q}(t) := \frac{\phi(t)}{L} \tag{3.7b}$$
$$u_S(t) := v(t) \tag{3.7c}$$
$$u_R(t) := R\ \dot{q}(t) \tag{3.7d}$$
$$\dot{\phi}(t) := u_S(t) - u_R(t) - u_C(t) \tag{3.7e}$$

y podemos proceder como sigue:

$$\begin{aligned} u_C(t) &:= \frac{q(t)}{C} \\ \dot{q}(t) &:= \frac{\phi(t)}{L} \\ u_S(t) &:= v(t) \\ u_R(t) &:= R\ \dot{q}(t) = R\ \frac{\phi(t)}{L} \\ \dot{\phi}(t) &:= u_S(t) - u_R(t) - u_C(t) = v(t) - R\ \frac{\phi(t)}{L} - \frac{q(t)}{C} \end{aligned}$$

de manera que las EE son:

$$\begin{aligned} \dot{q}(t) &= \frac{\phi(t)}{L} \\ \dot{\phi}(t) &= v(t) - R\ \frac{\phi(t)}{L} - \frac{q(t)}{C} \end{aligned}$$

3.2. Lazos Algebraicos

3.2.1. Ejemplo Introductorio

En el ejemplo de la Ec.(3.1) el algoritmo de causalización funcionó muy bien y permitió obtener un modelo explícito. La pregunta es ¿funcionará siempre?.

Veamos ahora en la Figura 3.2 otro circuito, en principio más simple que el anterior, junto al sistema de DAEs que definen sus relaciones constitutivas y estructurales.

$$u_{R_1}(t) - R_1\ i_{R_1}(t) = 0 \tag{3.8a}$$
$$u_{R_2}(t) - R_2\ i_{R_2}(t) = 0 \tag{3.8b}$$
$$\dot{\phi}(t) - u_L(t) = 0 \tag{3.8c}$$
$$\phi(t) - L\ i_L(t) = 0 \tag{3.8d}$$
$$i_L(t) + i_{R_1}(t) + i_{R_2}(t) = 0 \tag{3.8e}$$
$$u_{R_1}(t) - u_{R_2}(t) = 0 \tag{3.8f}$$
$$u_L(t) - u_{R_2}(t) = 0 \tag{3.8g}$$

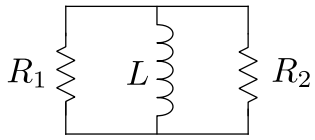

Figura 3.2: Circuito RL Paralelo

Este sistema de ecuaciones diferenciales algebraicas tiene, en principio, una variable de estado (ϕ) y 6 variables algebraicas (tensiones y corrientes en cada elemento) lo que constituye un sistema de 7 ecuaciones con 7 incógnitas (recordando que una de las incógnitas es la derivada del estado $\dot{\phi}$).

Veamos entonces que sucede al utilizar el procedimiento de ordenamiento que vimos anteriormente. Para esto comenzaremos eliminando las ecuaciones triviales (las últimas dos) y las variables algebraicas

correspondientes (u_{R_1} y u_{R_2}). Luego, observando que la derivada $\dot{\phi}$ aparece en una única ecuación, podemos aplicar la Regla 2:

$$\begin{aligned}
\boldsymbol{u_L(t)} - R_1\ \boldsymbol{i_{R_1}(t)} &= 0 \\
\boldsymbol{u_L(t)} - R_2\ \boldsymbol{i_{R_2}(t)} &= 0 \\
\boxed{\boldsymbol{\dot{\phi}(t)} - \boldsymbol{u_L(t)} = 0} & \\
\phi(t) - L\ \boldsymbol{i_L(t)} &= 0 \\
\boldsymbol{i_L(t)} + \boldsymbol{i_{R_1}(t)} + \boldsymbol{i_{R_2}}(t) &= 0
\end{aligned} \qquad \dot{\phi}(t) := u_L(t)$$

La ecuación siguiente tiene sólo una incógnita (iL), por lo que podemos aplicar ahora la Regla 1.

$$\begin{aligned}
\boldsymbol{u_L(t)} - R_1\ \boldsymbol{i_{R_1}(t)} &= 0 \\
\boldsymbol{u_L(t)} - R_2\ \boldsymbol{i_{R_2}(t)} &= 0 \\
\dot{\phi}(t) - u_L(t) &= 0 \\
\boxed{\phi(t) - L\ \boldsymbol{i_L(t)} = 0} & \\
\boldsymbol{i_L(t)} + \boldsymbol{i_{R_1}(t)} + \boldsymbol{i_{R_2}}(t) &= 0
\end{aligned} \qquad \begin{aligned} i_L(t) &:= \frac{\phi(t)}{L} \\ \\ \\ \dot{\phi}(t) &:= u_L(t) \end{aligned}$$

Tras este segundo paso, no podemos aplicar ninguna de las reglas ya que todas las ecuaciones tienen dos incógnitas y todas las incógnitas aparecen en al menos dos ecuaciones. En nuestro procedimiento nos quedaron entonces un par de ecuaciones ordenadas (la primera y la última) y tres ecuaciones sin ordenar:

$$\begin{aligned}
\boldsymbol{u_L(t)} - R_1\ \boldsymbol{i_{R_1}(t)} &= 0 \\
\boldsymbol{u_L(t)} - R_2\ \boldsymbol{i_{R_2}(t)} &= 0 \\
\dot{\phi}(t) - u_L(t) &= 0 \\
\phi(t) - L\ i_L(t) &= 0 \\
i_L(t) + \boldsymbol{i_{R_1}(t)} + \boldsymbol{i_{R_2}}(t) &= 0
\end{aligned} \qquad \begin{aligned} i_L(t) &:= \frac{\phi(t)}{L} \\ \boldsymbol{u_L(t)} - R_1\ \boldsymbol{i_{R_1}(t)} &= 0 \\ \boldsymbol{u_L(t)} - R_2\ \boldsymbol{i_{R_2}(t)} &= 0 \\ i_L(t) + \boldsymbol{i_{R_1}(t)} + \boldsymbol{i_{R_2}}(t) &= 0 \\ \dot{\phi}(t) &:= u_L(t) \end{aligned}$$

Observando el resultado, es claro que nos queda un sistema de tres ecuaciones con tres incógnitas (i_{R_1}, i_{R_2} y u_L) que deben resolverse simultáneamente. Estas ecuaciones, en este caso, constituyen lo que se denomina un *lazo algebraico*.

Cuando el procedimiento de ordenamiento se encuentra en una situación en la cuál no se puede aplicar ninguna de las reglas porque todas las ecuaciones tienen al menos dos incógnitas y todas las incógnitas aparecen en al menos dos ecuaciones, puede verse que hay uno o más lazos algebraicos.

3.2.2. Algoritmo de Rasgado

A priori, podríamos resolver el problema implementando algo similar a lo que hicimos con el Código 3.1. De hecho, si hacemos esto directamente sobre la Ec.(3.8) tendríamos un sistema de 7 ecuaciones y 7 incógnitas, que podría reducirse a 5 ecuaciones e incógnitas tras eliminar las ecuaciones triviales. Utilizando sin embargo la última expresión, podríamos reducir aún más esto a un problema de 3 ecuaciones con 3 incógnitas.

Código 3.5: Función de Matlab/Octave con el modelo de la Ec.(3.8)

```
function dx=rlcircuit(x,t)
  R1=1;R2=2;L=1; %parametros
  phi=x; %estado
```

```
  iL=phi/L;
  %la siguiente funcion debe ser 0
  F = @(z) rlcircuit_loop(z(1),z(2),z(3),iL);
  z0=zeros(3,1);
  z=fsolve(F,z0);
  uL=z(1);
  iR1=z(2);
  iR2=z(3);
  der_phi=uL;
  dx=der_phi;
end

function res=rlcircuit_loop(uL,iR1,iR2,iL)
  R1=1;R2=2;L=1; %parametros
  %la siguiente funcion debe ser 0
  res=zeros(3,1);
  res(1)=uL-R1*iR1;
  res(2)=uL-R2*iR2;
  res(3)=iL+iR1+iR2;
end
```

Si bien esto es más eficiente que resolver un sistema de 5 ecuaciones como el que resultaría al colocar directamente la DAE, todavía podemos hacer algo más.

Cuando en el procedimiento de ordenamiento nos encontramos con un lazo algebraico, un truco es suponer que alguna incógnita del lazo es conocida (aunque no lo sea), guardar una ecuación donde aparezca dicha incógnita y continuar el proceso de ordenamiento, colocando al final del mismo la ecuación que nos guardamos.

