Simulación de Sistemas Continuos.
Notas de Clase

Ernesto Kofman

Laboratorio de Sistemas Dinámicos
Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Universidad Nacional de Rosario

Índice general

Capítulo 1

Introducción a los Sistemas Continuos

El curso *Simulación de Sistemas Continuos* está basado casi en su totalidad en el libro *Continuous System Simulation* [2]. Este libro, si bien es áltamente autocontenido, presupone que el lector tiene ciertos conocimientos previos sobre modelos de sistemas continuos y propiedades de las ecuaciones diferenciales.

El objetivo de este capítulo es entonces el de introducir, de manera breve y algo informal, parte de estos conceptos que serán necesarios a lo largo del curso.

1.1. Sistemas Continuos y Ecuaciones Diferenciales.

Llamaremos *Sistemas Continuos* a los sistemas cuyas variables evolucionan continuamente en el tiempo. Los sistemas continuos se describen típicamente mediante *ecuaciones diferenciales*, ya sea ordinarias o en derivadas parciales.

Algunos ejemplos de sistemas continuos correspondientes a distintos dominios de aplicación son los siguientes:

Ejemplo 1.1. *Evolución de la concentración de una droga en el estómago [3].*

El modelo más simple de la evolución de la concentración de un fármaco en el estómago supone que la cantidad de droga que se absorve por unidad de tiempo es proporcional a la concentración de la misma.

Esto es, si llamamos $c_e(t)$ a la concetración de la droga en el estómago en el instante t, se cumplirá la ecuación:

$$\frac{\mathrm{d}c_e(t)}{\mathrm{d}t} = -r_a \cdot c_e(t) \tag{1.1}$$

Esta ecuación es una Ecuación Diferencial Ordinaria (ODE) de primer orden homogénea. La variable $c_e(t)$ se denomina *variable de estado.*

Además, se trata de un modelo lineal (el lado derecho de la ecuación contiene sólo operaciones lineales sobre c_e) y estacionario (si cambiamos la variable t por $\tilde{t} = t - \tau$ obtenemos exactamente el mismo sistema).

En general, utilizaremos la notación $\dot{x}$ para referirnos a la derivada temporal $\frac{\mathrm{d}x}{\mathrm{d}t}$. Es decir, reescribiremos la Ec.(1.1) como:

$$\dot{c}_e(t) = -r_a \cdot c_e(t) \tag{1.2}$$

Ejemplo 1.2. *Velocidad de una masa puntual en el aire.*

El esquema de la Fig.1.1 muestra el diagrama de fuerzas que actúan sobre una masa puntual bajo la acción de la gravedad y el rozamiento con el aire. Aplicando la segunda ley de Newton, se llega fácilmente a la ecuación:

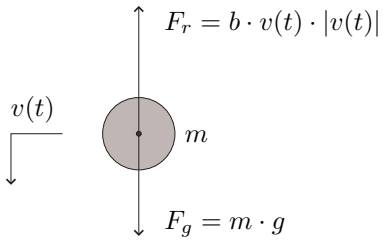

Figura 1.1: Sistema mecánico elemental.

$$\dot{v}(t) = g - \frac{b}{m} v(t)|v(t)| \tag{1.3}$$

que corresponde a un sistema de primer orden (la variable de estado es $v(t)$), no lineal (debido al término cuadrático de la fricción) y estacionario.

Ejemplo 1.3. *Sistema hidráulico.*

La Figura 1.2 representa un tanque de agua, cuya base tiene area A, a la que ingresa un líquido de densidad ρ. La cantidad de líquido que ingresa está dado por el caudal $Q(t)$ (una función del tiempo). El líquido acumulado en el tanque tiene un volumen $V(t)$ y como consecuencia de la altura de la columna de agua, la presión en el fondo del tanque es $P(t)$.

A la derecha del tanque hay una *estricción* que consideramos lineal. Esto es, el líquido que sale del tanque es proporcional a la presión $P(t)$.

Un modelo simple de este sistema está dado por:

$$\dot{V}(t) = -\frac{\rho \cdot g}{A \cdot R} V(t) + Q(t) \tag{1.4}$$

Este sistema es de primer orden (la variable de estado es el volumen $V(t)$), lineal y estacionario. Además, el sistema tiene una señal de *entrada* $Q(t)$.

Hay otras variables que pueden ser de interés en este sistemas, tales como la presión $P(t)$ o el caudal de salida $Q_s(t)$, que no aparecen explícitamente en

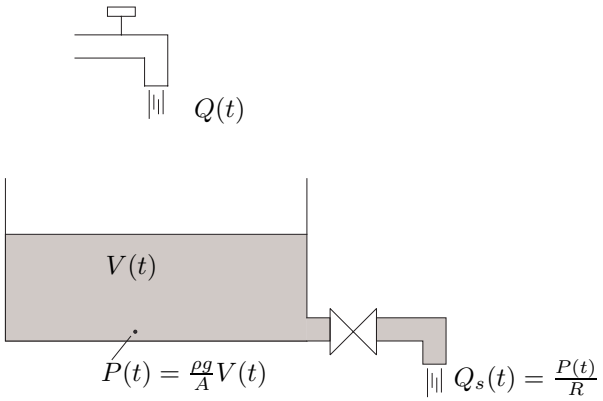

Figura 1.2: Sistema hidráulico elemental.

la ecuación diferencial (1.4). Por este motivo, los modelos en ecuaciones diferenciales suelen complementarse con *ecuaciones de salida* que calculan ciertas variables de interés (salidas) a partir de las variables de estado.

En este ejemplo podemos agregar las ecuaciones de salida:

$$P(t) = \frac{\rho \cdot g}{A} V(t)$$
$$Q_s(t) = \frac{\rho \cdot g}{A \cdot R} V(t)$$

Una observación muy importante es que el modelo dado por la ecuación (1.4) sólo es válido cuando $V(t) > 0$.

Ejemplo 1.4. *Evolución de la concentración de una droga entre el estómago y la sangre [3].*

El Ejemplo 1.1 puede completarse considerando ahora que la droga que se absorve del estómago pasa a la sangre, y que la cantidad de fármaco en la sangre se elimina también con una velocidad proporcional a su concentración.

Esta nueva hipótesis nos impone la necesidad de contar con una nueva variable de estados, ya que para describir lo que ocurre debemos tener en cuenta la concentración de fármaco en el estómago $c_e(t)$ y la concentración de fármaco en la sangre $c_s(t)$.

El sistema de ecuaciones diferenciales es entonces el siguiente:

$$\begin{aligned} \dot{c}_e(t) &= -r_a \cdot c_e(t) \\ \dot{c}_s(t) &= r_a \cdot c_e(t) - r_e \cdot c_s(t) \end{aligned} \tag{1.5}$$

Este sistema es lineal, de segundo orden (hay dos variables de estado), estacionario y autónomo (no hay entradas).

Ejemplo 1.5. *Sistema masa resorte.*

La Figura 1.3 muestra un sistema *masa–resorte*, donde se considera además la presencia de fricción (en este caso lineal) y de una fuerza externa aplicada sobre la masa $F(t)$.

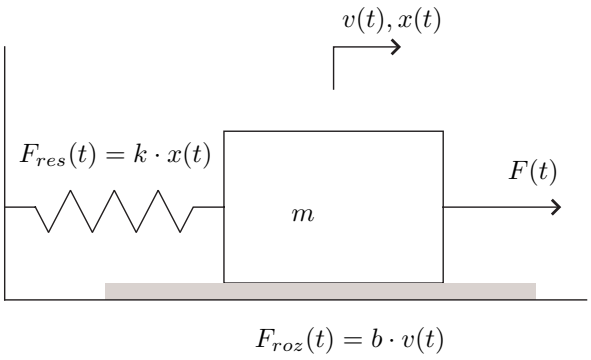

Figura 1.3: Sistema masa–resorte.

Un modelo de de la evolución de la posición $x(t)$ en este sistema (que se encuentra habitualmente en los cursos de Física) está dado por la ecuación diferencial

$$m \cdot \ddot{x}(t) + b \cdot \dot{x}(t) + k \cdot x(t) = F(t) \tag{1.6}$$

Aunque hay sólo una variable ($x(t)$), debido a la presencia de la derivada segunda, el sistema es en realidad de segundo orden.

En la Ec.(1.5) del ejemplo anterior representábamos al sistema de segundo orden como dos ecuaciones de primer orden en dos variables de estado (sistema de ecuaciones de estado). Evidentemente, la Ec.(1.6) corresponde a una forma distinta de representación.

Sin embargo, podemos fácilmente pasar a la forma de ecuaciones de estado utilizando como variables de estado a $x(t)$ y a $v(t) = \dot{x}(t)$, resultando el sistema:

$$\begin{aligned} \dot{x}(t) &= v(t) \\ \dot{v}(t) &= -\frac{k}{m}x(t) - \frac{b}{m}v(t) + \frac{1}{m}F(t) \end{aligned} \tag{1.7}$$

que es claramente un sistema lineal y estacionario, de segundo orden, con una entrada $F(t)$.

Los sistemas exresados como una ecuación diferencial con derivadas hasta orden n, tales como la Ecuación (1.6), pueden siempre reducirse a n sistemas de ecuaciones diferenciales de primer orden. Para esto basta con utilizar como variables de estado la variable de la ecuación diferencial original y sus $n-1$ derivadas.

En el curso vamos a trabajar casi exclusivamente con sistemas representados mediante ecuaciones de estado, es decir, n sistemas de ecuaciones diferenciales de primer orden.

Ejemplo 1.6. *Modelo de Lotka–Volterra [6].* Uno de los modelos más simple y difundidos de la dinámica de poblaciones es el de Lotka–Volterra, desarrollado en la década de 1920 para describir la evolución de dos especies: presa y depredador.

El modelo tiene la forma:

$$\begin{aligned} \dot{p}(t) &= r \cdot p(t) - a \cdot p(t) \cdot d(t) \\ \dot{d}(t) &= b \cdot p(t) \cdot d(t) - m \cdot d(t) \end{aligned} \tag{1.8}$$

donde $p(t)$ y $d(t)$ representan el número de presas y depredadores respectivamente.

El coeficiente r es la tasa de crecimiento del número de presas en ausencia de depredadores. El parámetro a es el coeficiente de depredación. Similarmente, b es la tasa de crecimiento del número de depredadores en presencia de presas y m es la tasa de mortalidad de los depredadores.

Como puede verse, se trata de un sistema no lineal de segundo orden.

Ejemplo 1.7. *Circuito Eléctrico.* El circuito eléctrico de la Fig. 1.4 consiste en una fuente de tensión $U(t)$, dos resistencias (R_1 y R_2), una inductancia L y un capacitor C. Consideramos todos elementos lineales.

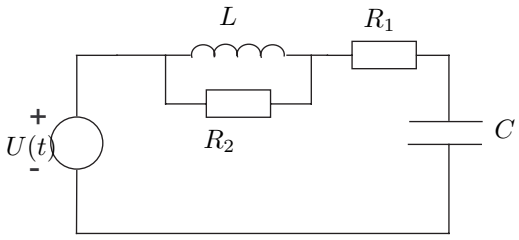

Figura 1.4: Circuito eléctrico simple.

Las ecuaciones que rigen la dinámica de dicho circuito son las siguientes:

$$\begin{aligned} \dot{u}_C(t) &= \frac{1}{C} i_L(t) + \frac{1}{R_2 C} u_L(t) \\ \dot{i}_L(t) &= \frac{1}{L} u_L(t) \end{aligned} \tag{1.9}$$

donde las variables de estado $u_C(t)$ e $i_L(t)$ representan la tensión en el capacitor y la corriente en la inductancia respectivamente. Además, aparece una variable $u_L(t)$ (tensión en la bobina) que debe cumplir:

$$U(t) - R_1 i_L(t) - u_C(t) - (1 + \frac{R_1}{R_2}) u_L(t) = 0 \tag{1.10}$$

En este caso, el sistema total, formado por las ecuaciones (1.9)–(1.10), es una *Ecuación Diferencial Algebraica* (DAE).

El sistema es lineal, de segundo orden, y en este caso (debido a la linealidad) puede transformarse en una ecuación diferencial ordinaria despejando $u_L(t)$ de (1.10) y reemplazando en (1.9). Sin embargo, si la característica de alguna de las resistencias fuera no lineal, esto sería en general imposible.

1.2. Modelado y Simulación.

Siendo este un curso sobre *simulación*, hay un problema que está íntimamente ligado al mismo y que no puede dejar de mencionarse: el *modelado*.

En el contexto de este curso, cuando hablamos de modelado nos referimos a la obtención de *modelos matemáticos* tales como los expresados por las Ecuaciones (1.2)–(1.10), a partir de descripciones similares a las de los ejemplos tratados.

Si bien este es un tema fundamental, lo dejaremos de lado y supondremos que ya contamos con los modelos y nos concentraremos casi exclusivamente en los problemas de simulación. Al lector que esté interesado en métodos de modelado podemos recomendarle el libro *Continuous System Modeling* [1], que contiene una muy buena síntesis de los métodos y herramientas para el modelado de sistemas de diversos dominios de la ciencia y de la técnica.

Cuando nos referimos a *simulación*, hablamos en general de un experimento que se realiza sobre un modelo.

En el caso de los sistemas continuos, dado que las variables evolucionan continuamente en el tiempo, hacer una simulación equivale a obtener datos sobre las trayectorias que describen dichas variables.

Más precisamente, lo que buscaremos obtener a través de las simulaciones son soluciones de las ecuaciones diferenciales que describen los sistemas a partir de condiciones iniciales conocidas. Esto no es ni más ni menos que lo que se conoce como *Problemas de Valor Inicial*.

1.3. Problemas de Valor Inicial.

El objetivo de este curso es el de estudiar métodos numéricos que permitan resolver de manera aproximada sistemas de ecuaciones tales como los que aparecieron en los Ejemplos 1.1–1.7 a partir de condiciones iniciales conocidas.

Sin embargo, algunas de las ecuaciones de estos casos particulares pueden resolverse de manera analítica.

Por ejemplo, la Ec.(1.2) del Ejemplo 1.1 tiene solución:

$$c_s(t) = e^{-r_a \cdot t} c_s(0) \tag{1.11}$$

donde $c_s(0)$ es la *condición inicial*, es decir, la cantidad de fármaco que hay originalmente en el estómago.

De manera similar, la Ec.(1.4) en el Ejemplo 1.3 tiene solución analítica

$$V(t) = e^{\lambda \cdot t} V(0) + \int_0^t e^{\lambda}(t-\tau) Q(\tau) d\tau \tag{1.12}$$

donde $\lambda \triangleq -\frac{\rho \cdot g}{A \cdot R}$. En este caso, $V(0)$ representa el volumen inicial de líquido en el tanque.

Veremos más adelante que en los sistemas lineales y estacionarios es relativamente facil encontrar la solución analítica. Sin embargo, esto no es en general

posible en los sistemas no lineales, y esta razón es una de las principales motivaciones para estudiar los métodos numéricos de integración.

Una característica de las soluciones analíticas (1.11) y (1.12) es que las fórmulas son válidas para cualquier condición inicial y, en el caso de las segunda, para cualquier trayectoria de entrada. Esta generalidad la perderemos cuando obtengamos soluciones numéricas, ya que cada solución numérica corresponderá a una determinada condición inicial y a una determinada trayectoria de entrada.

1.4. Ecuaciones de Estado

Los Ejemplos 1.1–1.6 resultaron en modelos de ecuaciones diferenciales ordinarias, que escribimos en la forma de *Ecuaciones de Estado*. En general, un sistema de ecuaciones de estado de orden n tiene la forma:

$$\begin{aligned} \dot{x}_1(t) &= f_1(x_1(t), \ldots, x_n(t), u_1(t), \ldots, u_m(t), t) \\ &\vdots \\ \dot{x}_n(t) &= f_n(x_1(t), \ldots, x_n(t), u_1(t), \ldots, u_m(t), t) \end{aligned} \tag{1.13}$$

donde $x_1, \ldots, x_n$ son las variables de estado y $u_1, \ldots, u_m$ son las variables de entrada. La Ec.(1.13) puede reescribirse utilizando notación vectorial:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \tag{1.14}$$

donde $\mathbf{x} = \begin{bmatrix} x_1 & \ldots & x_n \end{bmatrix}^T \in \mathbb{R}^n$ es el vector de estados y $\mathbf{u} = \begin{bmatrix} u_1 & \ldots & u_m \end{bmatrix}^T \in \mathbb{R}^m$ es el vector de entradas.

Cuando interesan algunas variables del sistema en particular (no necesariamente variables de estado), tal como en el caso del Ejemplo 1.3, suelen completarse los modelos con Ecuaciones de Salida de la forma:

$$\begin{aligned} y_1(t) &= g_1(x_1(t), \ldots, x_n(t), u_1(t), \ldots, u_m(t), t) \\ &\vdots \\ y_p(t) &= g_p(x_1(t), \ldots, x_n(t), u_1(t), \ldots, u_m(t), t) \end{aligned} \tag{1.15}$$

donde $y_1, \ldots, y_p$ son denominadas *variables de salida*. Las ecuaciones de salida también pueden escribirse en forma vectorial:

$$\mathbf{y}(t) = \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t), t) \tag{1.16}$$

1.5. Sistemas Lineales y Estacionarios.

Un caso particular de la Ecuación de Estados (1.14) se da cuando el lado derecho es lineal en los estados y entradas y no hay dependencia explícita del tiempo. En tales casos, la Ec.(1.14) toma la forma

$$\dot{\mathbf{x}}(t) = \mathbf{A} \cdot \mathbf{x}(t) + \mathbf{B} \cdot \mathbf{u}(t) \tag{1.17}$$

donde $\mathbf{A} \in \mathbb{R}^{n \times n}$ y $\mathbf{B} \in \mathbb{R}^{n \times m}$ son las matrices de evolución y de entrada respectivamente.

Para los fines de este curso, nos interesa particularmente el caso autónomo, es decir,

$$\dot{\mathbf{x}}(t) = \mathbf{A} \cdot \mathbf{x}(t) \tag{1.18}$$

ya que las principales propiedades de estabilidad y precisión de los distintos métodos que estudiaremos se analizan en base a este caso.

La razón de utilizar el sistema (1.18) para analizar las propiedades de los métodos es muy simple: para este sistema podemos calcular la solución analítica y podemos establecer de manera muy simple las condiciones que debe cumplir $\mathbf{A}$ para que el sistema sea estable. Por lo tanto, podemos comparar las soluciones aproximadas y las características de estabilidad que resultan de aplicar un método con las soluciones y las condiciones de estabilidad exactas del sistema original.

En virtud de esto, analizaremos a continuación las propiedades del sistema lineal y estacionario (1.18).

Comenzaremos entonces por resolver la ecuación (1.18) a partir de una condición inicial genérica $\mathbf{x}(0) = \mathbf{x}_0$. Para tal fin, comenzaremos haciendo un cambio de variables a través de una transformación lineal:

$$\mathbf{x}(t) = \mathbf{V} \cdot \mathbf{z}(t) \tag{1.19}$$

donde $\mathbf{V}$ es una matriz invertible. Reemplazando en (1.18) y operando obtenemos el sistema:

$$\dot{\mathbf{z}}(t) = \mathbf{V}^{-1}\mathbf{A} \cdot \mathbf{V} \cdot \mathbf{z}(t) = \mathbf{\Lambda} \cdot \mathbf{z}(t) \tag{1.20}$$

con la condición inicial $\mathbf{z}(0) = \mathbf{z}_0 = \mathbf{V}^{-1} \cdot \mathbf{x}_0$.

Cuando todos los autovalores de la matriz $\mathbf{A}$ son distintos, es siempre posible elegir $\mathbf{V}$ tal que la matriz $\mathbf{\Lambda}$ sea diagonal[1]. En ese caso, la matrices $\mathbf{\Lambda}$ (matriz de autovalores) y $\mathbf{V}$ (matriz de autovectores) toman la forma

$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n \end{bmatrix}, \quad \mathbf{V} = \begin{bmatrix} \mathbf{V}_1 & \mathbf{V}_2 & \dots & \mathbf{V}_n \end{bmatrix} \tag{1.21}$$

donde los λ_i son los autovalores y los $\mathbf{V}_i$ los correspondientes autovectores.

Recordemos que los autovalores son soluciones de la ecuación característica

$$\det(\lambda \cdot I - A) = 0 \tag{1.22}$$

y los autovectores satisfacen

$$\lambda_i \cdot \mathbf{V}_i = \mathbf{A} \cdot \mathbf{V}_i \tag{1.23}$$

[1] Cuando hay autovalores repetidos, en general no se puede diagonalizar, pero se puede elegir $\mathbf{V}$ tal que la matriz $\mathbf{\Lambda}$ sea una matriz de *Jordan*. De todas maneras, para los fines de este curso, no será necesario tener en cuenta estos casos no diagonalizables

Teniendo en cuenta la Ec.(1.21), el sistema (1.20) puede reescribirse como

$$\begin{aligned} \dot{z}_1(t) &= \lambda_1 \cdot z_1(t) \\ &\vdots \\ \dot{z}_n(t) &= \lambda_n \cdot z_n(t) \end{aligned} \tag{1.24}$$

lo que consiste en un sistema de n ecuaciones diferenciales de primer orden desacopladas, y que puede resolverse de manera trivial. La solución es

$$\begin{aligned} z_1(t) &= e^{\lambda_1 \cdot t} \cdot z_1(0) \\ &\vdots \\ z_n(t) &= e^{\lambda_n \cdot t} \cdot z_n(0) \end{aligned} \tag{1.25}$$

Utilizando notación matricial, podemos reescribir la solución

$$\mathbf{z}(t) = e^{\mathbf{\Lambda} \cdot t} \cdot \mathbf{z}(0) \tag{1.26}$$

donde definimos

$$e^{\mathbf{\Lambda} \cdot t} \triangleq \begin{bmatrix} e^{\lambda_1 \cdot t} & 0 & \dots & 0 \\ 0 & e^{\lambda_2 \cdot t} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & e^{\lambda_n \cdot t} \end{bmatrix} \tag{1.27}$$

Finalmente, utilizando la Ecuación (1.19) en la Ec.(1.26), y recordando que $\mathbf{z}(0) = \mathbf{V}^{-1} \cdot \mathbf{x}(0)$, podemos obtener la solución de la Ec.(1.18):

$$\mathbf{x}(t) = \mathbf{V} \cdot e^{\mathbf{\Lambda} \cdot t} \cdot \mathbf{V}^{-1} \mathbf{x}(0) = e^{\mathbf{A} \cdot t} \cdot \mathbf{x}(0) \tag{1.28}$$

donde definimos

$$e^{\mathbf{A} \cdot t} \triangleq \mathbf{V} \cdot e^{\mathbf{\Lambda} \cdot t} \cdot \mathbf{V}^{-1} \tag{1.29}$$

La matriz $e^{\mathbf{A} \cdot t}$ se denomina *matriz de transición*, y permite calcular la solución de (1.18) en un instante cualquiera. Esta denominación se debe a que si conocemos el valor del estado $\mathbf{x}(t_1)$ en un instante cualquiera t_1, al premultiplicar dicho estado por la matriz de transición $e^{\mathbf{A} \cdot t}$ obtenemos el valor del estado en el instante $t + t_1$.