Continuando con el ejemplo anterior, si suponemos conocida la variable algebraica u_L podríamos *guardar* la primera ecuación y ordenar la segunda que ahora tendría una única incógnita lo que permitiría utilizar la Regla 1.

$$
\begin{aligned}
u_L(t) - R_1\ \boldsymbol{i_{R_1}(t)} &= 0 \\
\boxed{u_L(t) - R_2\ \boldsymbol{i_{R_2}(t)} = 0} & \\
\dot{\phi}(t) - u_L(t) &= 0 \\
\phi(t) - L\ i_L(t) &= 0 \\
i_L(t) + \boldsymbol{i_{R_1}(t)} + \boldsymbol{i_{R_2}}(t) &= 0
\end{aligned}
\qquad
\begin{aligned}
i_L(t) &:= \frac{\phi(t)}{L} \\
i_{R_2}(t) &:= \frac{\boldsymbol{u_L(t)}}{R_2} \\
& \\
\dot{\phi}(t) &:= u_L(t)
\end{aligned}
$$

Tras este paso, la última ecuación tiene una única incógnita y podemos aplicar nuevamente la Regla 1:

$$
\begin{aligned}
u_L(t) - R_1\ \boldsymbol{i_{R_1}(t)} &= 0 \\
u_L(t) - R_2\ i_{R_2}(t) &= 0 \\
\dot{\phi}(t) - u_L(t) &= 0 \\
\phi(t) - L\ i_L(t) &= 0 \\
\boxed{i_L(t) + \boldsymbol{i_{R_1}(t)} + i_{R_2}(t) = 0} & \\
\end{aligned}
\qquad
\begin{aligned}
i_L(t) &:= \frac{\phi(t)}{L} \\
i_{R_2}(t) &:= \frac{\boldsymbol{u_L(t)}}{R_2} \\
i_{R_1}(t) &:= -i_{R_2}(t) - i_L(t) \\
\dot{\phi}(t) &:= u_L(t)
\end{aligned}
$$

En este punto, podemos ver que la ecuación que reservamos en rojo ya no tiene incógnitas, excepto

que en realidad no conocemos $u_L(t)$. En definitiva, el sistema ordenado sería:

$$i_L(t) := \frac{\phi(t)}{L} \tag{3.9a}$$

$$i_{R_2}(t) := \frac{\boldsymbol{u_L(t)}}{R_2} \tag{3.9b}$$

$$i_{R_1}(t) := -i_{R_2}(t) - i_L(t) \tag{3.9c}$$

$$\boldsymbol{u_L(t)} - R_1\ i_{R_1}(t) = 0 \tag{3.9d}$$

$$\dot{\phi}(t) := u_L(t) \tag{3.9e}$$

que tiene una única variable no explícita (u_L). De esta forma, podemos implementar una iteración sobre una única variable que resuelva el sistema de ecuaciones algebraicas que define la derivada del estado.

La función de Matlab/Octave del Código 3.6 brinda una implementación de esta idea y resulta mucho más eficiente que la del Código 3.5 .

Código 3.6: Función de Matlab/Octave con el modelo de la Ec.(3.9)

```
function dx=rlcircuit2(x,t)
  R1=1;R2=2;L=1; %parametros
  phi=x; %estado
  iL=phi/L;
  %la siguiente funcion debe ser 0
  F = @(uL) rlcircuit_loop2(uL,iL);
  uL=fsolve(F,0);
  der_phi=uL;
  dx=der_phi;
end

function res=rlcircuit_loop2(uL,iL)
  R1=1;R2=2;L=1; %parametros
  %la siguiente funcion debe ser 0
  iR2=uL/R2;
  iR1=-iR2-iL;
  res=uL-R1*iR1;
end
```

En este procedimiento de *rasgado* o *ruptura* del lazo (*tearing* en inglés) se denomina *variable de rasgado* a la variable que suponemos conocida (u_L en el ejemplo). En ocasiones se debe elegir más de una variable de rasgado y en modelos grandes es común que exista más de un lazo algebraico.

3.2.3. Obtención de las Ecuaciones de Estado

Podemos avanzar incluso un paso más y reemplazar en el sistema de ecuaciones del lazo utilizando las variables de rasgado y luego despejarlas de manera algebraica cuando esto sea posible, lo que transforma finalmente el sistema de DAEs en un sistema de ecuaciones de estado totalmente explícito.

En el caso del ejemplo, podemos reemplazar sucesivamente usando las Ecuaciones (3.9b) y luego la (3.9c) en la (3.9d), llegando a

$$u_L(t) - R_1\ \left(-\frac{u_L(t)}{R_2} - i_L(t)\right) = 0, \tag{3.10}$$

de donde finalmente

$$u_L(t) = -\frac{R_1}{1 + \dfrac{R_1}{R_2}}\, i_L(t). \tag{3.11}$$

Utilizando esta expresión, la Ec.(3.9) se torna totalmente explícita:

$$i_L(t) := \frac{\phi(t)}{L} \tag{3.12a}$$

$$u_L(t) := -\frac{R_1}{1+\dfrac{R_1}{R_2}}\, i_L(t) \tag{3.12b}$$

$$\dot{\phi}(t) := u_L(t). \tag{3.12c}$$

y de aquí es trivial pasar a la forma de una Ecuación de Estados (en este caso de primer orden) reemplazando las variables algebraicas restantes i_L y u_L sucesivamente, según:

$$u_L(t) = -\frac{R_1}{1+\dfrac{R_1}{R_2}}\, \frac{\phi(t)}{L},$$

y finalmente

$$\dot{\phi}(t) = -\frac{R_1}{1+\dfrac{R_1}{R_2}}\, \frac{\phi(t)}{L}. \tag{3.13}$$

El procedimiento de causalización que seguimos puede aplicarse con cualquier sistema, independientemente de que sea lineal o no lineal. La diferencia que encontraremos es que en casos no lineales es poco probable que podamos despejar la expresión de las variables de rasgado a partir de una Ecuación como la Ec.(3.10) para llegar a la Ec.(3.11). Por lo tanto, en tales casos no será posible en general obtener un sistema de Ecuaciones de Estado explícitas. Más aún, algunos sistemas no lineales contienen expresiones de las cuales no es posible despejar analíticamente la variable que corresponde al ordenamiento horizontal según el algoritmo básico de causalización y deben implementarse iteraciones aunque no haya un lazo algebraico.

3.3. Sistemas de Índice Alto

3.3.1. Ejemplo Introductorio

Consideremos ahora el circuito de la Figura 3.3 con sus relaciones constitutivas y estructurales dadas por la Ec.(3.14).

$$C_1\ \dot{u}_{C_1}(t) - i_{C_1}(t) = 0 \tag{3.14a}$$

$$C_2\ \dot{u}_{C_2}(t) - i_{C_2}(t) = 0 \tag{3.14b}$$

$$u_R(t) - R\ i_R(t) = 0 \tag{3.14c}$$

$$i_R(t) + i_{C_1}(t) + i_{C_2}(t) = 0 \tag{3.14d}$$

$$u_{C_1}(t) - u_R(t) = 0 \tag{3.14e}$$

$$u_{C_1}(t) - u_{C_2}(t) = 0 \tag{3.14f}$$

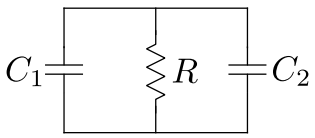

Figura 3.3: Circuito RC Paralelo

En principio este sistema tiene dos variables de estado (u_{C_1} y u_{C_2}) y cuatro variables algebraicas (i_{C_1}, i_{C_2}, i_R y u_R). Dado que tiene seis ecuaciones y seis incógnitas, el modelo estaría, a priori, bien planteado. Sin embargo, si intentamos aplicar el procedimiento de causalización, podemos ver inmediatamente que Ec.(3.14f) no tiene incógnitas (recordemos que los estados se asumen conocidos) y por lo tanto no puede utilizarse.

La causa del problema es bastante evidente: estamos intentando utilizar dos variables de estado (u_{C_1} y u_{C_2}) en un modelo que es de primer orden ya que hay una ecuación, la Ec.(3.14f), que dice que ambas variables deben ser iguales y por lo tanto no podemos elegir independientemente sus condiciones iniciales. En general, cuando una ecuación se queda sin incógnitas durante el proceso de causalización se dice que

la misma constituye una *Ecuación de Restricción* y esto se debe a que el modelo tiene un orden menor al número de variables que supusimos que eran estados porque aparecían sus derivadas.

Cuando ocurre esto, decimos que el sistema de DAEs es *estructuralmente singular*. En un caso general, podemos detectar la presencia de singularidades estructurales cuando al aplicar el procedimiento de causalización, en algún momento una ecuación se queda sin incógnitas.