En presencia de señales de entrada habíamos visto que un sistema lineal y estacionario tomaba la forma de la Ec.(1.17). En tal caso, la solución puede escribirse como

$$\mathbf{x}(t) = e^{\mathbf{A} \cdot t} \cdot \mathbf{x}(0) + \int_0^t e^{\mathbf{A} \cdot (t-\tau)} \cdot \mathbf{B} \cdot \mathbf{u}(\tau) \mathrm{d}\tau \tag{1.30}$$

1.6. Propiedades Cualitativas de la Solución.

Al analizar la solución $\mathbf{x}(t)$ en la Ec.(1.30), encontraremos que cada componente del vector de estado verifica la expresión:

$$x_i(t) = c_{i,1} \cdot e^{\lambda_1 \cdot t} + \ldots + c_{i,n} \cdot e^{\lambda_n \cdot t} \tag{1.31}$$

donde los coeficientes $c_{i,j}$ dependen del estado inicial $\mathbf{x}(0)$ y de la matriz de autovectores $\mathbf{V}$.

Independientemente de cuanto valgan estos coeficientes, la forma *cualitativa* de las soluciones dependerá totalmente de los autovalores.

Por este motivo, las funciones (exponenciales) $e^{\lambda_j \cdot t}$ se denominan *modos* del sistema. Por ejemplo, un autovalor real negativo aportará un modo correspondiente a una exponencial convergente. De manera similar, un par de autovalores imaginarios puros (conjugados) aportarán un modo oscilante (senoidal puro). Por otro lado, un par de autovalores complejos conjugados con parte real positiva aportarán un modo oscilante *modulado* por una exponencial divergente.

Una diferenciación fundamental entre los posibles modos de un sistema está dada por la parte real de los autovalores. Si $\mathbb{R}\mathrm{e}(\lambda_i) < 0$ es fácil de ver que $\lim_{t\to\infty} e^{\lambda_i \cdot t} = 0$, es decir, el modo se extingue a medida que avanza el tiempo. Esto es, se trata de un modo *estable*.

En cambio, cuando $\mathbb{R}\mathrm{e}(\lambda_i) > 0$ ocurre que $\lim_{t\to\infty} |e^{\lambda_i \cdot t}| = \infty$, por lo que se trata de un modo *inestable*.

El caso $\mathbb{R}\mathrm{e}(\lambda_i) = 0$ (suponiendo que no hay autovalores repetidos)[2] corresponde a modos *marginalmente estables*.

Para que un sistema lineal y estacionario sea *estable* todos los modos deben ser estables, es decir, todos los autovalores deben tener parte real negativa. Por el contrario, si uno o más autovalores tienen parte real positiva el sistema será inestable.

Es importante remarcar que no hace falta calcular la solución $\mathbf{x}(t)$ para tener información cualitativa sobre la solución, ya que para conocer los modos del sistema y/o la estabilidad es suficiente con calcular los autovalores de la matriz $\mathbf{A}$.

Por último, cabe mencionar que hay conceptos y herramientas de análisis de estabilidad (más avanzados que los vistos) que pueden aplicarse no sólo a los sistemas lineales y estacionarios, tales como (1.17), sino que también se aplican a sistemas no lineales en general como (1.14). Incluso, algunos de ellos se aplican en el contexto de los métodos numéricos de integración. Sin embargo, no trataremos aquí esos conceptos ya que exceden los objetivos de este curso. Para quien tenga interés en el tema, el libro *Nonlinear Systems* de H.Khalil [5] ofrece un muy buen resumen de las principales herramientas de análisis de sistemas no lineales.

[2] Cuando hay autovalores repetidos y el sistema no se puede diagonalizar, aparecen modos de la forma $t^j \cdot e^{\lambda_i \cdot t}$, donde $j = 1, \ldots, k$, siendo k la *multiplicidad* del autovalor.

1.7. Problemas Propuestos

[P1.1] Respuesta al escalón de un sistema de primer orden

Calcular y graficar la solución del sistema (1.4) considerando un volumen inicial nulo $V(0) = 0$, y que el caudal de entrada es una función escalón unitario, es decir, $Q(t) = \varepsilon(t)$ donde

$$\varepsilon(t) = \begin{cases} 0 & \text{si } t < 0 \\ 1 & \text{si } t \geq 0 \end{cases} \tag{P1.1a}$$

Utilizar los parámetros: $\rho = 1$, $g = 9.8$, $A = 1$ y $R = 1$.

[P1.2] Solución libre de un sistema de segundo orden

Para el sistema (1.5):

a) Reescribir la ecuación en forma matricial.

b) Calcular y graficar la solución con condiciones iniciales $c_e(0) = 1$, $c_s(0) = 0$, considerando los parámetros $r_a = 2$ y $r_e = 1$.

c) Repetir el punto anterior los parámetro $r_a = 20$, $r_e = 1$ y para $r_a = 0.1$, $r_e = 1$.

[P1.3] Respuesta al escalón de un sistema de segundo orden Para el sistema masa resorte (1.7):

a) Reescribir la ecuación en forma matricial.

b) Calcular y graficar la solución considerando condiciones iniciales nulas y que la fuerza de entrada es un escalón unitario, $F(t) = \varepsilon(t)$, con $\varepsilon(t)$ definido en la Ec.(P1.1a). Suponer que los parámetros valen $m = k = b = 1$.

[P1.4] Transformación de una DAE en una ODE explícita. Convertir la DAE (1.9)–(1.10) en una ecuación diferencial ordinaria y escribirla en forma matricial.

[P1.5] Estabilidad de los sitemas lineales y estacionarios

Analizar la estabilidad de los sistemas de los Ejemplos 1.1, 1.3, 1.4, 1.5 y 1.7. Considerar en todos los casos que todos los parámetros son positivos.

[P1.6] Punto de equilibrio de un sistema lineal y estacionario Para el sistema hidráulico (1.4), considerando $Q(t) = \bar{Q})$ (constante) y los parámetros utilizados en el Problema P1.1, calcular el *punto de equilibrio*. Es decir, se pide calcular el valor del estado (en este caso $V(t)$) para el cual la solución queda en un estado estacionario ($\dot{V}(t) = 0$).

Corroborar luego este resultado con la solución del sistema calculada en el Problema P1.1.

[P1.7] Puntos de equilibrio en sistemas no lineales Calcular el/los punto/s de equilibrio de los sistemas:

a) El sistema (1.3), considerando $b = m = 1$, $g = 9.8$.

b) El sistema presa–depredador (1.8), suponiendo $r = a = b = m = 0.1$.

Interpretar los resultados.

[P1.8] Pelota rebotando El siguiente modelo puede verse como una combinación del sistema (1.3) (suponiendo rozamiento lineal, y con $v(t)$ positiva hacia arriba) y el sistema (1.7) (con $F(t) = -m \cdot g$):

$$\begin{aligned} \dot{x}(t) &= v(t) \\ \dot{v}(t) &= \begin{cases} -\frac{b_a}{m} \cdot v(t) - g & \text{si } x(t) > 0 \\ -\frac{k}{m} \cdot x(t) - \frac{b}{m} \cdot v(t) - g & \text{si } x(t) \leq 0 \end{cases} \end{aligned} \tag{P1.8a}$$

Este sistema puede representar un modelo muy simplificado de una pelota que rebota contra el piso. Cuando $x(t) > 0$ la pelota está en el aire y cuando $x(t) \leq 0$ la pelota está en contacto con el piso y se modeliza como un sistema *masa–resorte*.

Desde el punto de vista matemático, la ecuación (7.1) es *seccionalmente lineal* ó *lineal a tramos*. Si bien no se puede calcular la solución analítica en forma exacta, es posible calcular tramos de la solución.

Considerando entonces los parámetros $b_a = 0.1$, $m = 1$, $b = 30$, $g = 9.8$ y $k = 100000$, se pide:

a) Calcular y graficar un tramo de la solución desde la condición inicial $x(0) = 10$, $v(0) = 0$, hasta el tiempo t_1, en el que se verifique $x(t_1) = 0$.

   Ayuda: Como la matriz A es singular, en este caso no puede calcularse la integral de $e^{A \cdot t}$ como $A^{-1} \cdot e^{A \cdot t}$. Una opción (para seguir utilizando esta forma de integrar) es modificar la matriz A, agregando un pequeño término que evite que sea singular.

   Además, notar que para calcular el tiempo t_1 va a ser necesario utilizar alguna forma de iteración.

b) Utilizando como condición inicial los valores finales del tramo obtenido en el punto anterior: $x(t_1)(= 0)$, $v(t_1)$, calcular un nuevo tramo de la solución hasta un instante t_2 en que se verifique nuevamente $x(t_2) = 0$.

c) Repetir el punto anterior varias veces y graficar la solución obtenida.

Capítulo 2

Bases de la Integración Numérica

Este capítulo es una traducción resumida del Capítulo 2 del libro *Continuous System Simulation* [2].

2.1. Introducción

Consideremos un modelo de ecuaciones de estado:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \tag{2.1}$$

donde $\mathbf{x}$ es el *vector de estados*, $\mathbf{u}$ es el *vector de entradas*, y t representa el tiempo, con condiciones iniciales:

$$\mathbf{x}(t = t_0) = \mathbf{x_0} \tag{2.2}$$

Una componente $x_i(t)$ del vector de estados representa la i^{th} trayectoria del estado en función del tiempo t. Siempre y cuando el modelo de ecuaciones de estado no contenga discontinuidades en $f_i(\mathbf{x}, \mathbf{u}, t)$ ni en sus derivadas, $x_i(t)$ será también una función continua. Además, la función podrá aproximarse con la precisión deseada mediante series de Taylor alrededor de cualquier punto de la trayectoria (siempre y cuando no haya escape finito, es decir, que la trayectoria tienda a infinito para un valor finito de tiempo).

Llamemos t^* al instante de tiempo en torno al cual queremos aproximar la trayectoria mediante una serie de Taylor, y sea $t^* + h$ el instante de tiempo en el cual queremos evaluar la aproximación. Entonces, la trayectoria en dicho punto puede expresarse como sigue:

$$x_i(t^* + h) = x_i(t^*) + \frac{dx_i(t^*)}{dt} \cdot h + \frac{d^2x_i(t^*)}{dt^2} \cdot \frac{h^2}{2!} + \dots \tag{2.3}$$

Reemplazando con la ecuación de estado (2.1), la serie (2.3) queda:

$$x_i(t^* + h) = x_i(t^*) + f_i(t^*) \cdot h + \frac{df_i(t^*)}{dt} \cdot \frac{h^2}{2!} + \dots \tag{2.4}$$

Los distintos algoritmos de integración difieren en la manera de aproximar las derivadas superiores del estado y en el número de términos de la serie de Taylos que consideran para la aproximación.

2.2. Precisión de la aproximación

Evidentemente, la precisión con la que se aproximan las derivadas de orden superior debe estar acorde al número de términos de la serie de Taylos que se considera. Si se tienen en cuenta $n + 1$ términos de la serie, la precisión de la aproximación de la derivada segunda del estado $d^2x_i(t^*)/dt^2 = df_i(t^*)/dt$ debe ser de orden $n - 2$, ya que este factor se multiplica por h^2. La precisión de la tercer derivada debe ser de orden $n - 3$ ya que este factor se multiplica por h^3, etc. De esta forma, la aproximación será correcta hasta h^n. Luego, n se denomina *orden de la aproximación* del método de integración, o, simplemente, se dice que el método de integración es de orden n.

El error de aproximación que se produce a causa del truncamiento de la serie de Taylor tras un número finito de términos se denomina *error de truncamiento*. El error de truncamiento contiene términos de h^{n+1}, h^{n+2}, etc. No puede contener ningún término de potencias de h menor que $n+1$. Sin embargo, como la magnitud de los términos superiores decrece muy rápidamente, el error de truncamiento se suele aproximar mediante el término de h^{n+1}.

Mientras mayor es el orden de un método, más precisa es la estimación de $x_i(t^* + h)$. En consecuencia, al usar métodos de orden mayor, se puede integrar utilizando pasos grandes. Por otro lado, al usar pasos cada vez más chicos, los términos de orden superior de la serie de Taylor decrecen cada vez más rápidos y la serie de Taylor puede truncarse antes.

El costo de cada paso depende fuertemente del orden del método en uso. En este sentido, los algoritmos de orden alto son mucho más costosos que los de orden bajo. Sin embargo, este costo puede compensarse por el hecho de poder utilizar un paso mucho mayor y entonces requerir un número mucho menor de pasos para completar la simulación. Esto implica que hay que hacer una buscar una solución de compromiso entre ambos factores.

Una regla empírica simple para elegir el orden apropiado a utilizar en una simulación puede ser la siguiente:

> Si la precisión local relativa requerida por una aplicación, es decir, el máximo error tolerado tras un paso de integración, es 10^{-n}, entonces conviene elegir un algoritmo de al menos orden n.

Por este motivo, la simulación de problemas de dinámica celeste requiere usar los algoritmos de mayor orden (normalmente se usan de orden 8). Por otro lado, la mayor parte de las simulaciones de sistemas económicos se hacen con métodos

de primer o segundo orden, ya que los parámetros de los modelos no son mucho más precisos que eso.

Muchas aplicaciones de la ingeniería requieren una precisión global relativa de aproximadamente 0.001. Normalmente se asume lo siguiente:

> Si el error de integración local, es decir, el error cometido tras un paso de integración es proporcional a h^{n+1}, luego el error de integración global, es decir, el error de los resultados al final de la simulación, es proporcional a h^n.

Esta suposición es correcta para un paso de integración h suficientemente pequeño, es decir, en la llamada *región asintótica* del algoritmo.

Un error relativo global de 0.001, lo que permiten la mayor parte de las aplicaciones de la ingeniería, requiere un algoritmo con un error global de h^3. De acuerdo con la observación anterior, esto corresponde a un algoritmo con error local de h^4 y error relativo local de 0.0001. Por esto la mayor parte de los sistemas de simulación ofrecen como algoritmo por defecto uno de cuarto orden.

Una segunda fuente de error de aproximación importante es la causada por la longitud de palabra finita de la computadora que realiza la simulación. Este error se denomina *error de redondeo* y es importante ya que en la integración numérica siempre se suman números pequeños con números grandes.

Debido a esto, en general el uso de algoritmo de orden alto requiere a la vez el uso de doble precisión, ya que de otra forma el error por redondeo se torna mayor al error por truncamiento y pierde sentido la utilización de una aproximación tan precisa. Además, por el mismo motivo, casi nunca se utilizan algoritmos de orden mayor que 8.

Una tercera fuente de error es el *error de acumulación*. Debido al truncamiento y al redondeo no se puede conocer con precisión $x(t^* + h)$. Por lo tanto, este error es heredado por el siguiente paso como un error en la condición inicial. Así, los errores se acumulan a través de pasos sucesivos. Afortunadamente, los efectos de las condiciones iniciales tienden a desaparecer de la solución analítica cuando el sistema es analíticamente estable. Por lo tanto, es de esperar que si el método de integración es numéricamente estable (ya veremos en detalle el significado de este término) entonces también se extinguirá con el tiempo el efecto de las condiciones iniciales y con esto, se atenuará el efecto de la imprecisión en las mismas.

Sin embargo, esto no ocurre así con los sistemas inestables, y el error aquí puede acumularse excesivamente a través de los pasos. En estos casos, puede ser conveniente (si es posible) integrar hacia atrás en el tiempo desde el final.

Por encima de estos errores aparecen también las imprecisiones del propio modelo. Estas se pueden clasificar entre *errores paramétricos*, es decir, imprecisión en los parámetros, y *errores estructurales*, también llamados *dinámica no modelada*.

2.3. Integración de Euler

El algoritmo de integración más simple se obtiene truncando la serie de Taylor tras el término lineal:

$$\mathbf{x}(t^* + h) \approx \mathbf{x}(t^*) + \dot{\mathbf{x}}(t^*) \cdot h \tag{2.5a}$$

o:

$$\mathbf{x}(t^* + h) \approx \mathbf{x}(t^*) + \mathbf{f}(\mathbf{x}(t^*), t^*) \cdot h \tag{2.5b}$$

Este esquema es particularmente simple ya que no requiere aproximar ninguna derivada de orden superior, y el término lineal está directamente disponible del modelo de ecuaciones de estado. Este esquema de integración se denomina *Método de Forward Euler*, y lo abreviaremos como FE.

La Fig.2.1 muestra una interpretación gráfica de la aproximación realizada por el método de FE.

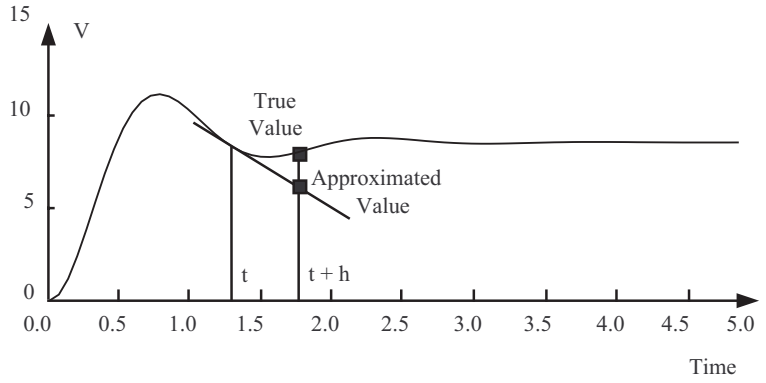

Figura 2.1: Integración numérica utilizanddo Forward Euler.

La simulación utilizando el método de FE es trivial. Dada la condición inicial $\mathbf{x}(t = t_0) = \mathbf{x_0}$ se puede proceder de la siguiente forma:

paso 1a:	$\dot{\mathbf{x}}(t_0)$	$= \mathbf{f}(\mathbf{x}(t_0), t_0)$
paso 1b:	$\mathbf{x}(t_0 + h)$	$= \mathbf{x}(t_0) + h \cdot \dot{\mathbf{x}}(t_0)$
paso 2a:	$\dot{\mathbf{x}}(t_0 + h)$	$= \mathbf{f}(\mathbf{x}(t_0 + h), t_0 + h)$
paso 2b:	$\mathbf{x}(t_0 + 2h)$	$= \mathbf{x}(t_0 + h) + h \cdot \dot{\mathbf{x}}(t_0 + h)$
paso 3a:	$\dot{\mathbf{x}}(t_0 + 2h)$	$= \mathbf{f}(\mathbf{x}(t_0 + 2h), t_0 + 2h)$
paso 3b:	$\mathbf{x}(t_0 + 3h)$	$= \mathbf{x}(t_0 + 2h) + h \cdot \dot{\mathbf{x}}(t_0 + 2h)$

etc.

La simulación se torna trivial ya que el método de integración utiliza sólo valores pasados de las variables de estado y sus derivadas. Un esquema de integración que exhibe esta característica se denomina *algoritmo de integración explícito*. La mayor parte de los métodos que se utilizan en los lenguajes de simulación de propósito general son de esta naturaleza.

Veamos ahora un algoritmo de integración diferente. La Figura 2.2 muestra un esquema con una pequeña modificación.

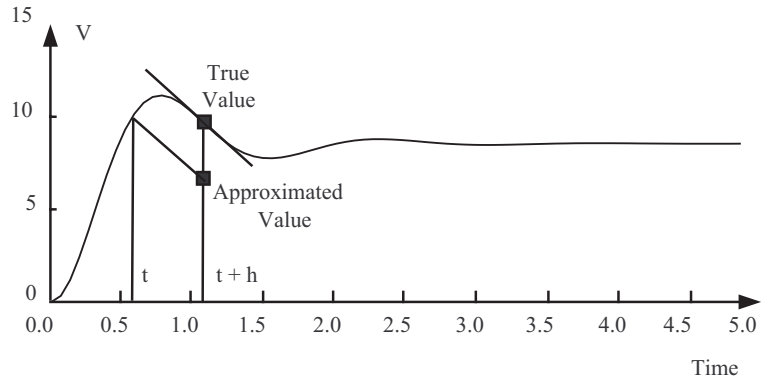

Figura 2.2: Integración numérica utilizando Backward Euler.

En este esquema, la solución $\mathbf{x}(t^* + h)$ se aproxima utilizando los valores de $\mathbf{x}(t^*)$ y $\mathbf{f}(\mathbf{x}(t^* + h), t^* + h)$ mediante la fórmula:

$$\mathbf{x}(t^* + h) \approx \mathbf{x}(t^*) + \mathbf{f}(\mathbf{x}(t^* + h), t^* + h) \cdot h \tag{2.6}$$

Este esquema se conoce como el método de integración de *Backward Euler* (BE).

Como puede verse, esta fórmula depende en el valor actual y en el valor pasado de las variables, lo que causa problemas. Para calcular $\mathbf{x}(t^* + h)$ en la Eq.(2.6), necesitamos conocer $\mathbf{f}(\mathbf{x}(t^* + h), t^* + h))$, pero para calcular $\mathbf{f}(\mathbf{x}(t^* + h), t^* + h))$ de la Ec.(2.1), necesitamos saber $\mathbf{x}(t^* + h)$. En consecuencia, estamos ante un *lazo algebraico* no lineal.

Este tipo de algoritmos se denominan *métodos de integración implícitos*.

Si bien los métodos implícitos son ventajosos desde el punto de vista numérico (veremos esto más adelante), la carga computacional adicional creada por la necesidad de resolver simultáneamente un sistema de ecuaciones algebraicas no lineales al menos una vez por cada paso de integración puede hacerlos inapropiados para su uso en software de simulación de propósito general excepto para aplicaciones específicas, tales como los sistemas stiff.

2.4. El Dominio de Estabilidad Numérica

Volvamos ahora a la solución de un sistema autónomo, lineal y estacionario:

$$\dot{\mathbf{x}} = \mathbf{A} \cdot \mathbf{x} \tag{2.7}$$

con las condiciones iniciales de la Ec.(2.2). La solución analítica es la siguiente:

$$\mathbf{x}(t) = \exp(\mathbf{A} \cdot t) \cdot \mathbf{x_0} \tag{2.8}$$

Esta solución es *analíticamente estable* si todas las trayectorias permanecen acotadas cuando el tiempo tiende a infinito. El sistema (2.7) es analíticamente estable si y sólo si todos los autovalores de $\mathbf{A}$ tienen parte real negativa:

$$\mathbb{R}\mathrm{e}\{\mathrm{Eig}(\mathbf{A})\} = \mathbb{R}\mathrm{e}\{\lambda\} < 0.0 \tag{2.9}$$

El dominio de estabilidad analítica en el plano complejo λ se muestra en la Fig.2.3.

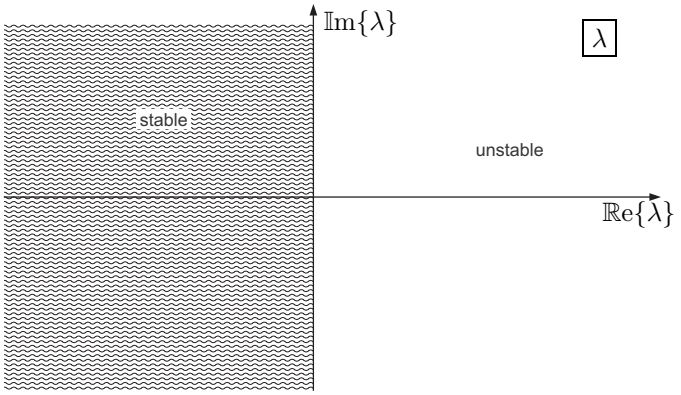

Figura 2.3: Dominio de estabilidad analítica.

Apliquemos entonces el algoritmo de FE a la solución numérica de este problema. Colocando el sistema de la Ec.(2.7) en el algoritmo de la Ec.(2.5), obtenemos:

$$\mathbf{x}(t^* + h) = \mathbf{x}(t^*) + \mathbf{A} \cdot h \cdot \mathbf{x}(t^*) \tag{2.10}$$

que puede reescribirse en forma más compacta como:

$$\mathbf{x}(k+1) = [\mathbf{I^{(n)}} + \mathbf{A} \cdot h] \cdot \mathbf{x}(k) \tag{2.11}$$

$\mathbf{I^{(n)}}$ es una matriz identidad de la misma dimensión que $\mathbf{A}$, es decir, $n \times n$. En lugar de referirnos explícitamente al tiempo de simulación, lo que hacemos es indexar el tiempo, es decir, k se refiere al k–ésimo paso de integración.

Lo que hicimos fue convertir el sistema continuo anterior en un sistema de *tiempo discreto* asociado:

$$\mathbf{x_{k+1}} = \mathbf{F} \cdot \mathbf{x_k} \tag{2.12}$$

donde la matriz de evolución discreta $\mathbf{F}$ puede calcularse a partir de la matriz de evolución continua $\mathbf{A}$ y del paso de integración h, como:

$$\mathbf{F} = \mathbf{I^{(n)}} + \mathbf{A} \cdot h \tag{2.13}$$

El sistema discreto de la Ec.(2.12) es analíticamente estable si y sólo si todos sus autovalores se encuentran dentro de un círculo de radio 1 alrededor del origen, llamado *círculo unitario*. Para que esto ocurra, de la Ec.(2.13) podemos concluir que todos los autovalore de $\mathbf{A}$ multiplicados por el paso de integración h deben caer en un círculo de radio 1.0 alrededor del punto -1.0.

Diremos que un sistema lineal y estacionario de tiempo continuo integrado con un dado método de integración de paso fijo es *numéricamente estable* si y sólo si el sistema de tiempo discreto asociado es analíticamente estable.

La Figura 2.4 muestra el dominio de estabilidad numérica del método de FE.

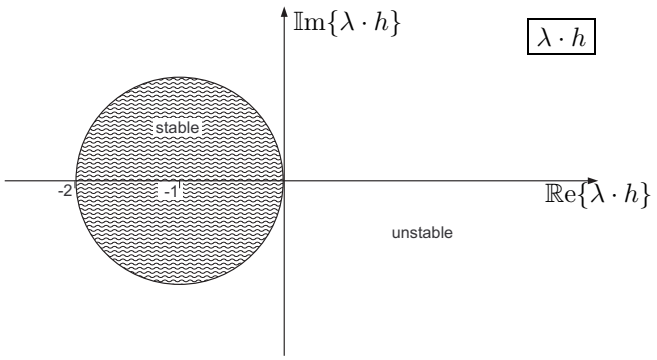

Figura 2.4: Dominio de estabilidad numérica de Forward Euler.

Es importante notar que el dominio de estabilidad numérica, en el sentido riguroso, queda sólo definido para sistemas lineales y estacionarios y puede aplicarse sólamente a algoritmos de paso fijo.

En el caso de FE, el dominio de estabilidad numérica nos muestra que cuando el paso de integración es grande, un sistema analíticamente estable puede dar un resultado numéricamente inestable.

La limitación en el valor de h es particularmente importante cuando los autovalores se encuentran cerca del eje imaginario. En particular, cuando están sobre dicho eje no existe ningún paso de integración que permita obtener una solución púramente oscilante, sino que siempre se obtiene un resultado inestable. Por este motivo, el método de FE no sirve para integrar sistemas con poco amortiguamiento.

Veamos ahora el método de BE. Colocando el modelo de la Ec.(2.7) en el algoritmo de la Ec.(2.5), se obtiene:

$$\mathbf{x}(t^* + h) = \mathbf{x}(t^*) + \mathbf{A} \cdot h \cdot \mathbf{x}(t^* + h) \tag{2.14}$$

que puede reescribirse como:

$$[\mathbf{I^{(n)}} - \mathbf{A} \cdot h] \cdot \mathbf{x}(t^* + h) = \mathbf{x}(t^*) \tag{2.15}$$

o:

$$\mathbf{x}(k+1) = [\mathbf{I^{(n)}} - \mathbf{A} \cdot h]^{-1} \cdot \mathbf{x}(k) \tag{2.16}$$

Luego:

$$\mathbf{F} = [\mathbf{I^{(n)}} - \mathbf{A} \cdot h]^{-1} \tag{2.17}$$

La Figura 2.5 muestra el dominio de estabilidad de este método.

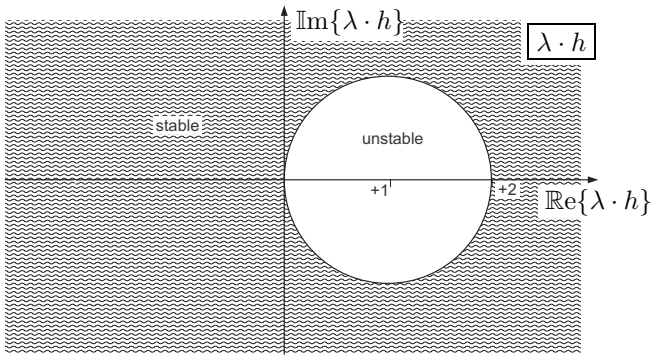

Figura 2.5: Dominio de estabilidad de Backward Euler.

El algoritmo de BE tiene la ventaja de que si el sistema es analíticamente estable, se garantizará la estabilidad numérica para cualquier paso de integración h. Este método es entonces mucho más apropiado que el de FE para resolver problemas con autovalores alejados sobre el eje real negativo del plano complejo. Esto es de crucial importancia en los sistemas *stiff*, es decir, sistemas con autovalores cuyas partes reales están desparramadas a lo largo del eje real negativo.

A diferencia de FE, en BE el paso de integración deberá elegirse exclusivamente en función de los *requisitos de precisión*, sin importar el *dominio de estabilidad numérica*.

Sin embargo, el dominio de estabilidad numérica de BE muestra una región estable en el semiplano derecho. Esto es particularmente peligroso ya que un sistema analíticamente inestable puede resultar numéricamente estable. Por lo tanto, al analizar resultados de simulación puede llegarse a la conclusión (errónea) de que el sistema es estable.

Como en el caso de FE, el método de BE no funciona con sistemas marginalmente estables (sistemas cuyos autovalores dominantes están sobre o cerca del

eje imaginario) ya que ningún paso de integración mostrará el comportamiento oscilatorio no amortiguado.

La obtención del dominio de estabilidad de BE no es tan directa como la de FE. Si bien existen métodos analíticos de determinar el dominio de estabilidad de un algoritmo de integración dado, estos suelen ser bastante complicados. Por esto, veremos una manera alternativa de determinar el dominio de estabilidad de cualquier algoritmo de integración mediante un programa de aproximación simple.

Para esto, comenzaremos armando una matriz $\mathbf{A}$ con autovalores que satisfacen $|\lambda| = 1.0$, y ángulos $\pm\alpha$. Luego, variaremos α entre 0 y pi. Para cada valor de α determinaremos el valor de h para el cual el método de integración conlleva un resultado marginalmente estable (los autovalores de $\mathbf{F}$ tienen módulo igual a 1.0).

De esta forma, el dominio de estabilidad del método se podrá dibujar graficando los valores de h obtenidos con el ángulo α.

La matriz de evolución mencionada tiene la forma

$$\mathbf{A} = \begin{pmatrix} 0 & 1 \\ -1 & 2\cos(\alpha) \end{pmatrix} \tag{2.18}$$

2.5. La Iteración de Newton

En los sistemas lineales, podemos utilizar el método de BE aplicando inversión matricial y llegando a una fórmula como la Ec.(2.16).

Sin embargo, esto no puede hacer en el caso no lineal. De alguna manera hay que resolver el conjunto implícito de ecuaciones algebraicas no lineales que se forman entre el modelo de ecuaciones de estado y el algoritmo de integración implícito. Para esto, necesitaremos algún procedimiento iterativo.

La primer idea en este sentido puede ser la de utilizar un método *predictor–corrector*: empezamos con un paso utilizando FE, y luego utilizamos el resultado de este paso (predictor) para calcular la derivada del estado del paso implícito de BE. Luego, repetimos iterativamente los pasos con BE (corrector).

predictor:
$$\begin{aligned} \dot{\mathbf{x}}_{\mathbf{k}} &= \mathbf{f}(\mathbf{x}_{\mathbf{k}}, t_k) \\ \mathbf{x}^{\mathbf{P}}_{\mathbf{k+1}} &= \mathbf{x}_{\mathbf{k}} + h \cdot \dot{\mathbf{x}}_{\mathbf{k}} \end{aligned}$$

1st corrector:
$$\begin{aligned} \dot{\mathbf{x}}^{\mathbf{P}}_{\mathbf{k+1}} &= \mathbf{f}(\mathbf{x}^{\mathbf{P}}_{\mathbf{k+1}}, t_{k+1}) \\ \mathbf{x}^{\mathbf{C1}}_{\mathbf{k+1}} &= \mathbf{x}_{\mathbf{k}} + h \cdot \dot{\mathbf{x}}^{\mathbf{P}}_{\mathbf{k+1}} \end{aligned}$$

2nd corrector:
$$\begin{aligned} \dot{\mathbf{x}}^{\mathbf{C1}}_{\mathbf{k+1}} &= \mathbf{f}(\mathbf{x}^{\mathbf{C1}}_{\mathbf{k+1}}, t_{k+1}) \\ \mathbf{x}^{\mathbf{C2}}_{\mathbf{k+1}} &= \mathbf{x}_{\mathbf{k}} + h \cdot \dot{\mathbf{x}}^{\mathbf{C1}}_{\mathbf{k+1}} \end{aligned}$$

3rd corrector:
$$\begin{aligned} \dot{\mathbf{x}}^{\mathbf{C2}}_{\mathbf{k+1}} &= \mathbf{f}(\mathbf{x}^{\mathbf{C2}}_{\mathbf{k+1}}, t_{k+1}) \\ \mathbf{x}^{\mathbf{C3}}_{\mathbf{k+1}} &= \mathbf{x}_{\mathbf{k}} + h \cdot \dot{\mathbf{x}}^{\mathbf{C2}}_{\mathbf{k+1}} \end{aligned}$$

etc.

La iteración termina cuando dos aproximaciones sucesivas de $\mathbf{x_{k+1}}$ difieren en menos que una tolerancia preestablecida. Este esquema de iteración se denomina *iteración de punto fijo*.

Lamentablemente, esta idea no funcionará. El problema es que la iteración sólo convergerá cuando los autovalores de $\mathbf{A} \cdot h$ se encuentren dentro del círculo unitario. En consecuencia, el dominio de estabilidad del método resultante es el de la Figura 2.6, es decir, sólo coincide con el de BE cuando los autovalores de $\mathbf{A} \cdot h$ son pequeños.

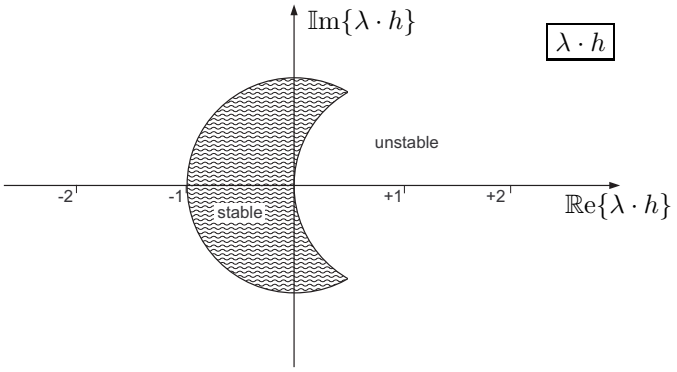

Figura 2.6: Dominio de estabilidad del predictor–corrector FE–BE.

Otro método para encontrar soluciones de ecuaciones algebraicas es la iteración de Newton, cuyo uso para encontrar donde una función se hace cero se ilustra en la Figura 2.7.

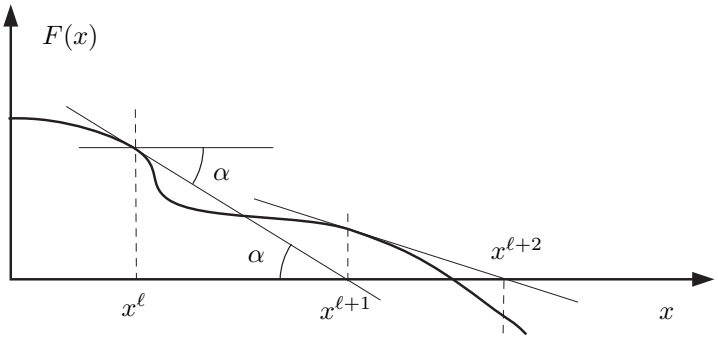

Figura 2.7: Iteración de Newton.

Dada una función arbitraria $\mathcal{F}(x)$, asumiendo que conocemos el valor de tal función y su derivada $\partial\mathcal{F}/\partial x$ en un punto x^{ℓ}, notemos que:

$$\tan\ \alpha = \frac{\partial \mathcal{F}^{\ell}}{\partial x} = \frac{\mathcal{F}^{\ell}}{x^{\ell} - x^{\ell+1}} \tag{2.19}$$

Entonces:

$$x^{\ell+1} = x^{\ell} - \frac{\mathcal{F}^{\ell}}{\partial \mathcal{F}^{\ell}/\partial x} \tag{2.20}$$

Veamos entonces como utilizar esta técnica para iterar en el método de BE aplicado a un sistema no lineal escalar, donde, en un punto x_k tenemos

$$\dot{x}_{k+1} = f(x_{k+1}, t_{k+1}) \tag{2.21}$$

por lo que al aplicar el algoritmo de BE

$$x_{k+1} = x_k + h \cdot \dot{x}_{k+1} \tag{2.22}$$

queda:

$$x_{k+1} = x_k + h \cdot f(x_{k+1}, t_{k+1}) \tag{2.23}$$

o

$$x_k + h \cdot f(x_{k+1}, t_{k+1}) - x_{k+1} = 0.0 \tag{2.24}$$

La Ecuación (2.24) está en la forma adecuada para aplicar la iteración de Newton. Aquí, la variable desconocida es x_{k+1}. Luego,

$$x_{k+1}^{\ell+1} = x_{k+1}^{\ell} - \frac{x_k + h \cdot f(x_{k+1}^{\ell}, t_{k+1}) - x_{k+1}^{\ell}}{h \cdot \partial f(x_{k+1}^{\ell}, t_{k+1})/\partial x - 1.0} \tag{2.25}$$

donde k es el número del paso de integración y ℓ es el número de veces que la iteración de Newton fue aplicada en dicho paso.

La fórmula para el caso matricial de la iteración de Newton tiene la siguiente forma:

$$\mathbf{x}^{\boldsymbol{\ell+1}} = \mathbf{x}^{\ell} - \left(\mathcal{H}^{\ell}\right)^{-1} \cdot \mathcal{F}^{\ell} \tag{2.26}$$

donde:

$$\mathcal{H} = \frac{\partial \mathcal{F}}{\partial \mathbf{x}} = \begin{pmatrix} \partial \mathcal{F}_1/\partial x_1 & \partial \mathcal{F}_1/\partial x_2 & \dots & \partial \mathcal{F}_1/\partial x_n \\ \partial \mathcal{F}_2/\partial x_1 & \partial \mathcal{F}_2/\partial x_2 & \dots & \partial \mathcal{F}_2/\partial x_n \\ \vdots & \vdots & \ddots & \vdots \\ \partial \mathcal{F}_n/\partial x_1 & \partial \mathcal{F}_n/\partial x_2 & \dots & \partial \mathcal{F}_n/\partial x_n \end{pmatrix} \tag{2.27}$$

es la *matriz Hessiana* del problema.

Utilizando este esquema de iteración en el modelo de espacio de estados con el método de BE obtenemos:

$$\mathbf{x}_{\mathbf{k+1}}^{\boldsymbol{\ell+1}} = \mathbf{x}_{\mathbf{k+1}}^{\ell} - [h \cdot \mathcal{J}_{\mathbf{k+1}}^{\ell} - \mathbf{I}^{(\mathbf{n})}]^{-1} \cdot [\mathbf{x_k} + h \cdot \mathbf{f}(\mathbf{x}_{\mathbf{k+1}}^{\ell}, t_{k+1}) - \mathbf{x}_{\mathbf{k+1}}^{\ell}] \tag{2.28}$$

donde:

$$\mathcal{J} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{pmatrix} \partial f_1/\partial x_1 & \partial f_1/\partial x_2 & \dots & \partial f_1/\partial x_n \\ \partial f_2/\partial x_1 & \partial f_2/\partial x_2 & \dots & \partial f_2/\partial x_n \\ \vdots & \vdots & \ddots & \vdots \\ \partial f_n/\partial x_1 & \partial f_n/\partial x_2 & \dots & \partial f_n/\partial x_n \end{pmatrix} \tag{2.29}$$

es la *matriz Jacobiana* del sistema dinámico.

Cualquier implementación de este esquema de iteración requiere, en general, el cómputo de al menos una aproximación de la matriz Jacobiana, y la inversión (refactorización) de la matriz Hessiana. Como ambas operaciones son bastante costosas, las diferentes implementaciones varían en que tan a menudo recalcular el Jacobiano (esto se denomina iteración de Newton modificada). Cuanto más no lineal sea el problema, más a menudo hay que recalcular el Jacobiano. Asímismo, al cambiar el paso de integración, hay que refactorizar el Hessiano.

Veamos entonces como este esquema de iteración afecta la solución y la estabilidad de un problema lineal.

El Jacobiano de un modelo lineal es simplemente la matriz de evolución:

$$\mathcal{J} = \mathbf{A} \tag{2.30}$$

Por esto, el Jacobiano nunca necesita recalcularse. Colocando el sistema lineal en la Ec. (2.28), encontramos:

$$\mathbf{x}_{\mathbf{k+1}}^{\boldsymbol{\ell+1}} = \mathbf{x}_{\mathbf{k+1}}^{\boldsymbol{\ell}} - [\mathbf{A} \cdot h - \mathbf{I}^{(\mathbf{n})}]^{-1} \cdot [(\mathbf{A} \cdot h - \mathbf{I}^{(\mathbf{n})}) \cdot \mathbf{x}_{\mathbf{k+1}}^{\boldsymbol{\ell}} + \mathbf{x_k}] \tag{2.31}$$

o directamente

$$\mathbf{x}_{\mathbf{k+1}}^{\boldsymbol{\ell+1}} = [\mathbf{I}^{(\mathbf{n})} - \mathbf{A} \cdot h]^{-1} \cdot \mathbf{x_k} \tag{2.32}$$

Evidentemente, la iteración de Newton no modifica las propiedades de estabilidad del algoritmo de BE aplicado a un sistema lineal. Esto vale en general para todos los métodos de integración, no sólo para el de BE.

2.6. Problemas Propuestos

[P2.1] Estabilidad Marginal Dado el siguiente sistema lineal y estacionario

$$\begin{aligned} \dot{\mathbf{x}} &= \begin{pmatrix} 1250 & -25113 & -60050 & -42647 & -23999 \\ 500 & -10068 & -24057 & -17092 & -9613 \\ 250 & -5060 & -12079 & -8586 & -4826 \\ -750 & 15101 & 36086 & 25637 & 14420 \\ 250 & -4963 & -11896 & -8438 & -4756 \end{pmatrix} \cdot \mathbf{x} + \begin{pmatrix} 5 \\ 2 \\ 1 \\ -3 \\ 1 \end{pmatrix} \cdot u \\ \mathbf{y} &= \begin{pmatrix} -1 & 26 & 59 & 43 & 23 \end{pmatrix} \cdot \mathbf{x} \end{aligned} \tag{P2.1a}$$

con condiciones iniciales:

$$\mathbf{x_0} = \begin{pmatrix} 1 & -2 & 3 & -4 & 5 \end{pmatrix}^T \tag{P2.1b}$$

Determinar el paso de integración, h_{marg}, para el cual FE dará resultados marginalmente estables.

Simular el sistema durante 10 segundos de tiempo de simulación con una entrada escalón utilizando el método de FE con los siguientes pasos de integración: (i) $h = 0.1 \cdot h_{\text{marg}}$, (ii) $h = 0.95 \cdot h_{\text{marg}}$, (iii) $h = h_{\text{marg}}$, (iv) $h = 1.05 \cdot h_{\text{marg}}$, and (v) $h = 2 \cdot h_{\text{marg}}$. Discutir los resultados.

[P2.2] Precisión de Integración Para el sistema del Prob.[P2.1], determinar el máximo paso de integración que permite obtener una precisión global del 1 %.