Dado que no podemos causalizar este sistema, una primera opción podría ser intentar nuevamente implementar una solución de *fuerza bruta* y tratar de simular el sistema iterando sobre la DAE para obtener la derivada del estado $\dot{\mathbf{x}}$ en cada paso. Esta idea puede llevarse a cabo con el Código 3.7.

Código 3.7: Función de Matlab/Octave con el modelo de la Ec.(3.14)

```
function dx=rccircuit_ode(x,t)
  F = @(z) rccircuit_dae(x,z(1:2),z(3:6),t);
  z0=zeros(6,1);
  z=fsolve(F,z0);
  dx=z(1:2);
end

function res=rc_circuit_dae(x,dx,a,t)
  C1=1;C2=2;R=1; %parametros
  uC1=x(1); uC2=x(2);  %estados
  der_uC1=dx(1); der_uC2=dx(2);%derivadas de estados
  uR=a(1);iR=a(2);iC1=a(3);iC2=a(4); %variables algebraicas
  res=zeros(6,1); %residuo (debe ser cero)
  res(1) = C1*der_uC1-iC1;
  res(2) = C2*der_uC2-iC2;
  res(3) = uR-R*iR;
  res(4) = iR+iC1+iC2;
  res(5) = uC1-uR;
  res(6) = uC1-uC2;
end
```

Desafortunadamente, la idea no va a funcionar. Un primer problema es que tenemos que dar dos condiciones iniciales (el estado tiene dos componentes). Poniendo de todas formas ambas condiciones iguales para cumplir con la última ecuación, al ejecutar

```
[t,x]=feuler(@rccircuit_ode,[1;1],0.1,0,1);
```

la función que resuelve las ecuaciones algebraicas de la DAE (`fsolve`) se encuentra con que la matriz que debe invertir para la iteración de Newton es singular y por lo tanto no puede resolver correctamente el sistema.

3.3.2. Algoritmo de Pantelides

¿Cómo se puede solucionar esto?. La idea básica, debida a Constantino Pantelides [Pan88], es derivar respecto al tiempo las ecuaciones de restricción. Si $u_{C_1}(t) - u_{C_2}(t) = 0$, luego debe cumplirse también que $\dot{u}_{C_1}(t) - \dot{u}_{C_2}(t) = 0$ por lo que agregaremos esta última ecuación al sistema. Dado que ahora tenemos una ecuación más, necesitamos también una incógnita más, lo que lograremos decidiendo que una de las variables de estado de la Ecuación de Restricción no sea más variable de estado.

Suponiendo entonces que u_{C_2} no es más un estado, por ejemplo, el nuevo sistema de ecuaciones queda:

$$C_1\ \dot{\boldsymbol{u}}_{\boldsymbol{C_1}}(\boldsymbol{t}) - \boldsymbol{i}_{\boldsymbol{C_1}}(\boldsymbol{t}) = 0 \tag{3.15a}$$
$$C_2\ \boldsymbol{du}_{\boldsymbol{C_2}}(\boldsymbol{t}) - \boldsymbol{i}_{\boldsymbol{C_2}}(\boldsymbol{t}) = 0 \tag{3.15b}$$
$$\boldsymbol{u}_{\boldsymbol{R}}(\boldsymbol{t}) - R\ \boldsymbol{i}_{\boldsymbol{R}}(\boldsymbol{t}) = 0 \tag{3.15c}$$
$$\boldsymbol{i}_{\boldsymbol{R}}(\boldsymbol{t}) + \boldsymbol{i}_{\boldsymbol{C_1}}(\boldsymbol{t}) + \boldsymbol{i}_{\boldsymbol{C_2}}(\boldsymbol{t}) = 0 \tag{3.15d}$$
$$u_{C_1}(t) - \boldsymbol{u}_{\boldsymbol{R}}(\boldsymbol{t}) = 0 \tag{3.15e}$$
$$u_{C_1}(t) - \boldsymbol{u}_{\boldsymbol{C_2}}(\boldsymbol{t}) = 0 \tag{3.15f}$$
$$\dot{\boldsymbol{u}}_{\boldsymbol{C_1}}(\boldsymbol{t}) - \boldsymbol{du}_{\boldsymbol{C_2}}(\boldsymbol{t}) = 0 \tag{3.15g}$$

En este nuevo sistema tenemos siete ecuaciones y siete incógnitas (ahora u_{C_2} es una variable algebraica). Además, lo que antes llamábamos $\dot{u}_{C_2}$ ahora es una nueva variable algebraica du_{C_2} que representa lo que se llama una derivada *ficticia* (*dummy derivative* en inglés). Si bien proviene de lo que era la derivada de u_{C_2}, para el nuevo sistema es sólo una nueva variable algebraica.

Veamos entonces el resultado de aplicar el algoritmo de causalización sobre este nuevo sistema. Podemos ver fácilmente que es posible aplicar la Regla 1 a las Ecs.(3.15e) y (3.15f).

$$\begin{aligned}
C_1\ \dot{\boldsymbol{u}}_{\boldsymbol{C_1}}(\boldsymbol{t}) - \boldsymbol{i}_{\boldsymbol{C_1}}(\boldsymbol{t}) &= 0 \\
C_2\ \boldsymbol{du}_{\boldsymbol{C_2}}(\boldsymbol{t}) - \boldsymbol{i}_{\boldsymbol{C_2}}(\boldsymbol{t}) &= 0 \\
u_R(t) - R\ \boldsymbol{i}_{\boldsymbol{R}}(\boldsymbol{t}) &= 0 \\
\boldsymbol{i}_{\boldsymbol{R}}(\boldsymbol{t}) + \boldsymbol{i}_{\boldsymbol{C_1}}(\boldsymbol{t}) + \boldsymbol{i}_{\boldsymbol{C_2}}(\boldsymbol{t}) &= 0 \\
u_{C_1}(t) - \boldsymbol{u}_{\boldsymbol{R}}(\boldsymbol{t}) &= 0 \\
u_{C_1}(t) - \boldsymbol{u}_{\boldsymbol{C_2}}(\boldsymbol{t}) &= 0 \\
\dot{\boldsymbol{u}}_{\boldsymbol{C_1}}(\boldsymbol{t}) - \boldsymbol{du}_{\boldsymbol{C_2}}(\boldsymbol{t}) &= 0
\end{aligned}
\qquad
\begin{aligned}
u_R(t) &:= u_{C_1}(t) \\
u_{C_2}(t) &:= u_{C_1}(t)
\end{aligned}$$

Tras estos dos primeros pasos, la tercera ecuación tiene una única incógnita y puede aplicarse nuevamente la Regla 1 para calcular $i_R(t)$.

$$\begin{aligned}
C_1\ \dot{\boldsymbol{u}}_{\boldsymbol{C_1}}(\boldsymbol{t}) - \boldsymbol{i}_{\boldsymbol{C_1}}(\boldsymbol{t}) &= 0 \\
C_2\ \boldsymbol{du}_{\boldsymbol{C_2}}(\boldsymbol{t}) - \boldsymbol{i}_{\boldsymbol{C_2}}(\boldsymbol{t}) &= 0 \\
u_R(t) - R\ \boldsymbol{i}_{\boldsymbol{R}}(\boldsymbol{t}) &= 0 \\
i_R(t) + \boldsymbol{i}_{\boldsymbol{C_1}}(\boldsymbol{t}) + \boldsymbol{i}_{\boldsymbol{C_2}}(\boldsymbol{t}) &= 0 \\
u_{C_1}(t) - \boldsymbol{u}_{\boldsymbol{R}}(\boldsymbol{t}) &= 0 \\
u_{C_1}(t) - \boldsymbol{u}_{\boldsymbol{C_2}}(\boldsymbol{t}) &= 0 \\
\dot{\boldsymbol{u}}_{\boldsymbol{C_1}}(\boldsymbol{t}) - \boldsymbol{du}_{\boldsymbol{C_2}}(\boldsymbol{t}) &= 0
\end{aligned}
\qquad
\begin{aligned}
u_R(t) &:= u_{C_1}(t) \\
u_{C_2}(t) &:= u_{C_1}(t) \\
i_R(t) &:= \frac{u_R(t)}{R}
\end{aligned}$$

Tras este paso, nos encontramos con un sistema de cuatro ecuaciones y cuatro incógnitas, donde todas las incógnitas aparecen en al menos dos ecuaciones y donde todas las ecuaciones contienen al menos dos incógnitas. Es decir, tenemos un *lazo algebraico* en las variables $\dot{u}_{C_1}$, i_{C_1}, i_{C_2} y du_{C_2}.