Para esto, primero hay que obtener la solución analítica del sistema y luego simularlo utilizando el método de FE (durante 10 segundos), hasta obtener que ambas soluciones difieran en un 1 %:

$$\varepsilon_{\text{global}} = \frac{\|\mathbf{x_{anal}} - \mathbf{x_{num}}\|}{\|\mathbf{x_{anal}}\|} \leq 0.01 \tag{P2.2a}$$

Repetir el mismo experimento con el método de BE. Dado que el sistema es lineal, se permite calcular la matriz $\mathbf{F}$ utilizando inversión matricial.

[P2.3] Method Blending Dado el siguiente sistema lineal y estacionario:

$$\begin{aligned} \dot{\mathbf{x}} &= \begin{pmatrix} 0 & 1 \\ -9.01 & 0.2 \end{pmatrix} \cdot \mathbf{x} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} \cdot u \\ \mathbf{y} &= \begin{pmatrix} 1 & 1 \end{pmatrix} \cdot \mathbf{x} + 2 \cdot u \end{aligned} \tag{P2.3a}$$

with initial conditions:

$$\mathbf{x_0} = \begin{pmatrix} 1 & -2 \end{pmatrix}^T \tag{P2.3b}$$

Encontrar la solución analítica. Simulando durante 25 segundos, determinar el máximo paso de integración que permite alcanzar una precisión global del 1 % utilizando FE. Luego, repetir para BE. Sacar conclusiones.

Un algoritmo mixto puede obtenerse de la siguiente forma: hacemos un paso de integración con FE y luego el mismo paso lo hacemos con BE, y damos como resultado el promedio de ambos valores. Para este método mixto, obtener el máximo paso que permite alcanzar una precisión del 1 %. Comparar el resultado con el de FE y BE por sí mismos.

[P2.4] Método Cíclico Repetir el Prob.[P2.3]. Sin embargo, esta vez utilizaremos otro algoritmo. En lugar de utilizar el promedio de FE y BE, alternaremos un paso con FE y otro con BE. Tales algoritmos se llaman *métodos cíclicos*.

Determinar nuevamente el máximo paso de integración que alcanza un 1 % de precisión y comparar con los métodos de FE y BE.

[P2.5] Dominio de estabilidad: Métodos mixtos y cíclicos Obtener el dominio de estabilidad para el método mixto del Prob.[P2.3]. ¿Que puede

concluirse al comparar dichos dominios de estabilidad con los de FE y BE?. ¿Cómo se puede explicar a partir de estos dominios de estabilidad los resultados del Prob.[P2.3]?

Repetir el análisis para el método cíclico del Prob.[P2.4]. Para analizar el dominio de estabilidad en este último caso, puede utilizar la siguiete ayuda:

En lugar de interpretar este método como la conmutación de dos métodos tras cada paso, puede pensarse que consiste en un simple paso grande formado por dos pasos pequeños:

$$\begin{aligned} \mathbf{x}(k+0.5) &= \mathbf{x}(k) + 0.5 \cdot h \cdot \dot{\mathbf{x}}(k) && \text{(P2.5a)} \\ \mathbf{x}(k+1) &= \mathbf{x}(k+0.5) + 0.5 \cdot h \cdot \dot{\mathbf{x}}(k+1) && \text{(P2.5b)} \end{aligned}$$

[P2.6] Ajuste del Dominio de Estabilidad: Método Mixto Veremos otro método derivado del método mixto antes analizado. Aquí, en lugar de utilizar el promedio de FE y BE, utilizaremos un promedio ponderado:

$$\mathbf{x}(k+1) = \vartheta \cdot \mathbf{x_{FE}}(k+1) \; + \; (1-\vartheta) \cdot \mathbf{x_{BE}}(k+1) \qquad \text{(P2.6a)}$$

Este método se denomina *método–ϑ*. Dibujar los dominios de estabilidad del método para

$$\vartheta = \{0, 0.1, 0.2, 0.24, 0.249, 0.25, 0.251, 0.26, 0.3, 0.5, 0.8, 1\} \qquad \text{(P2.6b)}$$

Interpretar los resultados.

[P2.7] Ajuste del Dominio de Estabilidad: Método Cíclico Veremos ahora otro método–ϑ. Esta vez, a partir de un método cíclico. El parámetro que variaremos es la longitud de los dos semi–pasos de la siguiente forma:

$$\begin{aligned} \mathbf{x}(k+\vartheta) &= \mathbf{x}(k) + \vartheta \cdot h \cdot \dot{\mathbf{x}}(k) && \text{(P2.7a)} \\ \mathbf{x}(k+1) &= \mathbf{x}(k+\vartheta) + (1-\vartheta) \cdot h \cdot \dot{\mathbf{x}}(k+1) && \text{(P2.7b)} \end{aligned}$$

Se pide entonces determinar el parámetro ϑ tal que el método tenga una región de estabilidad similar a BE, pero donde el borde de la estabilidad sobre el eje real positivo del plano $(\lambda \cdot h)$ esté localizado en $+10$ en lugar de $+2$.

Dibujar el dominio de estabilidad para este método.

Capítulo 3

Métodos de Integración Monopaso

Este capítulo es una traducción resumida del Capítulo 3 del libro *Continuous System Simulation* [2].

3.1. Introducción

El siguiente algoritmo es del tipo *predictor–corrector*.

$$\begin{array}{lll} \text{predictor:} & \dot{\mathbf{x}}_{\mathbf{k}} & = \mathbf{f}(\mathbf{x}_{\mathbf{k}}, t_k) \\ & \mathbf{x}^{\mathbf{P}}_{\mathbf{k+1}} & = \mathbf{x}_{\mathbf{k}} + h \cdot \dot{\mathbf{x}}_{\mathbf{k}} \\ \\ \text{corrector:} & \dot{\mathbf{x}}^{\mathbf{P}}_{\mathbf{k+1}} & = \mathbf{f}(\mathbf{x}^{\mathbf{P}}_{\mathbf{k+1}}, t_{k+1}) \\ & \mathbf{x}^{\mathbf{C}}_{\mathbf{k+1}} & = \mathbf{x}_{\mathbf{k}} + h \cdot \dot{\mathbf{x}}^{\mathbf{P}}_{\mathbf{k+1}} \end{array}$$

Para analizar el error del mismo podemos reunir las ecuaciones, obteniendo:

$$\mathbf{x_{k+1}} = \mathbf{x_k} + h \cdot \mathbf{f}(\mathbf{x_k} + h \cdot \mathbf{f_k}, t_k + h) \tag{3.1}$$

La idea es comparar con la expansión en series de Taylor de la verdadera solución (consideraremos sólo hasta el término lineal de la expansión).

Recordando que

$$f(x + \Delta x, y + \Delta y) \approx f(x, y) + \frac{\partial f(x, y)}{\partial x} \cdot \Delta x + \frac{\partial f(x, y)}{\partial y} \cdot \Delta y \tag{3.2}$$

resulta:

$$\mathbf{f}(\mathbf{x_k} + h \cdot \mathbf{f_k}, t_k + h) \approx \mathbf{f}(\mathbf{x_k}, t_k) + \frac{\partial \mathbf{f}(\mathbf{x_k}, t_k)}{\partial \mathbf{x}} \cdot (h \cdot \mathbf{f_k}) + \frac{\partial \mathbf{f}(\mathbf{x_k}, t_k)}{\partial t} \cdot h \tag{3.3}$$

donde $\partial \mathbf{f} / \partial \mathbf{x}$ es la matriz *Jacobiana* del sistema. Utilizando la Ec.(3.3) en la Ec.(3.1), queda:

$$\mathbf{x_{k+1}} \approx \mathbf{x_k} + h \cdot \mathbf{f}(\mathbf{x_k}, t_k) + h^2 \cdot (\frac{\partial \mathbf{f}(\mathbf{x_k}, t_k)}{\partial \mathbf{x}} \cdot \mathbf{f_k} + \frac{\partial \mathbf{f}(\mathbf{x_k}, t_k)}{\partial t}) \tag{3.4}$$

Cmparemos esta expresión con la verdadera serie de Taylor de $\mathbf{x_{k+1}}$ truncada tras el término cuadrático:

$$\mathbf{x_{k+1}} \approx \mathbf{x_k} + h \cdot \mathbf{f}(\mathbf{x_k}, t_k) + \frac{h^2}{2} \cdot \dot{\mathbf{f}}(\mathbf{x_k}, t_k) \tag{3.5}$$

donde:

$$\dot{\mathbf{f}}(\mathbf{x_k}, t_k) = \frac{d\mathbf{f}(\mathbf{x_k}, t_k)}{dt} = \frac{\partial \mathbf{f}(\mathbf{x_k}, t_k)}{\partial \mathbf{x}} \cdot \frac{d\mathbf{x_k}}{dt} + \frac{\partial \mathbf{f}(\mathbf{x_k}, t_k)}{\partial t} \tag{3.6}$$

y:

$$\frac{d\mathbf{x_k}}{dt} = \dot{\mathbf{x}}_\mathbf{k} = \mathbf{f_k} \tag{3.7}$$

En base a estas dos últimas ecuaciones podemos reescribir la aproximación del método predictor–corrector presentado según

$$\mathbf{x_{PC}}(k+1) \approx \mathbf{x_k} + h \cdot \mathbf{f}(\mathbf{x_k}, t_k) + h^2 \cdot \dot{\mathbf{f}}(\mathbf{x_k}, t_k) \tag{3.8}$$

que sólo difiere de la expresión (3.5) en un factor de 2 en el término cuadrático. Recordando la expresión de FE, podemos concluir que la siguiente fórmula nos dará una aproximación con una precisión de segundo orden:

$$\mathbf{x}(k+1) = 0.5 \cdot (\mathbf{x_{PC}}(k+1) + \mathbf{x_{FE}}(k+1)) \tag{3.9}$$

o, en otras palabras:

$$\begin{array}{lll} \text{predictor:} & \dot{\mathbf{x}}_\mathbf{k} & = \mathbf{f}(\mathbf{x_k}, t_k) \\ & \mathbf{x^P_{k+1}} & = \mathbf{x_k} + h \cdot \dot{\mathbf{x}}_\mathbf{k} \\ & & \\ \text{corrector:} & \dot{\mathbf{x}}^\mathbf{P}_\mathbf{k+1} & = \mathbf{f}(\mathbf{x^P_{k+1}}, t_{k+1}) \\ & \mathbf{x^C_{k+1}} & = \mathbf{x_k} + 0.5 \cdot h \cdot (\dot{\mathbf{x}}_\mathbf{k} + \dot{\mathbf{x}}^\mathbf{P}_\mathbf{k+1}) \end{array}$$

que es conocido como el*método de Heun*. A veces también se lo denomina *método de Euler modificado*.

Veremos entonces como generalizar esta idea para obtener métodos de orden superior.

3.2. Métodos de Runge–Kutta

El método de Heun utiliza un paso FE como predictor y luego una mezcla de FE y BE como corrector. Veamos como generalizar esta idea:

$$\begin{aligned}
\text{predictor:} \quad & \dot{\mathbf{x}}_\mathbf{k} = \mathbf{f}(\mathbf{x_k}, t_k) \\
& \mathbf{x^P} = \mathbf{x_k} + h \cdot \beta_{11} \cdot \dot{\mathbf{x}}_\mathbf{k} \\
\\
\text{corrector:} \quad & \dot{\mathbf{x}}^\mathbf{P} = \mathbf{f}(\mathbf{x^P}, t_k + \alpha_1 \cdot h) \\
& \mathbf{x^C_{k+1}} = \mathbf{x_k} + h \cdot (\beta_{21} \cdot \dot{\mathbf{x}}_\mathbf{k} + \beta_{22} \cdot \dot{\mathbf{x}}^\mathbf{P})
\end{aligned}$$

Este conjunto de métodos contiene cuatro parámetros. Los parámetros β_{ij} pesan las derivadas del estado que se calculan en cada paso, y el parámetro α_1 especifica el instante de tiempo en el cual se evalúa la primer etapa del método.

Agrupando las ecuaciones parametrizadas y desarrollando en serie de Taylor las funciones que no se evaluan en el tiempo t_k obtenemos:

$$\mathbf{x^C_{k+1}} = \mathbf{x_k} + h \cdot (\beta_{21} + \beta_{22}) \cdot \mathbf{f_k} + \frac{h^2}{2} \cdot [2 \cdot \beta_{11} \cdot \beta_{22} \cdot \frac{\partial \mathbf{f_k}}{\partial \mathbf{x}} \cdot \mathbf{f_k} + 2 \cdot \alpha_1 \cdot \beta_{22} \cdot \frac{\partial \mathbf{f_k}}{\partial t}] \quad (3.10)$$

mientras que la verdadera Serie de Taylor de $\mathbf{x_{k+1}}$ truncada tras el término cuadrático puede escribirse como:

$$\mathbf{x_{k+1}} \approx \mathbf{x_k} + h \cdot \mathbf{f_k} + \frac{h^2}{2} \cdot [\frac{\partial \mathbf{f_k}}{\partial \mathbf{x}} \cdot \mathbf{f_k} + \frac{\partial \mathbf{f_k}}{\partial t}] \quad (3.11)$$

La comparación de las ecuaciones (3.10) y (3.11) resulta en tres ecuaciones sobre cuatro parámetros desconocidos:

$$\beta_{21} + \beta_{22} = 1 \quad (3.12a)$$
$$2 \cdot \alpha_1 \cdot \beta_{22} = 1 \quad (3.12b)$$
$$2 \cdot \beta_{11} \cdot \beta_{22} = 1 \quad (3.12c)$$

Existen entonces infinitos algoritmos de este tipo. El método de Heun puede caracterizarse por:

$$\alpha = \begin{pmatrix} 1 \\ 1 \end{pmatrix}; \quad \beta = \begin{pmatrix} 1 & 0 \\ 0.5 & 0.5 \end{pmatrix} \quad (3.13)$$

α_2 caracteriza el tiempo cuando se evalua el corrector, lo que es obviamente en t_{k+1}, es decir, $\alpha_2 = 1.0$. β es una matriz triangular inferior.

En muchas referencias se representa este método en una forma algo distinta:

0	0	0
1	1	0
x	1/2	1/2

denominada *tabla de Butcher* del método.

En general, una tabla de Butcher de la forma:

a_1	$b_{1,1}$	$\dots$	$b_{1,n}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
a_n	$b_{n,1}$	$\dots$	$b_{n,n}$
x	c_1	$\dots$	c_n

denota el algoritmo

$$\begin{aligned}
&\text{etapa 0:} && \mathbf{k_1} = \mathbf{f}(\mathbf{x_k} + b_{1,1}\cdot h\cdot \mathbf{k_1} + \ldots + b_{1,n}\cdot h\cdot \mathbf{k_n}, t_k + a_1 h)\\
&\vdots && \vdots\\
&\text{etapa } n-1: && \mathbf{k_n} = \mathbf{f}(\mathbf{x_k} + b_{n,1}\cdot h\cdot \mathbf{k_1} + \ldots + b_{n,n}\cdot h\cdot \mathbf{k_n}, t_k + a_n h)\\
&\text{etapa } n: && \mathbf{x_{k+1}} = \mathbf{x_k} + c_1\cdot h\cdot \mathbf{k_1} + \ldots + c_n\cdot h\cdot \mathbf{k_n}
\end{aligned}$$

Otro algoritmo de dos etapas y de segundo orden muy utilizado es la llamada regla del punto medio explícita, caracterizada por:

$$\alpha = \begin{pmatrix} 0.5 \\ 1 \end{pmatrix};\quad \beta = \begin{pmatrix} 0.5 & 0 \\ 0 & 1 \end{pmatrix} \tag{3.14}$$

o equivalentemente por la tabla de Butcher

0	0	0
1/2	1/2	0
x	0	1

Puede implementarse como:

$$\begin{aligned}
&\text{predictor:} && \dot{\mathbf{x}}_\mathbf{k} = \mathbf{f}(\mathbf{x_k}, t_k)\\
& && \mathbf{x}^\mathbf{P}_{\mathbf{k}+\frac{\mathbf{1}}{\mathbf{2}}} = \mathbf{x_k} + \tfrac{h}{2}\cdot \dot{\mathbf{x}}_\mathbf{k}\\
&\text{corrector:} && \dot{\mathbf{x}}^\mathbf{P}_{\mathbf{k}+\frac{\mathbf{1}}{\mathbf{2}}} = \mathbf{f}(\mathbf{x}^\mathbf{P}_{\mathbf{k}+\frac{\mathbf{1}}{\mathbf{2}}}, t_{k+\frac{1}{2}})\\
& && \mathbf{x}^\mathbf{C}_{\mathbf{k+1}} = \mathbf{x_k} + h\cdot \dot{\mathbf{x}}^\mathbf{P}_{\mathbf{k}+\frac{\mathbf{1}}{\mathbf{2}}}
\end{aligned}$$

Esta técnica, que evalua el predictor en tiempo $t_k + h/2$, es algo más económica que el método de Heun ya que tiene un elemento nulo más en la matriz β (también puede verse en la tabla de Butcher).

Toda esta familia de métodos, que satisfacen las Ecs (3.12a–c), se denominan métodos de segundo orden de Runge Kutta (RK2).

La idea puede generalizarse. La familia entera de métodos explícitos de RK puede describirse como sigue:

$$\begin{aligned}
&\text{etapa 0:} && \dot{\mathbf{x}}^{\mathbf{P_0}} = \mathbf{f}(\mathbf{x_k}, t_k)\\
&\text{etapa } j: && \mathbf{x}^{\mathbf{P_j}} = \mathbf{x_k} + h\cdot \textstyle\sum_{i=1}^{j} \beta_{ji}\cdot \dot{\mathbf{x}}^{\mathbf{P_{i-1}}}\\
& && \dot{\mathbf{x}}^{\mathbf{P_j}} = \mathbf{f}(\mathbf{x}^{\mathbf{P_j}}, t_k + \alpha_j\cdot h)\\
&\text{última etapa:} && \mathbf{x_{k+1}} = \mathbf{x_k} + h\cdot \textstyle\sum_{i=1}^{\ell} \beta_{\ell i}\cdot \dot{\mathbf{x}}^{\mathbf{P_{i-1}}}
\end{aligned}$$

donde ℓ denota el número de etapas del método. El método más popular de estos es el de Runge–Kutta de orden cuatro (RK4) con tabla de Butcher:

0	0	0	0	0
1/2	1/2	0	0	0
1/2	0	1/2	0	0
1	0	0	1	0
x	1/6	1/3	1/3	1/6

o sea,

$$\text{etapa 0:} \quad \mathbf{k_1} = \mathbf{f}(\mathbf{x_k}, t_k)$$

$$\text{etapa 1:} \quad \mathbf{k_2} = \mathbf{f}(\mathbf{x_k} + \tfrac{h}{2} \cdot \mathbf{k_1}, t_k + \tfrac{h}{2})$$

$$\text{etapa 2:} \quad \mathbf{k_3} = \mathbf{f}(\mathbf{x_k} + \tfrac{h}{2} \cdot \mathbf{k_2}, t_k + \tfrac{h}{2})$$

$$\text{etapa 3:} \quad \mathbf{k_4} = \mathbf{f}(\mathbf{x_k} + h \cdot \mathbf{k_3}, t_k + h)$$

$$\text{etapa 4:} \quad \mathbf{x_{k+1}} = \mathbf{x_k} + \tfrac{h}{6} \cdot [\mathbf{k_1} + 2 \cdot \mathbf{k_2} + 2 \cdot \mathbf{k_3} + \mathbf{k_4}]$$

Este algoritmo es particularmente atractivo debido a que tiene muchos elementos nulos en la tabla de Butcher. Tiene, como puede verse, cuatro evaluaciones de la función $\mathbf{f}$.

Es importante notar que el número de etapas y el orden de la aproximación no son necesariamente iguales. Los algoritmos de RK de orden superior requieren un mayor número de etapas que lo que indica el orden. La Tabla 3.1 brinda un pantallazo histórico sobre el desarrollo de los métodos de RK.

Autor	Año	Orden	# de Etapas
Euler	1768	1	1
Runge	1895	4	4
Heun	1900	2	2
Kutta	1901	5	6
Huťa	1956	6	8
Shanks	1966	7	9
Curtis	1970	8	11

Cuadro 3.1: Historia de los Algoritmos de Runge–Kutta.

3.3. Dominio de Estabilidad de los Algoritmos RK

Como todos los métodos de RK vistos hasta aquí son explícitos, es de esperar que sus dominios de estabilidad sean similares al del algoritmo de FE, o sea, que el contorno de estabilidad marginal se cierre sobre el semiplano izquierdo del plano $(\lambda \cdot h)$.

Reemplazando el sistema lineal de la Ec.(2.7) en el método de Heun encontramos:

$$\begin{array}{lll} \text{predictor:} & \dot{\mathbf{x}}_{\mathbf{k}} = \mathbf{A} \cdot \mathbf{x}_{\mathbf{k}} \\ & \mathbf{x}^{\mathbf{P}}_{\mathbf{k+1}} = \mathbf{x}_{\mathbf{k}} + h \cdot \dot{\mathbf{x}}_{\mathbf{k}} \\ \\ \text{corrector:} & \dot{\mathbf{x}}^{\mathbf{P}}_{\mathbf{k+1}} = \mathbf{A} \cdot \mathbf{x}^{\mathbf{P}}_{\mathbf{k+1}} \\ & \mathbf{x}^{\mathbf{C}}_{\mathbf{k+1}} = \mathbf{x}_{\mathbf{k}} + 0.5 \cdot h \cdot (\dot{\mathbf{x}}_{\mathbf{k}} + \dot{\mathbf{x}}^{\mathbf{P}}_{\mathbf{k+1}}) \end{array}$$

o:

$$\mathbf{x}^{\mathbf{C}}_{\mathbf{k+1}} = [\mathbf{I}^{(\mathbf{n})} + \mathbf{A} \cdot h + \frac{(\mathbf{A} \cdot h)^2}{2}] \cdot \mathbf{x}_{\mathbf{k}} \tag{3.15}$$

es decir,

$$\mathbf{F} = \mathbf{I}^{(\mathbf{n})} + \mathbf{A} \cdot h + \frac{(\mathbf{A} \cdot h)^2}{2} \tag{3.16}$$

Dado que un algoritmo en dos etapas contiene sólo dos evaluaciones de la función, no pueden aparecer potencias de h mayores que 2 en la matriz $\mathbf{F}$. Dado que el método es aproximado hasta un segundo orden, debe aproximar la solución analítica:

$$\mathbf{F} = \exp(\mathbf{A} \cdot h) = \mathbf{I}^{(\mathbf{n})} + \mathbf{A} \cdot h + \frac{(\mathbf{A} \cdot h)^2}{2!} + \frac{(\mathbf{A} \cdot h)^3}{3!} + \ldots \tag{3.17}$$

hasta el término cuadrático. En consecuencia, todos los métodos RK2 de dos etapas tienen el mismo dominio de estabilidad, y lo mismo vale para todos los RK3 de tres etapas y los RK4 de cuatro etapas. Esta situación es un poco más complicada en el caso de los algoritmos de quinto orden ya que no existe un método RK5 de cinco etapas. En consecuencia, las matrices $\mathbf{F}$ de los RK5 contienen necesariamente un término de h^6 (con coeficiente incorrecto), y en principio los dominios de estabilidad serán algo distintos entre sí (según cual sea este coeficiente incorrecto).