Para continuar, utilizaremos como variable de rasgado a $\dot{u}_{C_1}$ reservando la primera ecuación. Asumiendo entonces conocida $\dot{u}_{C_1}$ se puede despejar du_{C_2} de la última ecuación, luego i_{C_2} de la segunda, tras esto i_{C_1} se obtiene de la suma de corrientes y esto deja a la primera ecuación sin otra incógnita que la variable de rasgado completando la ruptura del lazo.

Esto se traduce en el siguiente sistema de ecuaciones ordenadas que contienen sólo una incógnita:

$$u_R(t) := u_{C_1}(t) \tag{3.16a}$$
$$u_{C_2}(t) := u_{C_1}(t) \tag{3.16b}$$
$$i_R(t) := \frac{u_R(t)}{R} \tag{3.16c}$$
$$du_{C_2}(t) := \boldsymbol{\dot{u}_{C_1}(t)} \tag{3.16d}$$
$$i_{C_2}(t) := C_2\ du_{C_2} \tag{3.16e}$$
$$i_{C_1}(t) := -i_R(t) - i_{C_2}(t) \tag{3.16f}$$
$$C_1\ \boldsymbol{\dot{u}_{C_1}(t)} - i_{C_1}(t) = 0 \tag{3.16g}$$

La función de Matlab/Octave del Código 3.8 brinda una implementación a partir de las ecuaciones obtenidas tras el ordenamiento y el rasgado del lazo algebraico.

Código 3.8: Función de Matlab/Octave con el modelo de la Ec.(3.16)

```
function dx=rccircuit2(x,t)
  C1=1;C2=2;R=1; %parametros
  uC1=x; %estado
  uR=uC1;
  uC2=uC1;
  iR=uR/R;
  %la siguiente funcion debe ser 0
  F = @(dot_uC1) rccircuit_loop2(dot_uC1,iR);
  dot_uC1=fsolve(F,0);
  dx=dot_uC1;
end

function res=rccircuit_loop2(dot_uC1,iR)
  C1=1;C2=2;R=1; %parametros
  duC2=dot_uC1;
  iC2=C2*duC2;
  iC1=-iR-iC2;
  %la siguiente funcion debe ser 0
  res=C1*dot_uC1-iC1;
end
```

Podemos también obtener de manera explícita la ecuación de estados reemplazando las variables en función de $\dot{u}_{C_1}$ sucesivamente en las Ecs.(3.16d)-(3.16g), de donde se obtiene:

$$\begin{aligned} i_{C_2}(t) &= C_2\ \dot{u}_{C_1}(t) \implies \\ i_{C_1}(t) &= -i_R(t) - C_2\ \dot{u}_{C_1}(t) \implies \\ C_1\ \dot{u}_{C_1}(t) - (-i_R(t) - C_2\ \dot{u}_{C_1}(t)) &= 0 \end{aligned}$$

y reemplazando además la variable algebraica i_R en función de los estados,

$$\begin{aligned} u_R(t) &= u_{C_1}(t) \implies \\ i_R(t) &= \frac{u_{C_1}(t)}{R} \end{aligned}$$

y finalmente

$$\dot{u}_{C_1}(t) = -\frac{u_{C_1}(t)}{R\ (C_1 + C2)} \tag{3.17}$$

En definitiva, utilizando el Algoritmo de Pantelides transformamos un sistema que no puede causalizarse en uno que contiene un lazo algebraico.

El algoritmo de Pantelides se puede resumir mediante los siguientes pasos:

- Derivar respecto al tiempo las ecuaciones de restricción, agregarlas al sistema y transformar una variable de estado en algebraica por cada restricción.
- Al derivar, llamar $\dot{x}_i(t)$ a las derivadas de los estados $x_i(t)$ y $da_j(t)$ a las derivadas de las variables algebraicas $a_j(t)$.
- Si al derivar una restricción aparece una nueva derivada de una variable algebraica, se debe derivar también la ecuación donde se define dicha variable algebraica. Esta nueva ecuación se debe agregar al sistema.
- Esto último debe hacerse recursivamente hasta que no aparezcan nuevas derivadas de variables algebraicas.
- Verificar que el nuevo sistema sea causalizable. Si no lo es, se debe aplicar nuevamente el procedimiento.

3.3.3. El índice de perturbación de una DAE

Los sistemas de ecuaciones diferenciales algebraicas que no contienen lazos algebraicos y que pueden causalizarse completamente utilizando las Reglas 1 y 2 se denominan *DAEs de índice 0*. En estos casos, el sistema puede generalmente transformarse en una ODE explícita reordenando las ecuaciones.

Cuando un sistema contiene lazos algebraicos pero puede eventualmente obtenerse la derivada del vector de estados $\dot{\mathbf{x}}$ sin modificar el sistema se dice que es una *DAE de índice 1*. Cuando el sistema no se puede causalizar sin modificarlo, si el sistema está bien formulado, se dice que es una *DAE de índice alto*.

El circuito RC de la Ec.(3.14) constituye una DAE de índice 2 ya que fue necesario aplicar una única vez el algoritmo de Pantelides para que nos quede un sistema de índice 1. Hay sistemas en los cuales es necesario aplicar Pantelides más de una vez. Si en un sistema debemos aplicar dos veces el algoritmo para llegar a una DAE de índice 1, entonces diremos que el sistema tiene índice 3, y así sucesivamente para los índices superiores.

Veamos entonces un ejemplo algo más complejo que el anterior, consistente en el modelo del péndulo de la Fig.3.4 dado por las ecuaciones diferenciales algebraicas de la Ec.(3.18), donde x e y representan las coordenadas de la masa, v_x y v_y las velocidades horizontal y vertical y $F(t)$ es la fuerza que realiza el cable en el sentido longitudinal.

$$\dot{x}(t) - v_x(t) = 0 \qquad (3.18a)$$
$$\dot{y}(t) - v_y(t) = 0 \qquad (3.18b)$$
$$m\ \dot{v}_x(t) + \frac{x(t)}{L}\ F(t) = 0 \qquad (3.18c)$$
$$m\ \dot{v}_y(t) + \frac{y(t)}{L}\ F(t) - m\ g = 0 \qquad (3.18d)$$
$$x(t)^2 + y(t)^2 - L^2 = 0 \qquad (3.18e)$$

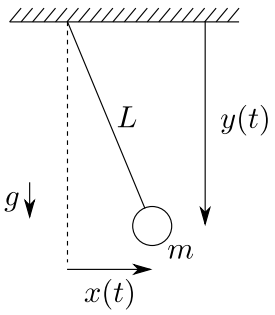

Figura 3.4: Péndulo

Tenemos en este caso 5 ecuaciones y 5 incógnitas, pero al igual que en el ejemplo anterior, la última ecuación no contiene a priori ninguna incógnita. Esto es nuevamente esperable ya que el sistema es claramente de segundo orden y lo estamos modelando con 4 potenciales variables de estado.

Procederemos entonces con el algoritmo de Pantelides derivando la ecuación de restricción, de donde se obtiene

$$\dot{\boldsymbol{x}}(\boldsymbol{t}) - v_x(t) = 0 \tag{3.19a}$$
$$\boldsymbol{dy}(\boldsymbol{t}) - v_y(t) = 0 \tag{3.19b}$$
$$m\ \dot{\boldsymbol{v}}_{\boldsymbol{x}}(\boldsymbol{t}) + \frac{x(t)}{L}\ \boldsymbol{F}(\boldsymbol{t}) = 0 \tag{3.19c}$$
$$m\ \dot{\boldsymbol{v}}_{\boldsymbol{y}}(\boldsymbol{t}) + \frac{\boldsymbol{y}(\boldsymbol{t})}{L}\ \boldsymbol{F}(\boldsymbol{t}) - m\ g = 0 \tag{3.19d}$$
$$x(t)^2 + \boldsymbol{y}(\boldsymbol{t})^2 - L^2 = 0 \tag{3.19e}$$
$$2\ x(t)\ \dot{\boldsymbol{x}}(\boldsymbol{t}) + 2\ \boldsymbol{y}(\boldsymbol{t})\ \boldsymbol{dy}(\boldsymbol{t}) = 0 \tag{3.19f}$$

Al derivar la ecuación de restricción, decidimos además que $y(t)$ no sea más variable de estado por lo que $dy(t)$ es una *derivada ficticia*.