Los dominios de estabilidad de los métodos RK1 (Euler) a RK4 se muestran en la Fig.3.1.

Puede notarse como al incrementarse el orden, los métodos aproximan mejor el dominio de estabilidad analítica. Esto es muy bueno ya que los métodos de orden superior permiten utilizar pasos más grandes.

3.4. Sistemas Stiff

Un sistema lineal y estacionario se dice que es *stiff* cuando es estable y hay autovaloras cuyas partes reales son muy distintas, es decir, hay modos muy rápidos y modos muy lentos.

El problema con los sistemas stiff es que la presencia de los modos rápidos obliga a utilizar un paso de integración muy pequeño para no caer en la zona de inestabilidad del método.

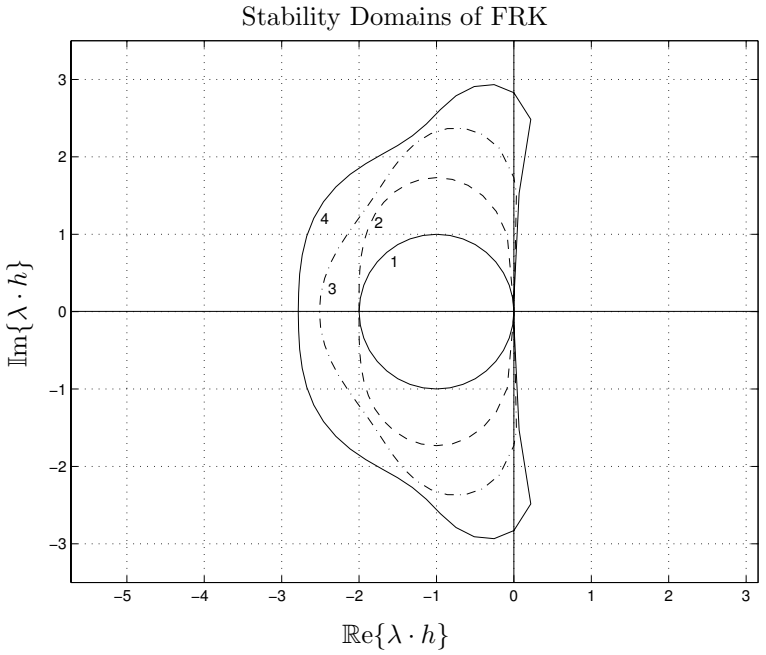

Figura 3.1: Dominios de estabilidad de los métodos explicitos RK.

El concepto también existe en el caso no lineal, pero aquí hace falta una definición un poco distinta:

> *Definición:* "Un sistema de Ecuaciones Diferenciales Ordinarias se dice stiff si, al integrarlo con un método de orden n y tolerancia de error local de 10^{-n}, el paso de integración del algoritmo debe hacerse más pequeño que el valor indicado por la estima del error local debido a las restricciones impuestas por la región de estabilidad numérica."

Para integrar entonces sistemas stiff sin tener que reducir el paso de integración a causa de la estabilidad, es necesario buscar métodos que incluyan en su región estable el semiplano izquierdo completo del plano $(\lambda \cdot h)$, o al menos una gran porción del mismo.

> *Definición:* Un método de integración que contiene en su región de estabilidad a todo el semiplano izquierdo del plano $(\lambda \cdot h)$ se denomina *absolutamente estable*, o, más simplemente, *A–estable*.

Una forma de obtener métodos A–estables es modificar la receta para desarrollar algoritmos RK permitiendo elementos no nulos sobre y por encima de la diagonal de la matriz β (equivalentemente, sobre y por encima de la diagonal de la tabla

de Butcher). Tales algoritmos son invariantemente implícitos y se denominan *Métodos implícitos de Runge–Kutta*, y se abrevian IRK.

Dentro de estos, juegan un rol especial los que sólo permiten elementos no nulos sobre la diagonal de la tabla de Butcher. Estos algoritmos se denominan *Métodos de Runge–Kutta diagonalmente implícitos* o DIRK. La particularidad de los métodos DIRK es que sólo son implícitos en cada etapa de forma separada y es relativamente fácil implementar la iteración de Newton.

Veremos luego algunas formas de obtener distintos métodos implícitos de Runge–Kutta.

3.5. Sistemas Marginalmente Estables

Vimos en el Capítulo 2 que ni el método de FE ni el de BE funcionan bien cuando los autovalores están sobre o muy cerca del eje imaginario. Desafortunadamente, esta situación ocurre muy frecuentemente. Mas aún, hay una clase completa de aplicaciones, correspondiente a las Ecuaciones en Derivadas Parciales hiperbólicas que, convertidas en conjunto de Ecuaciones Diferenciales Ordinarias, exhiben esta característica. Estudiaremos entonces el problema.

> *Definición:* Un sistema dinámico cuyo Jacobiano tiene sus autovalores dominantes sobre o muy cerca del eje imaginario se denomina *Marginalmente Estable*.

Los autovalores dominantes de una matriz son los que tienen la mayor parte real (es decir, que están más a la derecha en el plano λ.

Para atacar estos problemas, serán necesarios algoritmos de integración que aproximen bien sobre el eje imaginario. Veremos más adelante que estos algoritmos existen y que pueden construirse algoritmos de ordenarbitrario cuyo borde de estabilidad numérica coincida con el eje imaginario.

> *Definició:* Un método de integración numérica cuyo dominio de estabilidad numérica consiste exclusivamente del semiplano izquierdo completo del plano $(\lambda \cdot h)$ se denomina *fielmente estable*, o simplemente, *F–estable*.

Puede pensarse que los métodos F–estables deben ser los más apropiados para todos los problemas. Desafortunadamente, los algoritmos F–estables no funcionan bien con los sistemas stiff.

Cuando un autovalor tiene su parte real muy negativa se dice que tiene un gran amortiguamiento. En consecuencia, el modo correspondiente tiende a cero en un tiempo muy pequeño. Particularmente, cuando el autovalor tiende a $-\infty$, el amortiguamiento tiende a infinito y en tiempo tendiendo a cero el modo se extingue.

Sin embargo, al utilizar un método F–estable a un sistema con infinito amortiguamiento, resulta que el amortiguamiento de la solución numérica no es infinito (luego veremos esto en más detalle).

Por esto, es importante tener en cuenta la siguiente definición:

> *Definición:* Un método numérico de integración que es A–estable y, además, sus propiedades de amortiguamiento tienden a infinito cuando $\mathbb{R}\mathrm{e}\{\lambda\} \to -\infty$, se denomina *L–estable.*

Estas definiciones tienen sentido, en el sentido estricto, sólo en los sistemas lineales y estacionarios. Sin embargo normalmente son también buenos indicadores aplicados a los sistemas no lineales.

Al tratar con sistemas stiff, no es suficiente utilizar algortimos A–estables, sino también deben ser L–estables. Evidentemente, todos los algoritmos F–estables son A–estables pero nunca L–estables.

Un sistema que es a la vez marginalmente estable y stiff es, en consecuencia, dificil de tratar. Más adelante veremos esto con más detalle.

3.6. Métodos de Interpolación hacia Atrás

Para poder tratar los problemas stiff y marginalmente estables, necesitamos algoritmos A–estables. Veremos entonces una clase especial de algoritmos IRK que se denominan *métodos de backinterpolation*, o métodos de interpolación hacia atrás (métodos BI). Veremos que los métodos BI pueden hacerse F–estables, L–estables o algo en el medio de ambas características de acuerdo a la necesidad del usuario.

Veamos nuevamente el método de BE:

$$\mathbf{x_{k+1}} = \mathbf{x_k} + h \cdot \dot{\mathbf{x}}_{\mathbf{k+1}} \tag{3.18}$$

Podemos reescribir la Ec.(3.18) como sigue:

$$\mathbf{x_k} = \mathbf{x_{k+1}} - h \cdot \dot{\mathbf{x}}_{\mathbf{k+1}} \tag{3.19}$$

Luego, un paso hacia adelante utilizando BE puede interpretarse como un paso hacia atrás de valor $-h$ utilizando FE.

Una manera de implementar BE entonces es comenzar con una estimación de $\mathbf{x_{k+1}}$, integrar hacia atrás en el tiempo y luego iterar sobre la condición "inicial" desconocida $\mathbf{x_{k+1}}$ hasta acertar el valor "final" conocido $\mathbf{x_k}$.

Esta idea puede tambier extenderse a cualquier algoritmo RK. Los métodos resultantes se denominan *Backward Runge–Kutta* o métodos de Runge–Kutta hacia atrás, y se abrevian BRK.

Por ejemplo, podemos obtener un método BRK2 integrando hacia atrás con el método de Heun:

$$\begin{aligned}
\text{etapa 0:} \quad & \tilde{\mathbf{k}}_{\mathbf{1}} = \mathbf{f}(\mathbf{x_{k+1}}, t_{k+1}) \\
\text{etapa 1:} \quad & \tilde{\mathbf{k}}_{\mathbf{2}} = \mathbf{f}(\mathbf{x_{k+1}} - h \cdot \tilde{\mathbf{k}}_{\mathbf{1}}, t_{k+1} - h) \\
\text{etapa 2:} \quad & \mathbf{x_k} = \mathbf{x_{k+1}} - \tfrac{h}{2} \cdot [\tilde{\mathbf{k}}_{\mathbf{1}} + \tilde{\mathbf{k}}_{\mathbf{2}}]
\end{aligned}$$

Definiendo $\mathbf{k_1} \triangleq \tilde{\mathbf{k}}_{\mathbf{2}}$ y $\mathbf{k_2} \triangleq \tilde{\mathbf{k}}_{\mathbf{1}}$, y usando que $t_{k+1} = t_k + h$, podemos reescribir:

$$\text{etapa 0:} \quad \mathbf{k_1} = \mathbf{f}(\mathbf{x_k} + \tfrac{h}{2}\mathbf{k_1} - \tfrac{h}{2}\mathbf{k_2}, t_k)$$

$$\text{etapa 1:} \quad \mathbf{k_2} = \mathbf{f}(\mathbf{x_k} + \tfrac{h}{2}\mathbf{k_1} + \tfrac{h}{2}\mathbf{k_2}, t_k + h)$$

$$\text{etapa 2:} \quad \mathbf{x_{k+1}} = \mathbf{x_k} + \tfrac{h}{2} \cdot [\mathbf{k_1} + \mathbf{k_2}]$$

que tiene tabla de Butcher:

0	1/2	-1/2
1	1/2	1/2
x	1/2	1/2

Este método es implícito y requiere iterar sobre dos ecuaciones no lineales simultáneamente.

Sin embargo, en general no implementaremos los métodos BI de esta forma, sino que utilizaremos la idea descripta antes de dar un paso hacia atrás e iterar hasta que la condición "final" del paso hacia atrás coincida con el valor conocido de $\mathbf{x_{k+1}}$.

Para el sistema lineal (2.7), dar un paso hacia atrás con el método de Heun implica:

$$\mathbf{x_k} = [\mathbf{I^{(n)}} + \mathbf{A} \cdot (-h) + \frac{(\mathbf{A} \cdot (-h))^2}{2!}] \cdot \mathbf{x_{k+1}}$$

de donde, la matriz $\mathbf{F}$ del método BRK2 tiene la forma $[\mathbf{I^{(n)}} - \mathbf{A} \cdot h + \frac{(\mathbf{A} \cdot h)^2}{2!}]^{-1}$.

Siguiendo esta idea, podemos obtener una serie de algoritmos de orden creciente con matrices $\mathbf{F}$:

$$\mathbf{F_1} = [\mathbf{I^{(n)}} - \mathbf{A} \cdot h]^{-1} \tag{3.20a}$$

$$\mathbf{F_2} = [\mathbf{I^{(n)}} - \mathbf{A} \cdot h + \frac{(\mathbf{A} \cdot h)^2}{2!}]^{-1} \tag{3.20b}$$

$$\mathbf{F_3} = [\mathbf{I^{(n)}} - \mathbf{A} \cdot h + \frac{(\mathbf{A} \cdot h)^2}{2!} - \frac{(\mathbf{A} \cdot h)^3}{3!}]^{-1} \tag{3.20c}$$

$$\mathbf{F_4} = [\mathbf{I^{(n)}} - \mathbf{A} \cdot h + \frac{(\mathbf{A} \cdot h)^2}{2!} - \frac{(\mathbf{A} \cdot h)^3}{3!} + \frac{(\mathbf{A} \cdot h)^4}{4!}]^{-1} \tag{3.20d}$$

Los correspondientes dominios de estabilidad se muestran en la Fig.(3.2). Evidentemente, los dominios de estabilidad de los métodos BI son imágenes en espejo de los dominios de estabilidad de los métodos explícitos de RK. Esto se puede entender fácilmente ya que se trata de los mismos algoritmos con paso h en vez de $-h$.

Veamos ahora como podemos explotar esta idea de interpolación hacia atrás para generar algoritmos F–estables de orden creciente.

Un algoritmo F–estable muy conocido es la *regla trapezoidal*:

$$1^{\text{st}} \text{ stage:} \quad \mathbf{x_{k+\frac{1}{2}}} = \mathbf{x_k} + \tfrac{h}{2} \cdot \dot{\mathbf{x}}_\mathbf{k}$$

$$2^{\text{nd}} \text{ stage:} \quad \mathbf{x_{k+1}} = \mathbf{x_{k+\frac{1}{2}}} + \tfrac{h}{2} \cdot \dot{\mathbf{x}}_\mathbf{k+1}$$

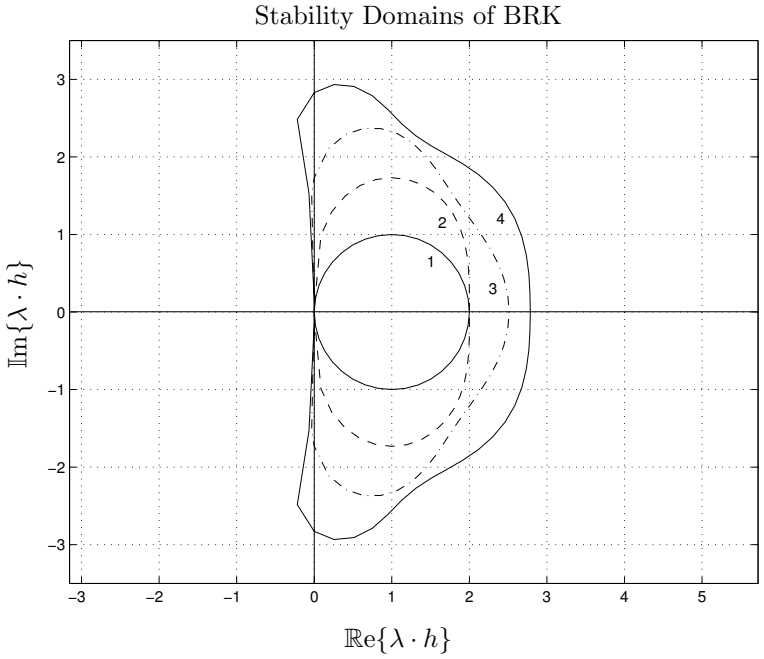

Figura 3.2: Dominios de estabilidad de los métodos básicos de interpolación hacia atrás.

Este es un método implícito que puede interpretarse como un método cíclico consistente en dos semi–pasos de longitud $h/2$ usando FE y luego BE.

Su matriz $\mathbf{F}$ es entonces:

$$\mathbf{F_{TR}} = [\mathbf{I^{(n)}} - \mathbf{A} \cdot \frac{h}{2}]^{-1} \cdot [\mathbf{I^{(n)}} + \mathbf{A} \cdot \frac{h}{2}] \tag{3.21}$$

La regla trapezoidal explota la simetria de los dominios de estabilidad de ambos semi–pasos.

Dado que podemos implementar el semi–paso de BE como un paso BRK1, podemos luego generalizar la idea combinando semipasos de métodos de orden mayor. Las matrices $\mathbf{F}$ resultantes serán:

$$\mathbf{F_1} = [\mathbf{I^{(n)}} - \mathbf{A} \cdot \frac{h}{2}]^{-1} \cdot [\mathbf{I^{(n)}} + \mathbf{A} \cdot \frac{h}{2}] \tag{3.22a}$$

$$\mathbf{F_2} = [\mathbf{I^{(n)}} - \mathbf{A} \cdot \frac{h}{2} + \frac{(\mathbf{A} \cdot h)^2}{8}]^{-1} \cdot [\mathbf{I^{(n)}} + \mathbf{A} \cdot \frac{h}{2} + \frac{(\mathbf{A} \cdot h)^2}{8}] \tag{3.22b}$$

$$\mathbf{F_3} = [\mathbf{I^{(n)}} - \mathbf{A} \cdot \frac{h}{2} + \frac{(\mathbf{A} \cdot h)^2}{8} - \frac{(\mathbf{A} \cdot h)^3}{48}]^{-1} \cdot [\mathbf{I^{(n)}} + \mathbf{A} \cdot \frac{h}{2} + \frac{(\mathbf{A} \cdot h)^2}{8} + \frac{(\mathbf{A} \cdot h)^3}{48}] \tag{3.22c}$$

$$\mathbf{F_4} = [\mathbf{I^{(n)}} - \mathbf{A} \cdot \frac{h}{2} + \frac{(\mathbf{A} \cdot h)^2}{8} - \frac{(\mathbf{A} \cdot h)^3}{48} + \frac{(\mathbf{A} \cdot h)^4}{384}]^{-1} \cdot [\mathbf{I^{(n)}} + \mathbf{A} \cdot \frac{h}{2} + \frac{(\mathbf{A} \cdot h)^2}{8} + \frac{(\mathbf{A} \cdot h)^3}{48} + \frac{(\mathbf{A} \cdot h)^4}{384}] \tag{3.22d}$$

Todos estos métodos son F–estables. El método $\mathbf{F_2}$ no es muy útil ya que $\mathbf{F_1}$ es, por casualidad, de segundo orden.

La implementación de estos algoritmos es trivial. Por ejemplo, $\mathbf{F_4}$ puede implementarse como sigue. Empezamos desde t_k e integramos hacia adelante con medio paso hacia adelante hasta $t_{k+\frac{1}{2}}$ usando algún método RK4. Guardamos entonces el resultado $\mathbf{x}^{\text{left}}_{\mathbf{k+\frac{1}{2}}}$ para reutilizar luego. Luego estimamos el valor de $\mathbf{x_{k+1}}$, por ejemplo, haciendo $\mathbf{x_{k+1}} = \mathbf{x}^{\text{left}}_{\mathbf{k+\frac{1}{2}}}$, e integramos hacia atrás desde t_{k+1} hasta $t_{k+\frac{1}{2}}$ con el mismo algoritmo RK4. Obtenemos entonces $\mathbf{x}^{\text{right}}_{\mathbf{k+\frac{1}{2}}}$. Luego iteramos sobre el estado desconocido $\mathbf{x_{k+1}}$ hasta que $\mathbf{x}^{\text{right}}_{\mathbf{k+\frac{1}{2}}} = \mathbf{x}^{\text{left}}_{\mathbf{k+\frac{1}{2}}}$. Entonces utilizamos el valor final de $\mathbf{x_{k+1}}$ como condición inicial para el siguiente paso.

Es necesario, de todas formas, analizar un poco mejor el proceso de iteración mencionado. Para aplicar la iteración de Newton en el algoritmo BI1, seleccionamos:

$$\mathcal{F}(\mathbf{x_{k+1}}) = \mathbf{x}^{\text{right}}_{\mathbf{k+\frac{1}{2}}} - \mathbf{x}^{\text{left}}_{\mathbf{k+\frac{1}{2}}} = 0.0 \tag{3.23}$$

y resulta:

$$
\begin{array}{ll}
\mathbf{x}^{\text{left}}_{\mathbf{k+\frac{1}{2}}} & = \text{FE}(\mathbf{x_k}, t_k, \frac{h}{2}) \\
\mathbf{x^0_{k+1}} & = \mathbf{x}^{\text{left}}_{\mathbf{k+\frac{1}{2}}} \\
 & \\
\mathbf{J^0_{k+1}} & = \mathcal{J}(\mathbf{x^0_{k+1}}, t_{k+1}) \\
\mathbf{x}^{\text{right\_1}}_{\mathbf{k+\frac{1}{2}}} & = \text{FE}(\mathbf{x^0_{k+1}}, t_{k+1}, -\frac{h}{2}) \\
\mathbf{H^1} & = \mathbf{I^{(n)}} - \frac{h}{2} \cdot \mathbf{J^0_{k+1}} \\
\mathbf{x^1_{k+1}} & = \mathbf{x^0_{k+1}} - \mathbf{H^{1^{-1}}} \cdot (\mathbf{x}^{\text{right\_1}}_{\mathbf{k+\frac{1}{2}}} - \mathbf{x}^{\text{left}}_{\mathbf{k+\frac{1}{2}}}) \\
\varepsilon^{\mathbf{1}}_{\mathbf{k+1}} & = \|\mathbf{x^1_{k+1}} - \mathbf{x^0_{k+1}}\|_\infty \\
 & \\
\mathbf{J^1_{k+1}} & = \mathcal{J}(\mathbf{x^1_{k+1}}, t_{k+1}) \\
\mathbf{x}^{\text{right\_2}}_{\mathbf{k+\frac{1}{2}}} & = \text{FE}(\mathbf{x^1_{k+1}}, t_{k+1}, -\frac{h}{2}) \\
\mathbf{H^2} & = \mathbf{I^{(n)}} - \frac{h}{2} \cdot \mathbf{J^1_{k+1}} \\
\mathbf{x^2_{k+1}} & = \mathbf{x^1_{k+1}} - \mathbf{H^{2^{-1}}} \cdot (\mathbf{x}^{\text{right\_2}}_{\mathbf{k+\frac{1}{2}}} - \mathbf{x}^{\text{left}}_{\mathbf{k+\frac{1}{2}}}) \\
\varepsilon^{\mathbf{2}}_{\mathbf{k+1}} & = \|\mathbf{x^2_{k+1}} - \mathbf{x^1_{k+1}}\|_\infty \\
 & \\
\text{etc.} &
\end{array}
$$

donde $\mathcal{J}$ denota el Jacobiano. Para el método de Heun (BI2), resulta:

$$
\begin{array}{ll}
\mathbf{x}^{\text{left}}_{\mathbf{k+\frac{1}{2}}} & = \text{Heun}(\mathbf{x_k}, t_k, \frac{h}{2}) \\
\mathbf{x^0_{k+1}} & = \mathbf{x}^{\text{left}}_{\mathbf{k+\frac{1}{2}}} \\
 & \\
\mathbf{J^0_{k+1}} & = \mathcal{J}(\mathbf{x^0_{k+1}}, t_{k+1}) \\
\mathbf{x}^{\text{right\_1}}_{\mathbf{k+\frac{1}{2}}} & = \text{Heun}(\mathbf{x^0_{k+1}}, t_{k+1}, -\frac{h}{2}) \\
\mathbf{J^0_{k+\frac{1}{2}}} & = \mathcal{J}(\mathbf{x}^{\text{right\_1}}_{\mathbf{k+\frac{1}{2}}}, t_{k+\frac{1}{2}}) \\
\mathbf{H^1} & = \mathbf{I^{(n)}} - \frac{h}{4} \cdot (\mathbf{J^0_{k+1}} + \mathbf{J^0_{k+\frac{1}{2}}} \cdot (\mathbf{I^{(n)}} - \frac{h}{2} \cdot \mathbf{J^0_{k+1}})) \\
\mathbf{x^1_{k+1}} & = \mathbf{x^0_{k+1}} - \mathbf{H^{1^{-1}}} \cdot (\mathbf{x}^{\text{right\_1}}_{\mathbf{k+\frac{1}{2}}} - \mathbf{x}^{\text{left}}_{\mathbf{k+\frac{1}{2}}}) \\
\varepsilon^{\mathbf{1}}_{\mathbf{k+1}} & = \|\mathbf{x^1_{k+1}} - \mathbf{x^0_{k+1}}\|_\infty \\
 & \\
\mathbf{J^1_{k+1}} & = \mathcal{J}(\mathbf{x^1_{k+1}}, t_{k+1}) \\
\mathbf{x}^{\text{right\_2}}_{\mathbf{k+\frac{1}{2}}} & = \text{Heun}(\mathbf{x^1_{k+1}}, t_{k+1}, -\frac{h}{2}) \\
\mathbf{J^1_{k+\frac{1}{2}}} & = \mathcal{J}(\mathbf{x}^{\text{right\_2}}_{\mathbf{k+\frac{1}{2}}}, t_{k+\frac{1}{2}}) \\
\mathbf{H^2} & = \mathbf{I^{(n)}} - \frac{h}{4} \cdot (\mathbf{J^1_{k+1}} + \mathbf{J^1_{k+\frac{1}{2}}} \cdot (\mathbf{I^{(n)}} - \frac{h}{2} \cdot \mathbf{J^1_{k+1}})) \\
\mathbf{x^2_{k+1}} & = \mathbf{x^1_{k+1}} - \mathbf{H^{2^{-1}}} \cdot (\mathbf{x}^{\text{right\_2}}_{\mathbf{k+\frac{1}{2}}} - \mathbf{x}^{\text{left}}_{\mathbf{k+\frac{1}{2}}}) \\
\varepsilon^{\mathbf{2}}_{\mathbf{k+1}} & = \|\mathbf{x^2_{k+1}} - \mathbf{x^1_{k+1}}\|_\infty \\
 & \\
\text{etc.} &
\end{array}
$$

El algoritmo permanece básicamente igual, excepto que ahora necesitamos evaluar dos Jacobianos en distintos instantes de tiempo, y la fórmula para el Hessiano resulta algo más complicada.

Si asumimos que el Jacobiano permanece casi sin cambios durante cada paso de integración (iteración de Newton modificada), podemos calcular tanto el Jacobiano como el Hessiano al principio del paso, y encontramos para BI1:

$$\mathbf{J} = \mathcal{J}(\mathbf{x_k}, t_k) \tag{3.24a}$$

$$\mathbf{H} = \mathbf{I^{(n)}} - \frac{h}{2} \cdot \mathbf{J} \tag{3.24b}$$

y para BI2:

$$\mathbf{J} = \mathcal{J}(\mathbf{x_k}, t_k) \tag{3.25a}$$

$$\mathbf{H} = \mathbf{I^{(n)}} - \frac{h}{2} \cdot \mathbf{J} + \frac{h^2}{8} \cdot \mathbf{J}^2 \tag{3.25b}$$

La secuencia para las matrices $\mathbf{H}$ de los métodos de orden superior es entonces:

$$\mathbf{H_1} = \mathbf{I^{(n)}} - \mathbf{J} \cdot \frac{h}{2} \tag{3.26a}$$

$$\mathbf{H_2} = \mathbf{I^{(n)}} - \mathbf{J} \cdot \frac{h}{2} + \frac{(\mathbf{J} \cdot h)^2}{8} \tag{3.26b}$$

$$\mathbf{H_3} = \mathbf{I^{(n)}} - \mathbf{J} \cdot \frac{h}{2} + \frac{(\mathbf{J} \cdot h)^2}{8} - \frac{(\mathbf{J} \cdot h)^3}{48} \tag{3.26c}$$

$$\mathbf{H_4} = \mathbf{I^{(n)}} - \mathbf{J} \cdot \frac{h}{2} + \frac{(\mathbf{J} \cdot h)^2}{8} - \frac{(\mathbf{J} \cdot h)^3}{48} + \frac{(\mathbf{J} \cdot h)^4}{384} \tag{3.26d}$$

Podríamos incluso decidir mantener el mismo Jacobiano durante varios pasos y, en tal caso, no necesitaríamos calcular un nuevo Hesiano tampoco, salvo que decidamos cambiar el paso de integración en el medio.

Una consideración es que podemos intercambiar el orden entre el paso hacia adelante y el paso hacia atrás (por ejemplo, al intercambiar los pasos en la regla trapezoidal, resulta el método de la *regla implícita del punto medio*).

Una manera de transformar los métodos BI F–estables en métodos con dominios de estabilidad más fuertes es romper la simetría entre el paso hacia adelante y el paso hacia atrás. La idea es que el semi–paso explícito tenga una longitud $\vartheta \cdot h$, y el implícito longitud $(1-\vartheta) \cdot h$. Tales algoritmos se denominan *métodos–ϑ*. Los algoritmos son del mismo orden que el de los dos semi–pasos. Usando esta idea, se puede dar forma al dominio de estabilidad.

El caso con $\vartheta > 0.5$ no es muy interesante, pero el de $\vartheta < 0.5$ es muy útil. Este produce una serie de métodos de creciente dominio de estabilidad hasta que, con $\vartheta = 0.0$, se obtiene un conjunto de algoritmos L–estables. Las matrices $\mathbf{F}$ de los métodos–ϑ son:

$$\mathbf{F_1} = [\mathbf{I^{(n)}} - \mathbf{A}(1-\vartheta)h]^{-1} \cdot [\mathbf{I^{(n)}} + \mathbf{A}\vartheta h] \tag{3.27a}$$

$$\mathbf{F_2} = [\mathbf{I^{(n)}} - \mathbf{A}(1-\vartheta)h + \frac{(\mathbf{A}(1-\vartheta)h)^2}{2!}]^{-1} \cdot [\mathbf{I^{(n)}} + \mathbf{A}\vartheta h + \frac{(\mathbf{A}\vartheta h)^2}{2!}] \tag{3.27b}$$

$$\mathbf{F_3} = [\mathbf{I^{(n)}} - \mathbf{A}(1-\vartheta)h + \frac{(\mathbf{A}(1-\vartheta)h)^2}{2!} - \frac{(\mathbf{A}(1-\vartheta)h)^3}{3!}]^{-1} \cdot [\mathbf{I^{(n)}} + \mathbf{A}\vartheta h + \frac{(\mathbf{A}\vartheta h)^2}{2!} + \frac{(\mathbf{A}\vartheta h)^3}{3!}] \tag{3.27c}$$

$$\mathbf{F_4} = [\mathbf{I^{(n)}} - \mathbf{A}(1-\vartheta)h + \frac{(\mathbf{A}(1-\vartheta)h)^2}{2!} - \frac{(\mathbf{A}(1-\vartheta)h)^3}{3!} + \frac{(\mathbf{A}(1-\vartheta)h)^4}{4!}]^{-1} \cdot [\mathbf{I^{(n)}} + \mathbf{A}\vartheta h + \frac{(\mathbf{A}\vartheta h)^2}{2!} + \frac{(\mathbf{A}\vartheta h)^3}{3!} + \frac{(\mathbf{A}\vartheta h)^4}{4!}] \tag{3.27d}$$

La Figura 3.3 muestra los dominios de estabilidad de los métodos BI para:

$$\vartheta = 0.4 \tag{3.28}$$

Estos métodos tienen muy buenos dominios de estabilidad con un gran región inestable en el semiplano derecho del plano $(\lambda \cdot h)$.

La selección del valor de ϑ es un compromiso. Debe elegirse suficientemente grande para generar una gran región inestable del lado derecho, pero debe ser a su vez suficientemente pequeño para amortiguar apropiadamente los modos rápidos en el semiplano derecho. El algoritmo de cuarto orden de la Fig.3.3 no es más A–estable, ya que su región inestable toca el semiplano izquierdo. Dicho método se denomina (A,α)–estable, donde α es el mayor ángulo que contiene sólamente terreno estable. El método $BI4_{0.4}$ es (A,86°)–estable.

Los antes mencionados métodos BRK son casos especiales de esta nueva clase de métodos–ϑ con $\vartheta = 0$, y los algoritmos RK explícitos son también casos particulares con $\vartheta = 1$. Los métodos BI F–estable son a su vez casos particulares con $\vartheta = 0.5$.

Un conjunto de métodos BI L–estables con $\vartheta > 0$ puede construirse utilizando en el semi–paso implícito un método un orden mayor que el del semi–paso explícito. Un muy buen método de este tipo es el que combina RK4 con BRK5 con $\vartheta = 0.45$ (denominado $BI4/5_{0.45}$).

3.7. Consideraciones sobre la Precisión

Una forma de analizar la precisión de los algoritmos estudiados podría consistir en simular un sistema lineal y estacionario de segundo orden, y graficar en el plano $(\lambda \cdot h)$ los puntos en los cuales se alcanzó una determinada precisión para cada ángulo de los autovalores.

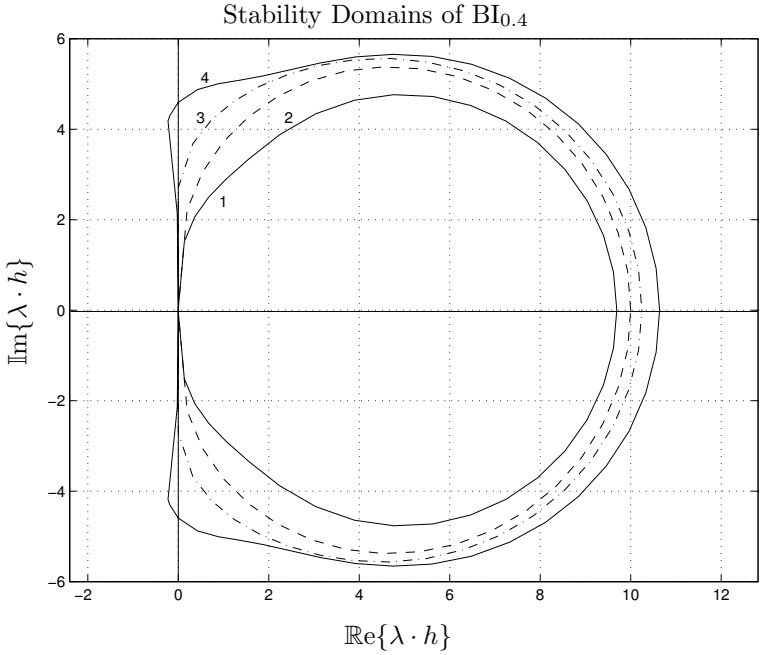

Figura 3.3: Dominios de Estabilidad de los métodos– ϑ BI.

Desafortunadamente, esto dependerá fuertemente de las condiciones iniciales adoptadas en la simulación, y el resultado de dichas gráficas no es tan interesante.

De todas maneras, antes de ver otra manera de estudiar de forma general el tema de la precisión, utilizaremos el sistema lineal y estacionario de segundo orden

$$\dot{\mathbf{x}} = \mathbf{A} \cdot \mathbf{x} \quad ; \quad \mathbf{x}(t_0) = \mathbf{x_0} \tag{3.29}$$

con la misma matriz de evolución $\mathbf{A}$ que utilizamos para construir los dominios de estabilidad:

$$\mathbf{A} = \begin{pmatrix} 0 & 1 \\ -1 & 2\cos(\alpha) \end{pmatrix} \tag{3.30}$$

y con la condición inicial estandarizada:

$$\mathbf{x_0} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \tag{3.31}$$

para la cual, la solución analítica es

$$\mathbf{x_{anal}} = \exp(\mathbf{A} \cdot (t - t_0)) \cdot \mathbf{x_o} \tag{3.32}$$

Definimos el error global como sigue:

$$\varepsilon_{\text{global}} = \|\mathbf{x_{anal}} - \mathbf{x_{simul}}\|_\infty \tag{3.33}$$

Colocando ambos autovalores en -1.0, variaremos el paso de integración h para determinar que tan precisa es cada simulación con los distintos métodos explícitos RK.

La Figura 3.4 muestra el resultado de este experimento. El número de evaluaciones de función es igual, en cada caso, al número total de pasos multiplicado por el número de etapas del método.

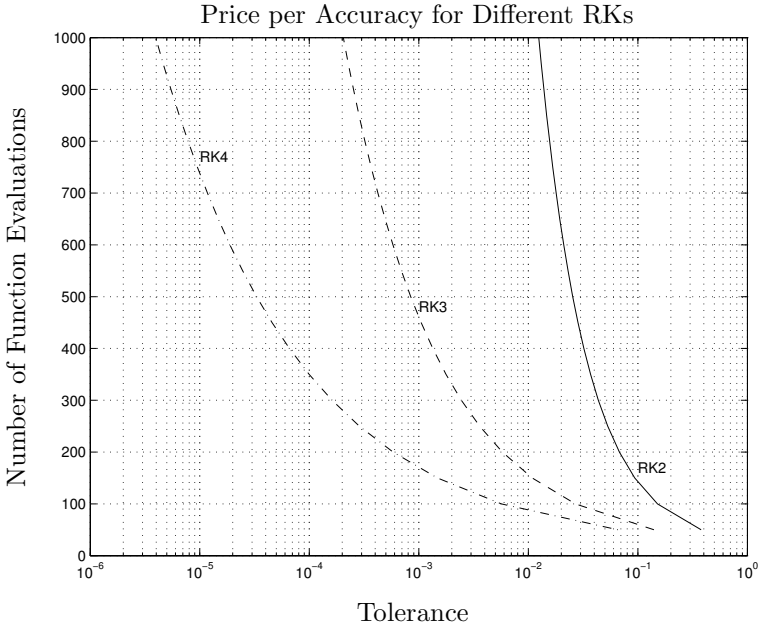

Figura 3.4: Costo de simulación en función de la precisión para diferentes RKs.

Resulta que RK4 es más económico que RK3, y éste es más económico que RK2 para todas las tolerancias. Esto no es sorprendente. Al dividir por dos el paso de integración, duplicamos el número de evaluaciones de función necesarias para terminar la simulación. Por el mismo precio, pordíamos mantener el paso de integración y duplicar el orden del método (al menos para métodos de orden bajo). O sea, podríamos ganar dos órdenes de magnitud en la mejora de la precisión contra un sólo orden al reducir el paso.

Para una precisión global del 10 %, los tres métodos tienen un precio comparable. Para un 1 % RK2 está fuera de la competencia, mientras RK3 y RK4 tienen un costo aceptable. Para un 0.1 % de precisión global, RK3 ya es demasiado costoso, mientras RK4 todavía funciona bien.

Recordando que normalmente controlamos el error *local* de integración, que es aproximadamente un orden de magnitud menor que el global, vemos que RK4 funcionará bien para un error local de hasta aproximadamente 10^{-4}. Si

queremos mejor precisión que esto, será conveniente utilizar algoritmos de orden mayor.

Veamos ahora una forma general de analizar las propiedades de estabilidad de los métodos en función de la posición de los autovalores.

Miremos una vez más nuestro sistema estándar de test de la Ec.(3.29) con la solución analítica Ec.(3.32). Obviamente, la solución analítica es válida para cualquier t, t_0, y $\mathbf{x_0}$. En particular, esto es correcto para $t_0 = t_k$, $\mathbf{x_0} = \mathbf{x_k}$ y $t = t_{k+1}$. Reemplazando con esto obtenemos

$$\mathbf{x_{k+1}} = \exp(\mathbf{A} \cdot h) \cdot \mathbf{x_k} \tag{3.34}$$

Luego, la matriz analítica $\mathbf{F}$ del sistema es:

$$\mathbf{F_{anal}} = \exp(\mathbf{A} \cdot h) \tag{3.35}$$

Tomando una componente de la solución Ec.(3.32), podemos escribir:

$$x_1(t) = c_1 \cdot \exp(-\sigma \cdot t) \cdot \cos(\omega \cdot t) + c_2 \cdot \exp(-\sigma \cdot t) \cdot \sin(\omega \cdot t) \tag{3.36}$$

donde σ es la parte real del autovalor, denominado *amortiguamiento*, mientras que ω es la parte imaginaria, denominada *frecuencia*.

Puede mostrarse muy fácilmente que los autovalores de la matriz $\mathbf{F_{anal}}$ analítica se relacionan con los de la matriz $\mathbf{A}$ mediante:

$$\text{Eig}\{\mathbf{F_{anal}}\} = \exp(\text{Eig}\{\mathbf{A}\} \cdot h) \tag{3.37}$$

o:

$$\lambda_{disc} = \exp(\lambda_{cont} \cdot h) = \exp((-\sigma + j \cdot \omega) \cdot h) = \exp(-\sigma \cdot h) \cdot \exp(j \cdot \omega \cdot h) \tag{3.38}$$

En consecuencia, el amortiguamiento σ en el plano λ se mapea como la distancia al origen en el plano $\exp(\lambda \cdot h)$, mientras que la frecuencia en el plano λ se traduce en un ángulo respecto al eje real positivo en el plano $\exp(\lambda \cdot h)$.

Notar que hemos introducido un nuevo plano: el plano $\exp(\lambda \cdot h)$. En la teoría de control, dicho plano se denomina dominio z.

$$z = \exp(\lambda \cdot h) \tag{3.39}$$

Com la matriz analítica $\mathbf{F_{anal}}$ depende del paso de integración h, tiene sentido definir el *amortiguamiento discreto* $\sigma_d = h \cdot \sigma$, y la *frecuencia discreta*, $\omega_d = h \cdot \omega$. Obviamente, podemos escribir:

$$|z| = \exp(-\sigma_d) \tag{3.40a}$$
$$\angle z = \omega_d \tag{3.40b}$$

Reemplacemos ahora la matriz analítica $\mathbf{F_{anal}}$ por la del método de integración numérica $\mathbf{F_{simul}}$. La matriz numérica $\mathbf{F_{simul}}$ es una aproximación racional

(métodos implícitos) o polinomial (métodos explícitos) de la matriz analítica $\mathbf{F_{anal}}$. Definimos:

$$\hat{z} = \exp(\hat{\lambda}_d) \tag{3.41}$$

con:

$$\hat{\lambda}_d = -\hat{\sigma}_d + j \cdot \hat{\omega}_d \tag{3.42}$$

Luego:

$$|\hat{z}| = \exp(-\hat{\sigma}_d) \tag{3.43a}$$
$$\angle\hat{z} = \hat{\omega}_d \tag{3.43b}$$

Como $\hat{z}$ aproxima a z, luego $\hat{\sigma}_d$ debería aproximar a σ_d, y $\hat{\omega}_d$ debería aproximar a ω_d.

Tiene sentido entonces estudiar la relación entre el amortiguamiento discreto analítico, σ_d, y el amortiguamiento discreto numérico, $\hat{\sigma}_d$.

De la misma forma, podemos estudiar la relación entre la frecuencia discreta analítica, ω_d, y la frecuencia discreta numérica, $\hat{\omega}_d$.

Podemos, más aún, ir moviendo el valor de σ_d y calcular los valores de $\hat{\sigma}_d$, graficando ambos valores. Dicha gráfica se denomina *gráfico de amortiguamiento*. La Figura 3.5 muestra el gráfico de amortiguamiento de RK4.

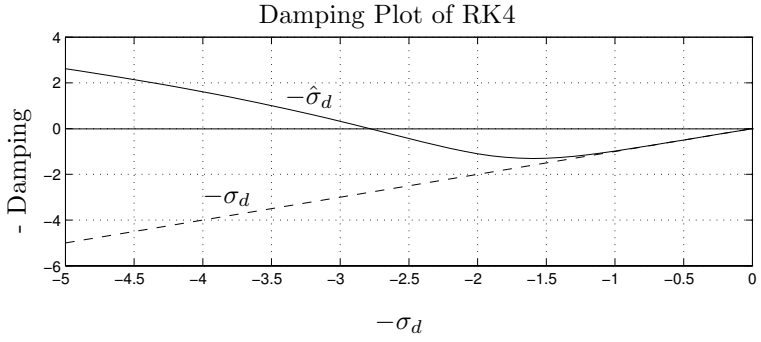

Figura 3.5: Gráfico de amortiguamiento de RK4.

En la vecindad del origen, el amortiguamiento numérico $\hat{\sigma}_d$ sigue muy bien al amortiguamiento analítico, σ_d. Sin embargo, para valores relativamente chicos el amortiguamiento numérico se desvía del analítico y, cerca de $\sigma_d = 2.8$, el amortiguamiento numérico se vuelve negativo, lo que coincide con el borde de la estabilidad numérica. La zona donde la aproximación de σ_d por $\hat{\sigma}_d$ es precisa se denomina *región asintótica*.