La antigua restricción de la Ec.(3.19e) contiene ahora una incógnita $y(t)$, que según la Regla 1 del algoritmo de causalización deberemos despejar usando dicha ecuación. De manera similar, la Ec.(3.19a) contiene sólo la incógnita $\dot{x}(t)$, por lo que la deberemos despejar de dicha ecuación. Esto dejará a la nueva Ec.(3.19f) con una única incógnita $dy(t)$ que deberemos despejar desde allí. Luego, tras los tres primeros pasos del algoritmo de causalización tendremos lo siguiente:

$$\begin{aligned}
\dot{\boldsymbol{x}}(\boldsymbol{t}) - v_x(t) &:= 0 \\
dy(t) - v_y(t) &= 0 \\
m\ \dot{\boldsymbol{v}}_{\boldsymbol{x}}(\boldsymbol{t}) + \frac{x(t)}{L}\ \boldsymbol{F}(\boldsymbol{t}) &= 0 \\
m\ \dot{\boldsymbol{v}}_{\boldsymbol{y}}(\boldsymbol{t}) + \frac{y(t)}{L}\ \boldsymbol{F}(\boldsymbol{t}) - m\ g &= 0 \\
x(t)^2 + \boldsymbol{y}(\boldsymbol{t})^2 - L^2 &= 0 \\
2\ x(t)\ \dot{\boldsymbol{x}}(\boldsymbol{t}) + 2\ \boldsymbol{y}(\boldsymbol{t})\ \boldsymbol{dy}(\boldsymbol{t}) &= 0
\end{aligned}$$

$$\begin{aligned}
y(t) &:= \sqrt{L^2 - x(t)^2} \\
\dot{x}(t) &:= v_x(t) \\
dy(t) &:= -\frac{x(t)}{y(t)}\ \dot{x}(t)
\end{aligned}$$

Esta vez la ecuación que vincula $dy(t)$ con $v_y(t)$ no tiene más incógnitas, lo que constituye una ecuación de restricción en el nuevo sistema.

Aplicaremos nuevamente el algoritmo de Pantelides, derivando esta restricción y eliminando v_y de la lista de variables de estado, por lo que su derivada que era $\dot{v}_y$ será ahora la derivada ficticia dv_y.

La derivada de la restricción $dy(t) - v_y(t) = 0$ entonces queda

$$ddy(t) - dv_y(t) = 0 \tag{3.20}$$

donde $ddy(t)$ es la variable algebraica que representa la derivada de $dy(t)$.

Cada vez que derivemos una expresión y aparezca una derivada de una variable algebraica, deberemos luego buscar en las ecuaciones previamente causalizadas la definición de dicha variable algebraica. En este caso, tenemos

$$dy(t) = -\frac{x(t)}{y(t)}\ \dot{x}(t)$$

por lo que, derivando esta expresión obtenemos:

$$ddy(t) = -\left(\frac{\dot{x}(t)^2 + x(t)\ d\dot{x}(t) + dy(t)^2}{y(t)}\right) \tag{3.21}$$

En este caso nos apareció una nueva variable algebraica $d\dot{x}(t)$ que representa la derivada de $\dot{x}(t)$, por lo que tendremos que derivar también la expresión que define $\dot{x}(t)$:

$$\dot{x}(t) = v_x(t)$$

que resulta

$$d\dot{x}(t) = \dot{v}_x(t) \tag{3.22}$$

y ahora, agregando las Ecs.(3.20), (3.21) y (3.22) al sistema de la Ec.(3.19) tenemos un nuevo sistema:

$$\dot{\boldsymbol{x}}(\boldsymbol{t}) - v_x(t) = 0 \tag{3.23a}$$
$$\boldsymbol{dy}(\boldsymbol{t}) - \boldsymbol{v_y}(t) = 0 \tag{3.23b}$$
$$m\ \dot{\boldsymbol{v}}_{\boldsymbol{x}}(\boldsymbol{t}) + \frac{x(t)}{L}\ \boldsymbol{F}(\boldsymbol{t}) = 0 \tag{3.23c}$$
$$m\ \boldsymbol{dv_y}(\boldsymbol{t}) + \frac{\boldsymbol{y}(\boldsymbol{t})}{L}\ \boldsymbol{F}(\boldsymbol{t}) - m\ g = 0 \tag{3.23d}$$
$$x(t)^2 + \boldsymbol{y}(\boldsymbol{t})^2 - L^2 = 0 \tag{3.23e}$$
$$2\ x(t)\ \dot{\boldsymbol{x}}(\boldsymbol{t}) + 2\ \boldsymbol{y}(\boldsymbol{t})\ \boldsymbol{dy}(\boldsymbol{t}) = 0 \tag{3.23f}$$
$$\boldsymbol{ddy}(\boldsymbol{t}) - \boldsymbol{dv_y}(\boldsymbol{t}) = 0 \tag{3.23g}$$
$$\boldsymbol{ddy}(\boldsymbol{t}) + \left(\frac{\dot{\boldsymbol{x}}(\boldsymbol{t})^2 + x(t)\ \boldsymbol{d\dot{x}}(\boldsymbol{t}) + \boldsymbol{dy}(\boldsymbol{t})^2}{\boldsymbol{y}(\boldsymbol{t})}\right) = 0 \tag{3.23h}$$
$$\boldsymbol{d\dot{x}}(\boldsymbol{t}) - \dot{\boldsymbol{v}}_{\boldsymbol{x}}(\boldsymbol{t}) = 0 \tag{3.23i}$$

Antes de verificar si ahora el sistema puede causalizarse, eliminaremos las tres ecuaciones triviales y las variables algebraicas correspondientes ($dy(t)$, $ddy(t)$ y $d\dot{x}(t)$).

$$\dot{\boldsymbol{x}}(\boldsymbol{t}) - v_x(t) = 0 \tag{3.24a}$$
$$m\ \dot{\boldsymbol{v}}_{\boldsymbol{x}}(\boldsymbol{t}) + \frac{x(t)}{L}\ \boldsymbol{F}(\boldsymbol{t}) = 0 \tag{3.24b}$$
$$m\ \boldsymbol{dv_y}(\boldsymbol{t}) + \frac{\boldsymbol{y}(\boldsymbol{t})}{L}\ \boldsymbol{F}(\boldsymbol{t}) - m\ g = 0 \tag{3.24c}$$
$$x(t)^2 + \boldsymbol{y}(\boldsymbol{t})^2 - L^2 = 0 \tag{3.24d}$$
$$2\ x(t)\ \dot{\boldsymbol{x}}(\boldsymbol{t}) + 2\ \boldsymbol{y}(\boldsymbol{t})\ \boldsymbol{v_y}(\boldsymbol{t}) = 0 \tag{3.24e}$$
$$\boldsymbol{dv_y}(\boldsymbol{t}) + \left(\frac{\dot{\boldsymbol{x}}(\boldsymbol{t})^2 + x(t)\ \dot{\boldsymbol{v}}_{\boldsymbol{x}}(\boldsymbol{t}) + \boldsymbol{v_y}(\boldsymbol{t})^2}{\boldsymbol{y}(\boldsymbol{t})}\right) = 0 \tag{3.24f}$$

Si se aplica el procedimiento de causalización a este sistema, podrá verse que el mismo encontrará un lazo algebraico en 3 variables ($\dot{v}_x(t)$, $F(t)$, $dv_y(t)$) que puede causalizarse usando como variable de rasgado $\dot{v}_x(t)$, por ejemplo, de donde se obtiene:

$$\dot{x}(t) := v_x(t) \tag{3.25a}$$
$$y(t) := \pm\sqrt{L^2 - x^2} \tag{3.25b}$$
$$v_y(t) := \frac{x(t)}{y(t)}\ \dot{x}(t) \tag{3.25c}$$
$$dv_y(t) := -\left(\frac{\dot{x}(t)^2 + x(t)\ \dot{\boldsymbol{v}}_{\boldsymbol{x}}(\boldsymbol{t}) + v_y(t)^2}{y(t)}\right) \tag{3.25d}$$
$$F(t) := \frac{m\ L}{y(t)}\ (g - dv_y(t)) \tag{3.25e}$$
$$m\ \dot{\boldsymbol{v}}_{\boldsymbol{x}}(\boldsymbol{t}) + \frac{x(t)}{L}\ F(t) = 0 \tag{3.25f}$$

Dado que tuvimos que utilizar dos veces el algoritmo de Pantelides, este problema es de índice 3, algo que es muy habitual en los sistemas mecánicos con restricciones.

3.3.4. Elección de las Variables de Estado

Si observamos el sistema de la Ec.(3.25) podemos ver un par de problemas. En primer lugar, el signo de $y(t)$ no queda bien definido. En segundo lugar, $y(t)$ aparece dividiendo en varias ecuaciones. Mientras asumamos que el péndulo se puede mover sin alcanzar un ángulo de 90 grados respecto de la posición vertical, no habrá problemas ya que podemos tomar siempre la solución positiva para $y(t)$ y nunca ocurrirá que $y(t) = 0$.