Veamos ahora el gráfico de amortiguamiento de BI4 (Fig.3.6).

Como BI4 es A–stable, el amortiguamiento numérico permanece siempre positivo para todos los valores de σ_d. Como BI4 es F–stable, el amortiguamiento

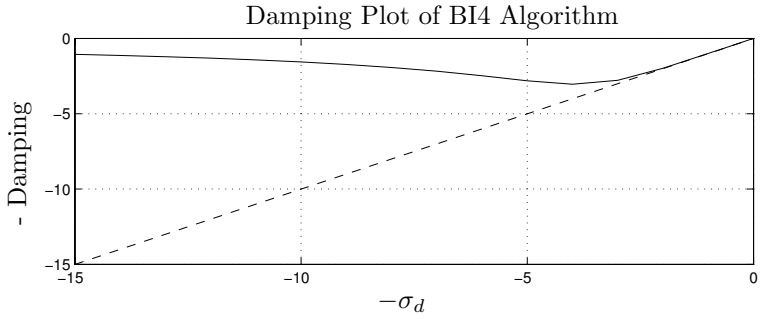

Figura 3.6: Gráfico de amortiguamiento de BI4.

numérico tiende a cero cuando λ tiende a $-\infty$. La Figura 3.7 muestra el gráfico de amortiguamiento del método ϑ con $\vartheta = 0.4$.

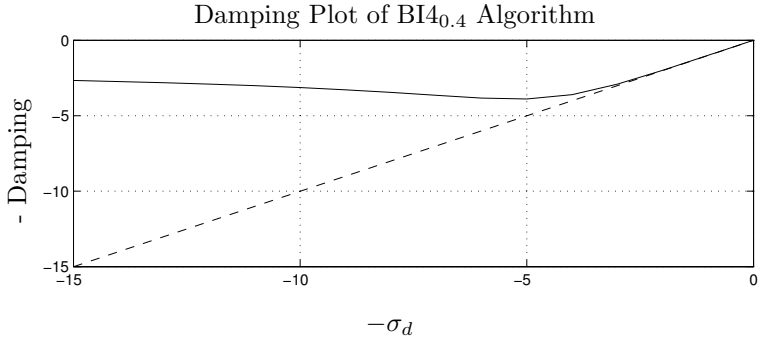

Figura 3.7: Gráfico de amortiguamiento de BI4 con $\vartheta = 0.4$.

Ahora, el amortiguamiento no se aproxima más a cero, pero tampoco tiende a infinito.

El algoritmo con $\vartheta = 0.0$, es decir, BRK4, es L–estable, ya que $\hat{\sigma}_d(-\infty) \rightarrow -\infty$. Desafortunadamente esto no quiere decir demasiado. Como puede verse en el gráfico de amortiguamiento de BRK4 en la Fig.3.8, el amortiguamiento numérico tiende muy lentamente a $-\infty$.

De la misma forma que para σ_d, podemos barrer un rango de valores de ω_d y dibujar ambas, ω_d y $\hat{\omega}_d$ en función de ω_d. Dicha gráfica se denomina *gráfico de frecuencia*. La Figura 3.9 muestra la gráfica de frecuencia de RK4.

Puede verse que tanto los gráficos de amortiguamiento como los de frecuencia tienen regiones asintóticas, es decir, regiones donde $\mathbf{F_{simul}}$ se desvía muy poco de $\mathbf{F_{anal}}$ en términos de amortiguamiento y frecuencia. La región asintótica rodea al origen, como puede verse en la Figura 3.10.

Es importante tener en cuenta las gráficas de amortiguamiento y frecuencia sólo tienen en cuenta el error local de truncamiento, ya que al comparar $\mathbf{F_{simul}}$ con $\mathbf{F_{anal}}$ están comparando la solución numérica con la solución analítica dis-

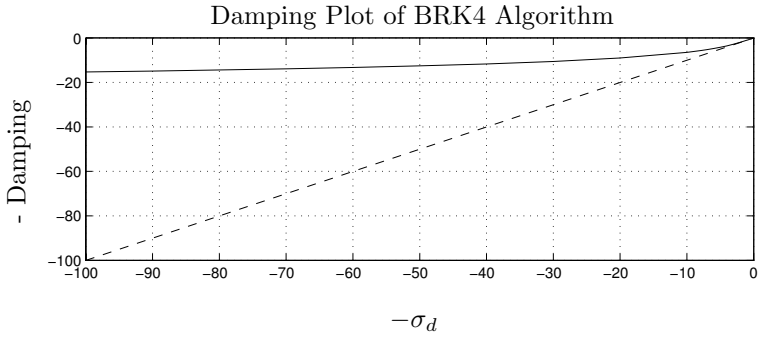

Figura 3.8: Damping plot of BRK4.

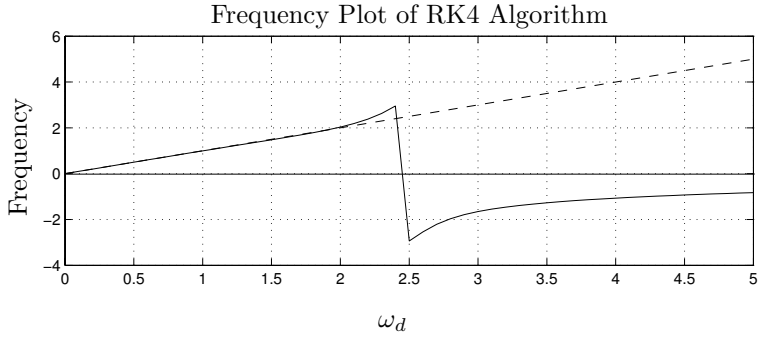

Figura 3.9: Frequency plot of RK4.

creta en un paso. Esto no tiene en cuenta ni el error de acumulación ni el error de redondeo.

3.8. Control de Paso y de Orden

Vimos hasta aquí que el uso de pasos de integración pequeños produce en general menores errores de integración pero con mayores costos computacionales. El paso de integración correcto es entonces un compromiso entre error y costo.

Como la relación error/costo con el paso de integración depende en gran medida de las propiedades numéricas del sistema a ser integrado, no está claro que un mismo paso de integración produzca el mismo error a lo largo de toda la simulación. Podría ser necesario entonces, variar el paso de integración durante la simulación para mantener el error en un nivel más o menos constante. Esta observación nos lleva a la necesidad de buscar *metodos de paso variable* que a su vez requerirán de *algoritmos de control de paso.*

El siguiente algoritmo podría funcionar. Tomemos dos algoritmos distintos RK, y repitamos dos veces el mismo paso, uno con cada algoritmo. Los resultados

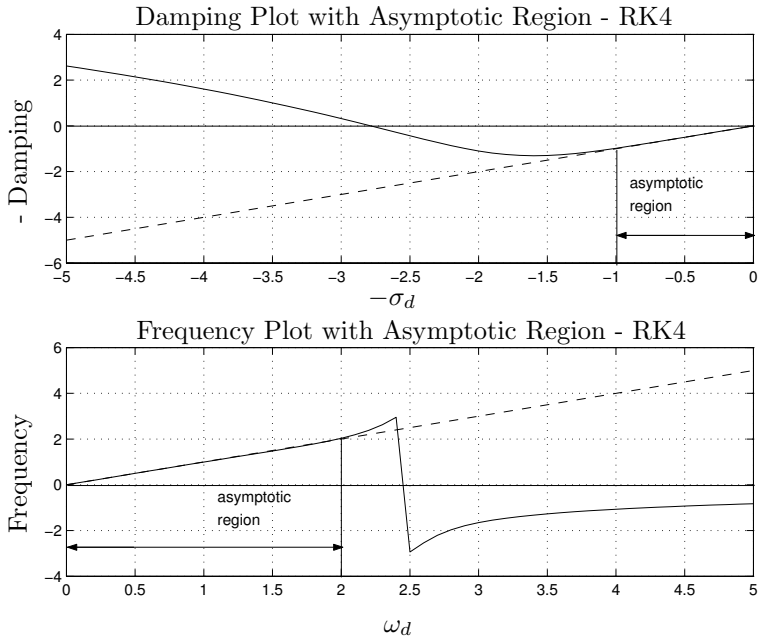

Figura 3.10: Regiones asintóticas de RK4.

de ambos diferirán en ε. Si ε es mayor que el error tolerado tol_{abs}, el paso se rechaza y se repite con la mitad del paso de integración. Si ε es menor que $0.5 \cdot tol_{\text{abs}}$ durante cuatro pasos consecutivos, el siguiente paso se computará con $1.5 \cdot h$.

Este algoritmo es excesivamente heurístico y probablemente no funcione muy bien, pero la idea es clara. La diferencia entre ambas soluciones, ε, se utiliza como estima del error de integración local y se compara con el error tolerado. Si el error estimado es mayor que el tolerado, hay que reducir el paso. Si es menor, puede aumentarse.

La primer objeción es utilizar el error absoluto como medida de performance. Tiene mucho más sentido utilizar el error relativo.

Esto también trae dificultades ya que si el estado vale 0.0 o es muy chico en algún punto habrá problemas. En este caso, llamando x_1 y x_2 a los valores arrojados por ambos métodos, conviene definir el error según

$$\varepsilon_{\text{rel}} = \frac{|x_1 - x_2|}{\text{máx}(|x_1|, |x_2|, \delta)} \tag{3.44}$$

donde δ es un factor pequeño, por ejemplo,$\delta = 10^{-10}$.

Notar que podríamos definir una tolerancia de error relativo distinta para cada variable de estado, lo que daría n condiciones distintas sobre el paso (y habría que tomar la menor).

Otro tema es el costo que hay que pagar por este procedimiento. Cada paso debe calcularse al menos dos veces. Sin embargo, esto no tiene que ser así necesariamente.

Una idea, debida a Edwin Fehlberg, es utilizar dos algoritmos de RK que coincidan en sus primeras etapas (se denominan algoritmos de RK empotrados). Particularmente, Fehlberg desarrolló pares de métodos de RK que difieren en un orden y que comparte la mayor parte de sus primeras etapas. El más utilizado de estos métodos es el método RKF4/5 que tiene tabla de Butcher:

0	0	0	0	0	0	0
1/4	1/4	0	0	0	0	0
3/8	3/32	9/32	0	0	0	0
12/13	1932/2197	−7200/2197	7296/2197	0	0	0
1	439/216	-8	3680/513	−845/4104	0	0
1/2	−8/27	2	−3544/2565	1859/4104	−11/40	0
x_1	25/216	0	1408/2565	2197/4104	−1/5	0
x_2	16/135	0	6656/12825	28561/56430	−9/50	2/55

En el caso escalar, definiendo $q \triangleq \lambda \cdot h$ resultan las matrices F (en este caso escalares):

$$F_1(q) = 1 + q + \frac{1}{2}q^2 + \frac{1}{6}q^3 + \frac{1}{24}q^4 + \frac{1}{104}q^5 \tag{3.45a}$$

$$F_2(q) = 1 + q + \frac{1}{2}q^2 + \frac{1}{6}q^3 + \frac{1}{24}q^4 + \frac{1}{120}q^5 + \frac{1}{2080}q^6 \tag{3.45b}$$

Luego, x_1 es un RK4 en cinco etapas, y x_2 es un RK5 en seis etapas. Sin mbargo, ambos algoritmos comparten las primeras cinco etapas. Luego, el método completo con el paso controlado es un algoritmo es un RK5 de seis etapas y el único costo adicional debido al control de paso es el cálculo del corrector de RK4. En definitiva, el control de paso de integración es casi gratuito.

Respecto a la heurística que planteamos antes para modificar el paso, podemos hacer algo mucho mejor. Dado que tenemos expresiones explícitas para $F_1(q)$ y $F_2(q)$, podemos dar una fórmula explícita para la estima del error:

$$\varepsilon(q) = F_1(q) - F_2(q) = \frac{1}{780}q^5 - \frac{1}{2080}q^6 \tag{3.46}$$

En una primer aproximación escribimos

$$\varepsilon \sim h^5 \tag{3.47}$$

o:

$$h \sim \sqrt[5]{\varepsilon} \tag{3.48}$$

Tiene sentido entonces el siguiente algoritmo de control de paso:

$$h_{\text{new}} = \sqrt[5]{\frac{tol_{\text{rel}} \cdot \text{máx}(|x_1|, |x_2|, \delta)}{|x_1 - x_2|}} \cdot h_{\text{old}} \tag{3.49}$$

El sentido de este algoritmo es simple. Mientras:

$$\frac{|x_1 - x_2|}{\text{máx}(|x_1|, |x_2|, \delta)} = tol_{\text{rel}} \tag{3.50}$$

tendremos el paso correcto, y el paso no cambiará. Sin embargo, cuando $|x_1 - x_2|$ sea demasiado grande, el paso deberá reducirse. Por otro lado, si $|x_1 - x_2|$ se torna muy pequeño, agrandaremos el paso de integración.

A diferencia del algoritmo antes propuesto, nunca repetiremos pasos. El algoritmo aceptará errores excesivos en cada paso, pero tratará de no repetirlos en el siguiente al disminuir el paso. Los algoritmos que hacen esto se denominan *algoritmos optimistas*, mientras que los que repiten pasos cuando encuentran errores muy grandes se denominan *algoritmos conservadores*.

Otra idea que proponen algunos autores es variar el orden, en lugar de variar el paso. Sin embargo, esta idea no es muy buena con los métodos de RK. Luego la veremos en el contexto de los métodos multipaso.

3.9. Problemas Propuestos

[P3.1] Familia de Métodos RK2 Explícitos Verificar que la Ec.(3.10) es en efecto la expansión correcta en series de Taylor que describe todos los métodos explícitos de RK2 en dos etapas.

[P3.2] Familia de Métodos RK3 Explícitos Deducir las ecuaciones de restricción en los parámetros α_i y β_{ij} que caracterizan la familia de todos los métodos explícitos RK3 en tres etapas.

Para esto, la serie de Taylor de $f(x + \Delta x, y + \Delta y)$ deberá expandirse hasta el término cuadrático:

$$f(x + \Delta x, y + \Delta y) \approx f(x, y) + \frac{\partial f(x, y)}{\partial x} \cdot \Delta x + \frac{\partial f(x, y)}{\partial y} \cdot \Delta y + \frac{\partial^2 f(x, y)}{\partial x^2} \cdot \frac{\Delta x^2}{2} + \frac{\partial^2 f(x, y)}{\partial x \cdot \partial y} \cdot \Delta x \cdot \Delta y + \frac{\partial^2 f(x, y)}{\partial y^2} \cdot \frac{\Delta y^2}{2} \tag{P3.2a}$$

[P3.3] Algoritmo de Runge–Kutta–Simpson Dado el método de Runge–Kutta en cuatro etapas caracterizado por la tabla de Butcher:

0	0	0	0	0
1/3	1/3	0	0	0
2/3	-1/3	1	0	0
1	1	-1	1	0
x	1/8	3/8	3/8	1/8

Escribir las etapas del método. Determinar el orden lineal de precisión de este método. Comparar este método con otros algoritmos de RK vistos en el capítulo.

[P3.4] Aumento del Orden en RK por Combinación Dados dos métodos distintos de RK de orden de aproximación n^{th} de al menos $(n+1)$ etapas:

$$F_1(q) = 1 + q + \frac{q^2}{2!} + \cdots + \frac{q^n}{n!} + c_1 \cdot q^{n+1} \tag{P3.4a}$$

$$F_2(q) = 1 + q + \frac{q^2}{2!} + \cdots + \frac{q^n}{n!} + c_2 \cdot q^{n+1} \tag{P3.4b}$$

donde $c_2 \neq c_1$.

Mostrar que siempre es posible utilizar la combinación (blending):

$$\mathbf{x^{blended}} = \vartheta \cdot \mathbf{x^1} + (1 - \vartheta) \cdot \mathbf{x^2} \tag{P3.4c}$$

donde $\mathbf{x^1}$ es la solución utilizando $F_1(q)$ y $\mathbf{x^2}$ es la solución utilizando $F_2(q)$, de forma tal que $\mathbf{x^{blended}}$ tiene orden $(n+1)$.

Encontrar una fórmula para ϑ que haga que el algoritmo combinado tenga orden $(n+1)$.

[P3.5] Dominio de Estabilidad de RKF4/5 Encontrar los dominios de estabilidad de los dos algoritmos utilizados en RKF4/5. Interpretar y decidir cual de los dos resultados (el de cuarto o el de quinto orden) conviene utilizar para continuar con el siguiente paso.

[P3.6] Integración con Runge–Kutta Dado el sistema lineal y estacionario:

$$\begin{aligned} \dot{\mathbf{x}} &= \begin{pmatrix} 1250 & -25113 & -60050 & -42647 & -23999 \\ 500 & -10068 & -24057 & -17092 & -9613 \\ 250 & -5060 & -12079 & -8586 & -4826 \\ -750 & 15101 & 36086 & 25637 & 14420 \\ 250 & -4963 & -11896 & -8438 & -4756 \end{pmatrix} \cdot \mathbf{x} + \begin{pmatrix} 5 \\ 2 \\ 1 \\ -3 \\ 1 \end{pmatrix} \cdot u \\ \mathbf{y} &= \begin{pmatrix} -1 & 26 & 59 & 43 & 23 \end{pmatrix} \cdot \mathbf{x} \end{aligned} \tag{P3.6a}$$

con condiciones iniciales:

$$\mathbf{x_0} = \begin{pmatrix} 1 & -2 & 3 & -4 & 5 \end{pmatrix}^T \tag{P3.6b}$$

(es el mismo sistema utilizado en el Prob.[P2.1]. Simular el sistema durante 10 segundos con la entrada escalón utilizando el método RK4 de la página 33. Probar con los siguientes pasos de integración:

1. $h = 0.32$,
2. $h = 0.032$,
3. $h = 0.0032$.

Dibujar las tres trayectorias superpuestas y concluir sobre la precisión de los resultados.

[**P3.7**] **Integración con BRK4** Repetir el problema anterior utilizando un BRK4 basado en el RK4 de la página 33.

Como el sistema es lineal, puede utilizarse inversión matricial en lugar de la iteración de Newton.

[**P3.8**] **Dominio de Estabilidad de BI4/5$_\vartheta$** Encontrar los dominios de estabilidad de BI4/5$_\vartheta$ utilizando la aproximación de Ec.(3.45a) para el semi–paso explícito, y la aproximación de la Ec.(3.45b) para el semi–paso implícito, usando los siguientes valores de ϑ:

$$\vartheta = \{0.4, 0.45, 0.475, 0.48, 0.5\} \tag{P3.8a}$$

[**P3.9**] **BI4/5$_{0.45}$ en Sistemas Lineales** Repetir el Prob.[P3.6] utilizando BI4/5$_{0.45}$. El semipaso explícito utiliza la aproximación de cuarto orden de RKF4/5 (no hace falta calcular el corrector de quinto orden). El semipaso implícito, en tanto, utiliza la aproximación de quinto orden (y no hay necesidad de calcular el corrector de cuarto orden).

Como el sistema es lineal, puede reemplazarse la iteración de Newton por inversión matricial.

Comparar la precisión de este algoritmo con la de RK4 y BRK4.

[**P3.10**] **Algoritmos RK Empotrados** Dados los siguientes dos algoritmos RK empotrados:

0	0	0	0
1	1	0	0
1/2	1/4	1/4	0
x_1	1/2	1/2	0
x_2	1/6	1/6	2/3

Escribir las etapas de los métodos y determinar el orden lineal de precisión de la aproximación de cada uno de ellos.

Capítulo 4

Métodos Multipaso

4.1. Introducción

En el capítulo anterior, vimos métodos de integración que, de una forma u otra, tratan de aproximar la expansión de Taylor de la solución desconocida en torno al instante de tiempo actual. La idea básica era no calcular las derivadas superiores en forma explícitas, sino reemplazarlas por distintas evaluaciones de la función $\mathbf{f}$ en varios puntos diferentes dentro del paso de integración.

Una desventaja de este enfoque es que cada vez que comenzamos un nuevo paso, dejamos de lado todas las evaluaciones anteriores de la función $\mathbf{f}$ en el paso anterior.

Veremos entonces en este capítulo otra familia de métodos que, en lugar de evaluar varias veces la función en un paso para incrementar el orden de la aproximación, tratan de aprovechar las evaluaciones realizadas en pasos anteriores.

Para esto, lograr esto, estos nuevos métodos utilizan como base polinomios de interpolación o de extrapolación de orden alto. Por esto, antes de presentar esta nueva familia de métodos, vamos a estudiar los *Polinomios de Newton–Gregory*.

4.2. Polinomios de Newton–Gregory

Dada una función del tiempo, $f(t)$, llamaremos a los valores de dicha función en los puntos t_0, t_1, t_2, etc. como f_0, f_1, f_2, etc. Introduciremos el *operador diferencia hacia adelante* Δ, tal que $\Delta f_0 = f_1 - f_0$, $\Delta f_1 = f_2 - f_1$, etc.

En base a este operador, podemos definir operadores de orden superior:

$$\Delta^2 f_0 = \Delta(\Delta f_0) = \Delta(f_1 - f_0) = \Delta f_1 - \Delta f_0 = f_2 - 2f_1 + f_0 \tag{4.1a}$$

$$\Delta^3 f_0 = \Delta(\Delta^2 f_0) = f_3 - 3f_2 + 3f_1 - f_0 \tag{4.1b}$$

etc.

En general:

$$\Delta^n f_i = f_{i+n} - n\cdot f_{i+n-1} + \frac{n(n-1)}{2!}\cdot f_{i+n-2} - \frac{n(n-1)(n-2)}{3!}\cdot f_{i+n-3} + \dots \quad (4.2)$$

o:

$$\Delta^n f_i = \binom{n}{0} f_{i+n} - \binom{n}{1} f_{i+n-1} + \binom{n}{2} f_{i+n-2} - \binom{n}{3} f_{i+n-3} + \cdots \pm \binom{n}{n} f_i \quad (4.3)$$

Supongamos ahora que los puntos de tiempo t_i están equiespaciados por una distancia fija h. Nos proponemos encontrar un polinomio de interpolación (o extrapolación) de orden n que pase por los $n+1$ valores conocidos de la función $f_0, f_1, f_2, \dots, f_n$ en los instantes $t_0, t_1 = t_0 + h, t_2 = t_0 + 2h, \dots, t_n = t_0 + n\cdot h$.