Ahora bien, si hubiésemos elegido $y(t)$ como variable de estado en lugar de $x(t)$ tendríamos un problema más grave. En tal caso, el signo de $x(t)$ no estaría definido y eso sí es un problema cuando el péndulo se mueve cerca de la posición de reposo. Más aún, en lugar de divisiones por $y(t)$, tendríamos divisiones por $x(t)$ lo que aparejaría divisiones por cero cada vez que el péndulo queda en posición vertical.

Esto puede verse directamente en el modelo original de la Figura 3.4 donde es evidente que conocer el valor de $x(t)$ nos permite conocer $y(t)$ mientras que lo contrario no es cierto ya que hay dos soluciones. Además, con el péndulo en posición vertical no es posible calcular la aceleración horizontal $\dot{v}_x(t)$ a partir de conocer la aceleración vertical $\dot{v}_y(t)$ en el modelo original de la Ec.(3.18).

Todo esto delata que nuestra elección de $x(t)$ y de $v_x(t)$ como estados fue acertada. De otra forma, hubiéramos llegado a un modelo *correcto* desde el punto de vista del ordenamiento pero incapaz de ser simulado.

En este caso, una mejor opción hubiera sido incorporar el ángulo $\theta(t)$ a partir de la relación $x(t) = L\ \cos(\theta(t))$ y elegirlo como variable de estado dejando $x(t)$ e $y(t)$ como algebraicas. De esta manera desaparecerían los problemas de indeterminación entre $x(t)$ e $y(t)$

3.4. DAEs y Diagramas de Bloques

3.4.1. Diagramas de Bloques y Relaciones Causales

Los Diagramas de Bloques constituyen una herramienta para expresar relaciones matemáticas de manera gráfica. Una diferencia primordial entre las DAEs y los DBs es que los últimos, además de expresar relaciones matemáticas entre variables, asumen relaciones causales entre las mismas. Es decir, dada una relación

$$K\ a_1(t) - a_2(t) = 0$$

en un Diagrama de Bloques tenemos que asumir que $a_2(t) := K\ a_1(t)$ o bien que $a_2(t) := \frac{a_1(t)}{K}$. Estas opciones se traducen en los dos posibles Diagrama de Bloques de la Figura 3.5.

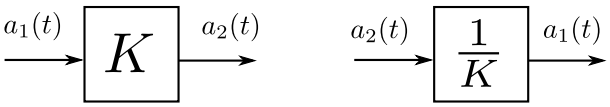

Figura 3.5: Dos posibles Diagramas de Bloques para la relación $K\ a_1(t) - a_2(t) = 0$.

Esta falta de unicidad en la representación de las relaciones matemáticas hace que los DBs no sean una herramienta muy adecuada para construir modelos. De hecho, asumir una determinada causalidad entre variables de un modelo puede provocar que luego no podamos *conectar* dicho modelo con otro. Por este motivo, las DAEs y los lenguajes acausales en general son las representaciones más adecuadas para las tareas de modelado.

Sin embargo, los DBs son una herramienta útil en muchos casos ya que permiten analizar de manera simple algunas propiedades importantes de los modelos que representan. Veremos entonces como construir de manera simple Diagramas de Bloques a partir de las DAEs y como en ese proceso de construcción podremos vislumbrar algunas características estructurales (como el índice de perturbación) del modelo.

3.4.2. Procedimiento para obtener un DB desde una DAE

El procedimiento para obtener un Diagrama de Bloques desde una DAE es muy simple y puede resumirse en los siguientes pasos:

1. Eliminar las ecuaciones triviales y los correspondientes alias (no necesario este paso, es sólo para simplificar).
2. Elegir una variable de la DAE para comenzar y colocar una flecha en el DB con dicha variable. Preferentemente comenzar por una variable de estado.
3. Tomar una variable del DB que no esté calculada por ningún bloque.
4. Buscar una ecuación de la DAE donde aparezca dicha variable teniendo en cuenta lo siguiente:
   - Si la variable es un estado $x_i(t)$ usar la ecuación $x_i(t) = \int \dot{x}_i(t)$.
   - Si la variable es la derivada de un posible estado $\dot{x}_i(t)$ y no quedara ninguna ecuación, usar la ecuación $\dot{x}_i(t) = \frac{\mathrm{d}}{\mathrm{d}t} x_i(t)$.
5. Agregar un bloque que calcule la variable en cuestión utilizando la ecuación elegida. Si se usó la ecuación $x_i(t) = \int \dot{x}_i(t)$ el bloque será un integrador mientras que si se usó $\dot{x}_i(t) = \frac{\mathrm{d}}{\mathrm{d}t} x_i(t)$ el bloque será un derivador.
6. Eliminar la ecuación utilizada de la DAE.
7. Si las variables de entrada al bloque agregado ya están calculadas en el DB como salida de otro bloque, conectarlas.
8. Si queda alguna variable en el DB que aún no es salida de ningún bloque, volver al paso 3.

Veremos entonces estos pasos sobre el modelo del circuito RLC serie de la Ec.(3.1), simplificado tras eliminar las ecuaciones triviales.

$$u_R(t) - R\ \dot{q}(t) = 0 \tag{3.26a}$$
$$q(t) - C\ u_C(t) = 0 \tag{3.26b}$$
$$\phi(t) - L\ \dot{q}(t) = 0 \tag{3.26c}$$
$$u_S(t) - v(t) = 0 \tag{3.26d}$$
$$\dot{\phi}(t) + u_R(t) + u_C(t) - u_S(t) = 0 \tag{3.26e}$$

Tomando entonces el estado $q(t)$ para comenzar y colocando un integrador nos queda $\dot{q}(t)$ como entrada del bloque. Utilizando la ecuación $\phi(t) - L\ \dot{q}(t) = 0$ para calcular $\dot{q}(t)$, nos queda $\phi(t)$ como entrada. El procedimiento completo puede verse en la Figura 3.6.

Para que el DB tenga esta forma tuvimos que elegir la ecuación $\phi(t) - L\ \dot{q}(t) = 0$ para calcular $\dot{q}(t)$. Si hubiésemos elegido la ecuación $u_R(t) - R\ \dot{q}(t) = 0$ nos habría quedado un derivador en lugar de un intergador en este caso. Esto no sería un problema a priori, pero muchas de las propiedades que pueden analizarse sobre los DBs requieren que el mismo contenga sólo integradores.

Una forma más sencilla de construir el DB es partiendo del sistema ya ordenado, como la Ec.(3.3), colocar un integrador por cada estado y luego seguir una por una las asignaciones obteniendo las restantes señales hacia adelante.

3.4.3. Diagramas de Bloques y Lazos Algebraicos

Consideremos nuevamente el ejemplo del circuito RL que contenía un lazo algebraico, dado por la Ec.(3.8). Eliminando las ecuaciones triviales y alias obtenemos:

$$\dot{\phi}(t) - R_1\ i_{R_1}(t) = 0 \tag{3.27a}$$
$$\dot{\phi}(t) - R_2\ i_{R_2}(t) = 0 \tag{3.27b}$$
$$\phi(t) - L\ i_L(t) = 0 \tag{3.27c}$$
$$i_L(t) + i_{R_1}(t) + i_{R_2}(t) = 0 \tag{3.27d}$$

Si seguimos el procedimiento para obtener el DB a partir de estas ecuaciones, podemos construir el Diagrama de Bloques de la Figura 3.7. En el mismo puede verse que hay un camino cerrado entre las

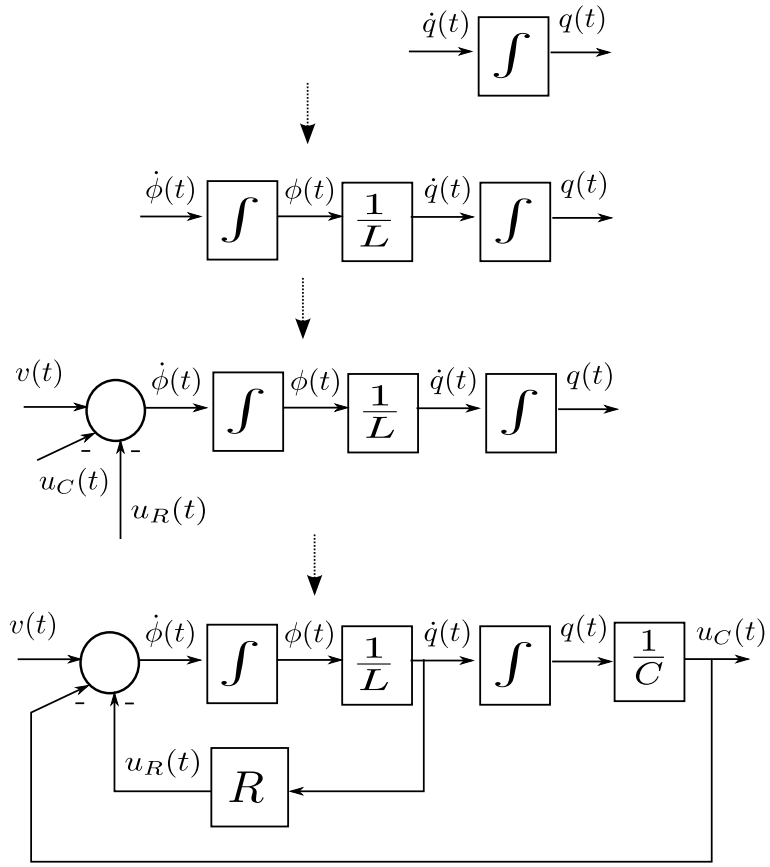

Figura 3.6: Construcción del DB a partir del sistema de DAEs de la Ec.(3.26).

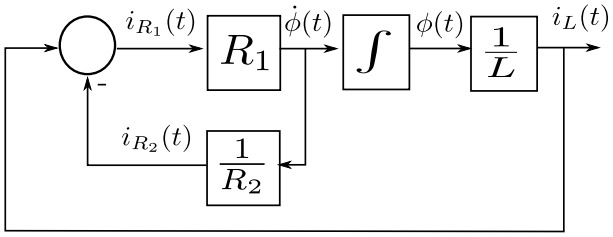

Figura 3.7: Diagrama de Bloques a partir de la Ec.(3.27).

variables $i_{R_1}(t)$, $i_{R_2}(t)$ y $\dot{\phi}(t)$ que no pasa por ningún integrador. Dicho camino refleja el lazo algebraico que habíamos detectado en la DAE correspondiente.

En casos generales, no sólo en este ejemplo, los lazos algebraicos de un sistema de DAEs se reflejan en la presencia de un camino cerrado a través de bloques estáticos en el Diagrama de Bloques.

3.4.4. Diagramas de Bloques y Sistemas de Índice Alto

Si construimos ahora un Diagrama de Bloques para el modelo del péndulo dado por la Ec.(3.18), podemos obtener el que se muestra en la Figura 3.8 donde los bloques con doble recuadro indican operaciones

no lineales. Estos bloques calculan las señales:

$$y(t) := \sqrt{L^2 - x(t)^2}$$
$$F(t) := \frac{L}{y(t)} \ (m\ g - m\ \dot{v}_y(t))$$
$$\dot{v}_x(t) := -\frac{x(t)\ F(t)}{m\ L}$$

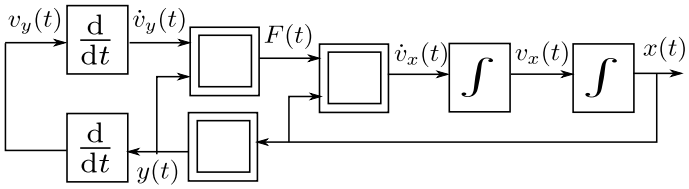

Figura 3.8: Diagrama de Bloques a partir de la Ec.(3.18).

En este caso, independientemente de las ecuaciones que elijamos para calcular cada variable, siempre necesitaremos bloques derivadores.

En las épocas de la computación analógica, la presencia de un derivador en un DB traía aparejados problemas de ruido, ya que las simulaciones se realizaban utilizando circuitos analógicos que emulaban los elementos de los diagramas de bloques. Si bien ese problema ya no existe, la presencia de derivadores hará difícil el análisis de ciertas propiedades. De hecho, la obtención de las Ecuaciones de Estado a partir de un DB como el de la Fig.3.8 es tan complicada como la de obtener dichas ecuaciones a partir de la DAE correspondiente de índice alto (ya sabemos que hay que usar un método de reducción de índice como el de Pantelides).

Más allá del ejemplo analizado, vale en general que las DAEs de índice alto implican que el DB correspondiente contendrá derivadores.

3.5. Herramientas de Software para Causalización de DAEs

Distintas variantes de los algoritmos que vimos en este capítulo se encuentran implementados en las distintas herramientas de software de modelado y simulación. Particularmente, estos algoritmos constituyen un pilar fundamental de las herramientas que utilizan el lenguaje Modelica.

Gracias a estos procedimientos podemos utilizar directamente los modelos de DAEs sin preocuparnos demasiado por la presencia de lazos algebraicos o singularidades estructurales.

Los compiladores de Modelica como Dymola, WofframSystemModeler y OpenModelica comienzan el procesamiento de los modelos con un procedimiento de *aplanado* en caso que el modelo utilice conexiones y otras características orientadas a objetos (que veremos más adelante en el Capítulo 5) obteniendo un conjunto de DAEs.

Luego, se eliminan las ecuaciones triviales y las variables algebraicas (*alias*) correspondientes. En el siguiente paso se realiza la causalización, que sigue en parte las ideas que vimos en este capítulo. Finalmente produce el código en lenguaje C con las ecuaciones del modelo causalizadas.

Este código puede ser utilizado por distintos solvers (DOPRI, DASSL, etc) de manera muy similar a lo que hicimos nosotros cuando invocábamos desde Octave o Matlab al método de Forward Euler.

Como vimos, un problema con los sistemas de índice alto es decidir qué variables se utilizarán como estados. Para esto, los compiladores de Modelica generan código con distintas opciones y luego, durante la simulación, deciden cuál es la mejor (evitando divisiones por cero, por ejemplo). Esto se denomina *selección dinámica de estados*. El lenguaje Modelica además brinda la posibilidad de decidir que una variable sea tratada como estado de manera obligatoria o preferencial, o lo contrario, que nunca sea considerada como estado.

Un modelo para el péndulo que refleja directamente la Ec.(3.18) en Modelica y que aclara que preferimos usar x como variable de estado cuando sea posible se muestra en el Código 3.9.

Código 3.9: Modelo de la Ec.(3.18) en Modelica

```
model Pendulum
 Real x(start=0.1,stateSelect=StateSelect.prefer),vx,y,vy,F;
 parameter Real L=1,m=1,g=9.8;
equation
  der(x)-vx=0;
  der(y)-vy=0;
  m*der(vx)+x/L*F=0;
  m*der(vy)+y/L*F-m*g=0;
  x^2+y^2-L^2=0;
end Pendulum;
```

Utilizando OpenModelica, a partir de este modelo el compilador genera un código con asignaciones casi idénticas a las de la Ec.(3.25) y la simulación funciona correctamente. Si cambiamos la opción `stateSelect=StateSelect.prefer` a la variable y, en cambio, aparecen distintos errores durante la simulación.

En OpenModelica pueden verse las ecuaciones causalizadas generadas mediante el *Transformational Debugger*.

3.6. Bibliografía Complementaria

Este capítulo, como mencionamos al comienzo del mismo, está basado en gran medida en el Capítulo 7 del libro *Continuous System Simulation* [CK06]. Una descripción mucho más detallada de los algoritmos y su implementación en Modelica puede encontrarse en el libro de Peter Fritzson *Principles of object-oriented modeling and simulation with Modelica 3.3: a cyber-physical approach* [Fri14].

Los algoritmos que vimos en el capítulo son en realidad versiones simplificadas y adaptadas de los originales, que se basan en *Teoría de Grafos*. En el procedimiento de causalización que se usa en las herramientas de software se comienza resolviendo un problema de *máximo matching* entre ecuaciones e incógnitas lo que permite detectar singularidades estructurales. Si las hubiera, se aplica el algoritmo de Pantelides en su versión original que utiliza análisis sobre el grafo de estructura [Pan88]. Una vez reducido el índice, se deben separar los distintos lazos algebraicos, lo que se logra detectando los *componentes fuertemente conexos* del grafo para lo que se utiliza generalmente el *Algoritmo de Tarjan* [Tar72] que resuelve a la vez el ordenamiento vertical de los sistemas de ecuaciones.

3.7. Problemas Propuestos

[P3.1] Rectificador de Media Onda

El circuito rectificador de la Figura 3.9 puede modelarse mediante las relaciones constitutivas y estructurales de la Ec.(P3.1a).

$$
\begin{aligned}
\dot{\phi}(t) - u_L(t) &= 0 \\
L\ i_L(t) - \phi(t) &= 0 \\
u_R(t) - R\ i_R(t) &= 0 \\
u_D(t) - g(i_D(t)) &= 0 \\
u_S(t) - U\ \sin(\omega\ t) &= 0 \\
i_L(t) - i_S(t) &= 0 \\
i_L(t) - i_R(t) &= 0 \\
i_L(t) - i_D(t) &= 0 \\
u_S(t) - u_R(t) - u_D(t) - u_L(t) &= 0
\end{aligned}
\qquad \text{(P3.1a)}
$$

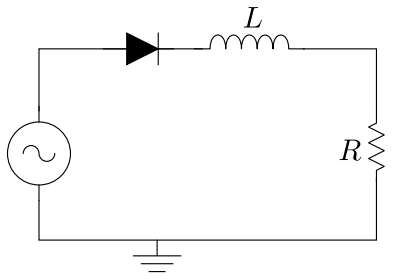

Figura 3.9: Rectificador de Media Onda con carga RL serie.

En las ecuaciones, la función $g(i_D(t))$ representa relación Volt-Ampère (no lineal) del diodo.
Se pide:

1. Enumerar las variables de estado y las variables algebraicas del modelo.
2. Eliminar todas las ecuaciones triviales posibles y las correspondientes variables algebraicas.
3. Aplicar el algoritmo de causalización, detectando la posible presencia de lazos algebraicos o singularidades estructurales.
4. Suponer que los parámetros son $R = 100\Omega$, $L = 0.001$Hy, $\omega = 2\ \pi\ 50$, $U = 220\sqrt{2}$V y que la función $g(i_D)$ está definida como

$$
g(i_D) = \begin{cases} R_{\text{on}}\ i_D & \text{si } i_D > 0 \\ R_{\text{off}}\ i_D & \text{en otro caso} \end{cases}
\qquad \text{(P3.1b)}
$$

   donde $R_{\text{on}} = 10^{-6}\Omega$ y $R_{\text{off}} = 10^{6}\Omega$. Con estos valores,

   a) Escribir una función de Octave o Matlab que represente las ecuaciones y simular el circuito usando Forward Euler y Backward Euler con un paso de integración $h = 0.001$ hasta un tiempo final $t_f = 0.1$.

   b) Implementar la simulación con OpenModelica y comparar los resultados. Una ley discontinua como la Ec.(P3.1b) puede implementarse como

```
u_D = if iD > 0 then Ron * iD else Roff * iD;
```

5. Escribir el modelo en Ecuaciones de Estado.

[P3.2] Rectificador de Media Onda II

El circuito rectificador de la Figura 4.2 es similar al del Problema P3.1 salvo que ahora tiene una resistencia en paralelo con el inductor. El mismo puede modelarse mediante las relaciones constitutivas y estructurales de la Ec.(P3.2a).

$$
\begin{aligned}
\dot{\phi}(t) - u_L(t) &= 0 \\
L\ i_L(t) - \phi(t) &= 0 \\
u_R(t) - R\ i_R(t) &= 0 \\
u_D(t) - g(i_D(t)) &= 0 \\
u_S(t) - U\ \sin(\omega\ t) &= 0 \\
u_{R_2}(t) - R_2\ i_{R_2}(t) &= 0 \\
i_{R_2}(t) - i_L(t) - i_R(t) &= 0 \\
i_S(t) - i_D(t) &= 0 \\
i_D(t) - i_{R_2}(t) &= 0 \\
u_S(t) - u_L(t) - u_D(t) - u_{R_2}(t) &= 0 \\
u_R(t) - u_L(t) &= 0
\end{aligned}
\qquad \text{(P3.2a)}
$$

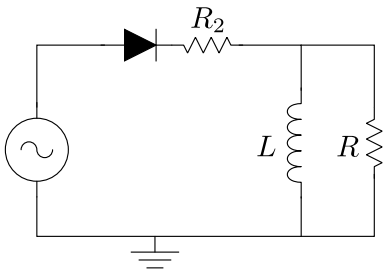

Figura 3.10: Rectificador de Media Onda con carga RL en paralelo.

Se pide entonces:

1. Repetir los primeros tres puntos del Problema P3.1.
2. Si hubiera un lazo algebraico, aplicar el procedimiento de *rasgado* tomando como variable de rasgado la tensión en el diodo $u_D(t)$ y reservando la ecuación constitutiva del diodo.
3. Repetir el punto anterior reservando la ecuación $u_S(t) - u_L(t) - u_D(t) - u_{R_2}(t)$. Analizar que ocurre en este caso si en la Ec.(P3.1b) se usa el parámetro $R_{\text{on}} = 0$.

[P3.3] Polea Simple

El esquema de la Figura 3.11 representa una masa suspendida mediante un resorte a través de una polea simple con momento de inercia J. El modelo matemático correspondiente puede caracterizarse por las Ecs.(P3.3a).

$$
\begin{aligned}
m\ \dot{v}_{y_m}(t) - F_m(t) &= 0 \\
\dot{y}_m(t) - v_{y_m}(t) &= 0 \\
k\ \Delta x_r(t) - F_r(t) &= 0 \\
J\ \dot{\omega}(t) - \tau(t) &= 0 \\
\dot{\theta}(t) - \omega(t) &= 0 \\
r\ F_1(t) - r\ F_2(t) - \tau(t) &= 0 \\
F_1(t) - F_r(t) &= 0 \\
F_2(t) - m\ g - F_m(t) &= 0 \\
r\ \theta(t) + \Delta x_r(t) &= 0 \\
\Delta x_r(t) + y_m(t) &= 0
\end{aligned}
\qquad \text{(P3.3a)}
$$

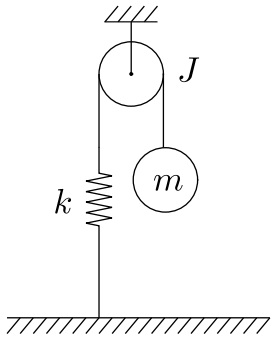

Figura 3.11: Polea Simple.

Se pide entonces

1. Repetir los tres primeros puntos del Problema P3.1 para este nuevo sistema y explicar la causa del índice alto en la DAE.
2. Aplicar el algoritmo de Pantelides hasta obtener un sistema de índice 1 y determinar el índice de perturbación de la DAE original.
3. Aplicar el procedimiento de rasgado y obtener un sistema explícito de Ecuaciones de Estado.

Bibliografía

[Ack71] Ackoff, Russell L: *Towards a system of systems concepts.* Management science, 17(11):661–671, 1971.

[CK06] Cellier, François y Ernesto Kofman: *Continuous System Simulation.* Springer, New York, 2006.

[DB08] Dorf, Richard C. y Robert H. Bishop: *Modern Control Systems.* Pearson Education, Inc., 11th edición, 2008.

[Fri14] Fritzson, Peter: *Principles of object-oriented modeling and simulation with Modelica 3.3: a cyber-physical approach.* John Wiley & Sons, 2014.

[Fri15] Fritzson, Peter: *Introducción al modelado y simulación de sistemas técnicos y físicos con modelica.* Linköping University Electronic Press, 2015.

[HNW00] Hairer, Ernst, Syvert Nørsett y Gerhard Wanner: *Solving Ordinary Differential Equations I: Nonstiff Problems.* Springer, Berlin, Germany, 2^{nd} edición, 2000.

[HW91] Hairer, Ernst y Gerhard Wanner: *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems.* Springer, Berlin, 1991.

[Jun15] Junco, Sergio: *Sistemas Dinámicos y Modelos Matemáticos*, 2015.

[Kof06] Kofman, Ernesto: *Simulación de Sistemas Continuos. Notas de Clase.* Departamento de Control, FCEIA, UNR. Disponible online en `http://www.fceia.unr.edu.ar/control/ssc/notas_ssc.pdf`, 2006.

[Kuo96] Kuo, Benjamin: *Sistemas de Control Automático. Séptima Edición.* Prentice Hall Hispanoamericana, México, 1996.

[Lju98] Ljung, Lennart: *System identification.* Springer, 1998.

[Pan88] Pantelides, Constantinos C: *The consistent initialization of differential-algebraic systems.* SIAM Journal on Scientific and Statistical Computing, 9(2):213–231, 1988.

[PTVF07] Press, William H., Saul A. Teukolsky, William T. Vetterling y Brian P. Flannery: *Numerical Recipes: The Art of Scientific Computing.* Cambridge University Press, New York, 3^{rd} edición, 2007.

[Sky05] Skyttner, Lars: *General systems theory: Problems, perspectives, practice.* World scientific, 2005.

[Tar72] Tarjan, Robert: *Depth-first search and linear graph algorithms.* SIAM journal on computing, 1(2):146–160, 1972.

[ZMK18] Zeigler, Bernard P, Alexandre Muzy y Ernesto Kofman: *Theory of modeling and simulation: discrete event & iterative system computational foundations.* Academic press, 2018.