Para esto, vamos a introducir una variable auxiliar s, definida como:

$$s \triangleq \frac{t - t_0}{h} \quad (4.4)$$

Luego, para $t = t_0 \leftrightarrow s = 0.0$, para $t = t_1 \leftrightarrow s = 1.0$, etc. La variable real s toma valores enteros en los instantes de muestreo (e iguales al índice de la muestra).

El polinomio de interpolación puede escribirse como una función de s:

$$f(s) \approx \binom{s}{0} f_0 + \binom{s}{1} \Delta f_0 + \binom{s}{2} \Delta^2 f_0 + \cdots + \binom{s}{n} \Delta^n f_0 \quad (4.5)$$

Este polinomio se llama *Polinomio de Newton–Gregory hacia adelante*. Puede verse fácilmente que es un polinomio de orden n en t, y que efectivamente pasa por los puntos f_i en los instantes t_i.

A veces, vamos a necesitar polinomios de orden n que pasen a través de $(n+1)$ puntos del pasado. El *Polinomio de Newton–Gregory hacia atrás* se puede escribir como:

$$\begin{aligned} f(s) \quad \approx \quad & f_0 + \binom{s}{1} \Delta f_{-1} + \binom{s+1}{2} \Delta^2 f_{-2} + \binom{s+2}{3} \Delta^3 f_{-3} + \dots \\ & + \binom{s+n-1}{n} \Delta^n f_{-n} \end{aligned} \quad (4.6)$$

Para simplificar la notación, podemos utilizar el *operador de diferencia hacia atrás*, ∇, definido como:

$$\nabla f_i = f_i - f_{i-1} \quad (4.7)$$

y los operadores de orden superior:

$$\begin{aligned} \nabla^2 f_i &= \nabla(\nabla f_i) = \nabla(f_i - f_{i-1}) = \nabla f_i - \nabla f_{i-1} \\ &= f_i - 2\, f_{i-1} + f_{i-2} \quad (4.8a) \\ \nabla^3 f_i &= \nabla(\nabla^2 f_i) = f_i - 3 f_{i-1} + 3 f_{i-2} - f_{i-3} \quad (4.8b) \end{aligned}$$

etc.

o, en general:

$$\nabla^n f_i = \binom{n}{0} f_i - \binom{n}{1} f_{i-1} + \binom{n}{2} f_{i-2} - \binom{n}{3} f_{i-3} + \cdots \pm \binom{n}{n} f_{i-n} \tag{4.9}$$

El polinomio de Newton–Gregory hacia atrás entonces puede expresarse utilizando el operador ∇ como:

$$f(s) \approx f_0 + \binom{s}{1} \nabla f_0 + \binom{s+1}{2} \nabla^2 f_0 + \binom{s+2}{3} \nabla^3 f_0 + \cdots + \binom{s+n-1}{n} \nabla^n f_0 \tag{4.10}$$

4.3. Integración Numérica Mediante Extrapolación Polinómica

La idea en que se bsan los métodos multipaso es trivial. Podemos usar un polinomio de Newton–Gregory hacia atrás, elegir $t_k = t_0$ y evaluar para $s = 1.0$. Esto debería darnos una estimación de $x(t_{k+1}) = f_1$. Los valores pasados f_0, f_{-1}, f_{-2}, etc. son las soluciones previamente calculadas $x(t_k)$, $x(t_{k-1})$, $x(t_{k-2})$, etc.

Si bien presentamos los polinomios de Newton Gregory para el caso escalar, el concepto se extiende sin complicaciones para el caso vectorial.

Será necesario de todas formas modificar algo para incluir los valores de $\dot{f}$, ya que los mismos contienen la información sobre el modelo de las ecuaciones de estado.

4.4. Fórmulas Explícitas de Adams–Bashforth

Escribamos entonces el polinomio de Newton–Gregory hacia atrás para la derivada del vector de estado $\dot{\mathbf{x}}(t)$ en torno al instante t_k:

$$\dot{\mathbf{x}}(t) = \mathbf{f}_k + \binom{s}{1} \nabla \mathbf{f}_k + \binom{s+1}{2} \nabla^2 \mathbf{f}_k + \binom{s+2}{3} \nabla^3 \mathbf{f}_k + \ldots \tag{4.11}$$

donde:

$$\mathbf{f}_k = \dot{\mathbf{x}}(t_k) = \mathbf{f}(\mathbf{x}(t_k), t_k) \tag{4.12}$$

Si queremos una expresión para $\mathbf{x}(t_{k+1})$, debemos integrar la Ec.(4.11) en el intervalo $[t_k, t_{k+1}]$:

$$\begin{aligned}
&\int_{t_k}^{t_{k+1}} \dot{\mathbf{x}}(t)dt = \mathbf{x}(t_{k+1}) - \mathbf{x}(t_k) \\
&= \int_{t_k}^{t_{k+1}} \left[\mathbf{f}_k + \binom{s}{1}\nabla\mathbf{f}_k + \binom{s+1}{2}\nabla^2\mathbf{f}_k + \binom{s+2}{3}\nabla^3\mathbf{f}_k + \dots\right] dt \\
&= \int_{0.0}^{1.0} \left[\mathbf{f}_k + \binom{s}{1}\nabla\mathbf{f}_k + \binom{s+1}{2}\nabla^2\mathbf{f}_k + \binom{s+2}{3}\nabla^3\mathbf{f}_k + \dots\right] \cdot \frac{dt}{ds} \cdot ds \qquad (4.13)
\end{aligned}$$

Entonces:

$$\begin{aligned}
\mathbf{x}(t_{k+1}) =& \mathbf{x}(t_k) + h\int_0^1 \left[\mathbf{f}_k + s\nabla\mathbf{f}_k + \left(\frac{s^2}{2} + \frac{s}{2}\right)\nabla^2\mathbf{f}_k\right. \\
&\left. + \left(\frac{s^3}{6} + \frac{s^2}{2} + \frac{s}{3}\right)\nabla^3\mathbf{f}_k + \dots\right] ds \qquad (4.14)
\end{aligned}$$

y luego:

$$\mathbf{x}(t_{k+1}) = \mathbf{x}(t_k) + h\left(\mathbf{f}_k + \frac{1}{2}\nabla\mathbf{f}_k + \frac{5}{12}\nabla^2\mathbf{f}_k + \frac{3}{8}\nabla^3\mathbf{f}_k + \dots\right) \qquad (4.15)$$

Truncando la Ec.(4.15) tras el término cuadrático, y expandiendo el operador ∇, obtenemos:

$$\mathbf{x}(t_{k+1}) = \mathbf{x}(t_k) + \frac{h}{12}\left(23\mathbf{f}_k - 16\mathbf{f}_{k-1} + 5\mathbf{f}_{k-2}\right) \qquad (4.16)$$

que es el famoso método de Adams–Bashforth de tercer orden (AB3).

Si truncamos la Ec.(4.15) tras el término cúbico, obtenemos:

$$\mathbf{x}(t_{k+1}) = \mathbf{x}(t_k) + \frac{h}{24}\left(55\mathbf{f}_k - 59\mathbf{f}_{k-1} + 37\mathbf{f}_{k-2} - 9\mathbf{f}_{k-3}\right) \qquad (4.17)$$

que corresponde al método AB4.

La familia de métodos AB puede representarte mediante un vector α y una matriz β:

$$\alpha = \begin{pmatrix} 1 & 2 & 12 & 24 & 720 & 1440 \end{pmatrix}^T \qquad (4.18a)$$

$$\beta = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & -1 & 0 & 0 & 0 & 0 \\ 23 & -16 & 5 & 0 & 0 & 0 \\ 55 & -59 & 37 & -9 & 0 & 0 \\ 1901 & -2774 & 2616 & -1274 & 251 & 0 \\ 4277 & -7923 & 9982 & -7298 & 2877 & -475 \end{pmatrix} \qquad (4.18b)$$

Aquí, la i^{esima} fila contiene los coeficientes del método ABi. En la matriz β están los términos que multiplican los vectores $\mathbf{f}$ en los distintos puntos de tiempo, y en el vector α se encuentra el denominador común.

Todos estos algoritmos son explícitos. AB1 es:

$$\mathbf{x}(t_{k+1}) = \mathbf{x}(t_k) + \frac{h}{1}\left(1\mathbf{f}_k\right) \tag{4.19}$$

que se reconoce enseguida como Forward Euler.

Veamos los dominios de estabilidad de los métodos ABi. Para esto, debemos encontrar las matrices $\mathbf{F}$. Tomemos como ejemplo AB3. Aplicando Ec.(4.16) al problema lineal y estacionario encontramos:

$$\mathbf{x}(t_{k+1}) = \left[\mathbf{I^{(n)}} + \frac{23}{12}\mathbf{A}h\right]\mathbf{x}(t_k) - \frac{4}{3}\mathbf{A}h\ \mathbf{x}(t_{k-1}) + \frac{5}{12}\mathbf{A}h\ \mathbf{x}(t_{k-2}) \tag{4.20}$$

Podemos transformar esta ecuación en diferencias vectorial de tercer orden en tres ecuaciones en diferencias de primer orden definiendo las variables:

$$\mathbf{z_1}(t_k) = \mathbf{x}(t_{k-2}) \tag{4.21a}$$
$$\mathbf{z_2}(t_k) = \mathbf{x}(t_{k-1}) \tag{4.21b}$$
$$\mathbf{z_3}(t_k) = \mathbf{x}(t_k) \tag{4.21c}$$

Reemplazando, queda:

$$\mathbf{z_1}(t_{k+1}) = \mathbf{z_2}(t_k) \tag{4.22a}$$
$$\mathbf{z_2}(t_{k+1}) = \mathbf{z_3}(t_k) \tag{4.22b}$$
$$\mathbf{z_3}(t_{k+1}) = \frac{5}{12}\mathbf{A}h\ \mathbf{z_1}(t_k) - \frac{4}{3}\mathbf{A}h\ \mathbf{z_2}(t_k) + \left[\mathbf{I^{(n)}} + \frac{23}{12}\mathbf{A}h\right]\ \mathbf{z_3}(t_k) \tag{4.22c}$$

o, en forma matricial:

$$\mathbf{z}(t_{k+1}) = \begin{pmatrix} \mathbf{O^{(n)}} & \mathbf{I^{(n)}} & \mathbf{O^{(n)}} \\ \mathbf{O^{(n)}} & \mathbf{O^{(n)}} & \mathbf{I^{(n)}} \\ \frac{5}{12}\mathbf{A}h & -\frac{4}{3}\mathbf{A}h & (\mathbf{I^{(n)}} + \frac{23}{12}\mathbf{A}h) \end{pmatrix} \cdot \mathbf{z}(t_k) \tag{4.23}$$

Luego, para una matriz $\mathbf{A}$ de 2×2 , obtenemos una matriz $\mathbf{F}$ de $2i \times 2i$ $\mathbf{F}$ al usar ABi.

Los dominios de estabilidad que resultan de estas matrices $\mathbf{F}$ se muestran en la Fig.4.1.

Como los métodos ABi son explícitos, sus bordes de estabilidad se cierran en el semiplano complejo $\lambda \cdot h$.

Desafortunadamente, los dominios de estabilidad son bastante decepcionantes, ya que al aumentar el orden (lo que nos permitiría utilizar pasos más grandes) los dominios se achican (y la estabilidad nos limitará el tamaño de los pasos).

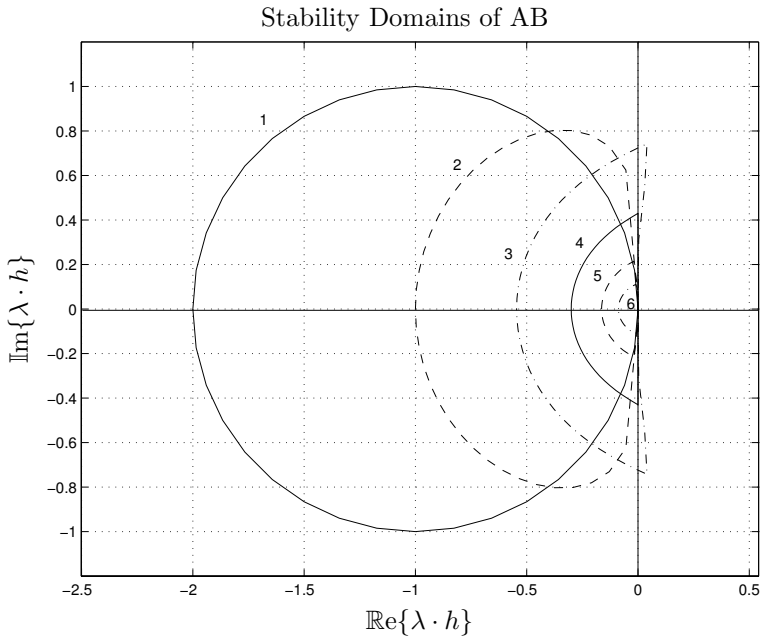

Figura 4.1: Dominios de estabilidad de los algoritmos explícitos AB.

En comparación con RK, si bien es cierto que necesitamos sólamente una evaluación de la función por paso, es probable que debamos utilizar pasos considerablemente menores debido a la región de estabilidad.

4.5. Fórmulas Implícitas de Adams–Moulton

Veamos ahora si tenemos más suerte con los algoritmos implícitos. Para esto, desarrollaremos nuevamente $\dot{\mathbf{x}}(t)$ en polinomios de Newton–Gregory hacia atrás, pero esta vez en torno al punto t_{k+1}.

$$\dot{\mathbf{x}}(t) = \mathbf{f}_{k+1} + \binom{s}{1}\nabla\mathbf{f}_{k+1} + \binom{s+1}{2}\nabla^2\mathbf{f}_{k+1} + \binom{s+2}{3}\nabla^3\mathbf{f}_{k+1} + \dots \quad (4.24)$$

Integraremos nuevamente desde t_k hasta t_{k+1}. Sin embargo, ahora $s = 0.0$ corresponde a $t = t_{k+1}$. Entonces, necesitamos intergar en el intervalo $s \in [-1.0, 0.0]$. Resulta:

$$\mathbf{x}(t_{k+1}) = \mathbf{x}(t_k) + h\left(\mathbf{f}_{k+1} - \frac{1}{2}\nabla\mathbf{f}_{k+1} - \frac{1}{12}\nabla^2\mathbf{f}_{k+1} - \frac{1}{24}\nabla^3\mathbf{f}_{k+1} + \dots\right)$$

Expandiendo el operador ∇ y truncando tras el término cuadrático, encontramos:

$$\mathbf{x}(t_{k+1}) = \mathbf{x}(t_k) + \frac{h}{12}\left(5\mathbf{f}_{k+1} + 8\mathbf{f}_k - \mathbf{f}_{k-1}\right) \tag{4.25}$$

que es el algoritmo implícito de Adams–Moulton de tercer orden (AM3).

Truncando ahora tras el término cúbico queda:

$$\mathbf{x}(t_{k+1}) = \mathbf{x}(t_k) + \frac{h}{24}\left(9\mathbf{f}_{k+1} + 19\mathbf{f}_k - 5\mathbf{f}_{k-1} + \mathbf{f}_{k-2}\right) \tag{4.26}$$

lo que corresponde al algoritmo AM4.

La familia de métodos AM también puede representarte mediante un vector α y una matriz β:

$$\alpha = \begin{pmatrix} 1 & 2 & 12 & 24 & 720 & 1440 \end{pmatrix} \tag{4.27a}$$

$$\beta = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 5 & 8 & -1 & 0 & 0 & 0 \\ 9 & 19 & -5 & 1 & 0 & 0 \\ 251 & 646 & -264 & 106 & -19 & 0 \\ 475 & 1427 & \text{-}798 & 482 & \text{-}173 & 27 \end{pmatrix} \tag{4.27b}$$

Resulta claro que AM1 es el mismo método que BE. El método de segundo orden AM2 coincide con la regla trapezoidal. Es decir, AM1 es L–estable y AM2 es F–estable.

Veamos ahora los dominios de estabilidad de los métodos de orden superior. Utilizando AM3 con un sistema lineal y estacionario resulta:

$$\left[\mathbf{I}^{(\mathbf{n})} - \frac{5}{12}\mathbf{A}h\right]\mathbf{x}(t_{k+1}) = \left[\mathbf{I}^{(\mathbf{n})} + \frac{2}{3}\mathbf{A}h\right]\mathbf{x}(t_k) - \frac{1}{12}\mathbf{A}h\ \mathbf{x}(t_{k-1}) \tag{4.28}$$

Cambiando variables como en ABi, tenemos:

$$\mathbf{F} = \begin{pmatrix} \mathbf{O}^{(\mathbf{n})} & \mathbf{I}^{(\mathbf{n})} \\ -[\mathbf{I}^{(\mathbf{n})} - \frac{5}{12}\mathbf{A}h]^{-1} \cdot [\frac{1}{12}\mathbf{A}h] & [\mathbf{I}^{(\mathbf{n})} - \frac{5}{12}\mathbf{A}h]^{-1} \cdot [\mathbf{I}^{(\mathbf{n})} + \frac{2}{3}\mathbf{A}h] \end{pmatrix} \tag{4.29}$$

Para una matriz de 2×2 $\mathbf{A}$, tenemos una matriz $\mathbf{F}$ de $2(i-1) \times 2(i-1)$ al usar AMi. Los dominios de estabilidad de los métodos hasta AM6 se muestran en la Fig.4.2.

Como en el caso de los algoritmos ABi, los resultados son decepcionantes. AM1 y AM2 son algoritmos útiles . . . pero ya los conocíamos con otros nombres. A partir de AM3, los dominios de estabilidad se cierran sobre el semiplano izquierdo. Entonces, en principio, no parece tener sentido pagar el precio de utilizar iteraciones para obtener un método que no sirve para sistemas stiff.

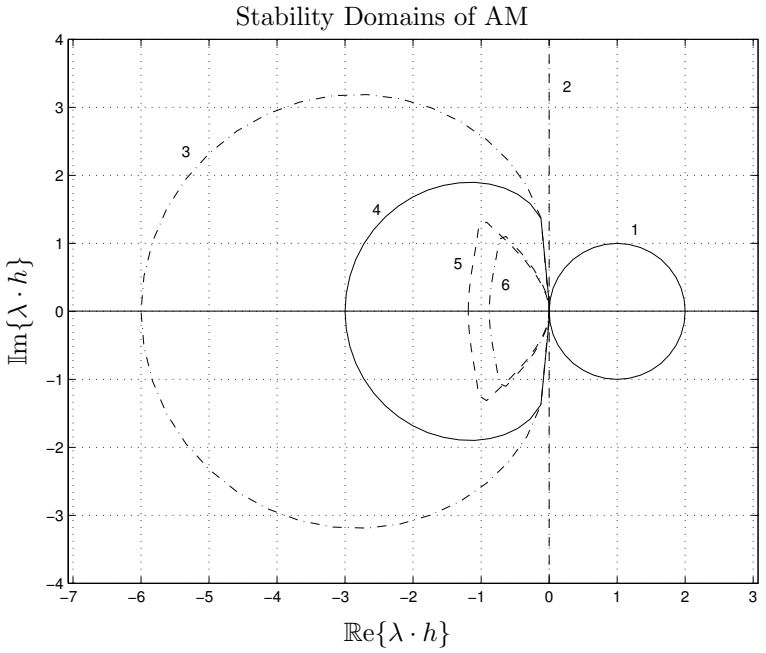

Figura 4.2: Dominios de estabilidad de los algoritmos implícitos de AM.

4.6. Fórmulas Predictor–Corrector de Adams–Bashforth–Moulton

Los métodos de ABi y de AMi por sí solos no tienen casi ventajas debido a que sus regiones de estabilidad son muy pobres.

Intentaremos ahora construir un método predictor–corrector con un paso de ABi (predictor) y uno de AMi (corrector). Con esta idea, el método de tercer orden ABM3 será el siguiente:

$$\begin{aligned}
\text{predictor:}\quad & \dot{\mathbf{x}}_{\mathbf{k}} = \mathbf{f}(\mathbf{x}_{\mathbf{k}}, t_k) \\
& \mathbf{x}^{\mathbf{P}}_{\mathbf{k+1}} = \mathbf{x}_{\mathbf{k}} + \tfrac{h}{12}(23\dot{\mathbf{x}}_{\mathbf{k}} - 16\dot{\mathbf{x}}_{\mathbf{k-1}} + 5\dot{\mathbf{x}}_{\mathbf{k-2}}) \\
\text{corrector:}\quad & \dot{\mathbf{x}}^{\mathbf{P}}_{\mathbf{k+1}} = \mathbf{f}(\mathbf{x}^{\mathbf{P}}_{\mathbf{k+1}}, t_{k+1}) \\
& \mathbf{x}^{\mathbf{C}}_{\mathbf{k+1}} = \mathbf{x}_{\mathbf{k}} + \tfrac{h}{12}(5\dot{\mathbf{x}}^{\mathbf{P}}_{\mathbf{k+1}} + 8\dot{\mathbf{x}}_{\mathbf{k}} - \dot{\mathbf{x}}_{\mathbf{k-1}})
\end{aligned}$$

Evidentemente el algoritmo completo es explícito y no se necesita ninguna iteración de Newton. Sin embargo, hay un costo adicional respecto de AB3 al tener que evaluar dos veces la función por cada paso. Veremos si este costo se compensa con una región de estabilidad más grande.

Reemplazando para la ecuación lineal y estacionaria queda:

$$\mathbf{x}(t_{k+1}) = \left[\mathbf{I^{(n)}} + \frac{13}{12}\mathbf{A}h + \frac{115}{144}(\mathbf{A}h)^2\right]\mathbf{x}(t_k) - \left[\frac{1}{12}\mathbf{A}h + \frac{5}{9}(\mathbf{A}h)^2\right]\mathbf{x}(t_{k-1}) + \frac{25}{144}(\mathbf{A}h)^2\mathbf{x}(t_{k-2}) \tag{4.30}$$

con la matriz $\mathbf{F}$:

$$\mathbf{F} = \begin{pmatrix} \mathbf{O^{(n)}} & \mathbf{I^{(n)}} & \mathbf{O^{(n)}} \\ \mathbf{O^{(n)}} & \mathbf{O^{(n)}} & \mathbf{I^{(n)}} \\ \frac{25}{144}(\mathbf{A}h)^2 & -\left[\frac{1}{12}\mathbf{A}h + \frac{5}{9}(\mathbf{A}h)^2\right] & \left[\mathbf{I^{(n)}} + \frac{13}{12}\mathbf{A}h + \frac{115}{144}(\mathbf{A}h)^2\right] \end{pmatrix} \tag{4.31}$$

Los dominios de estabilidad se muestran en la Fig.4.3.

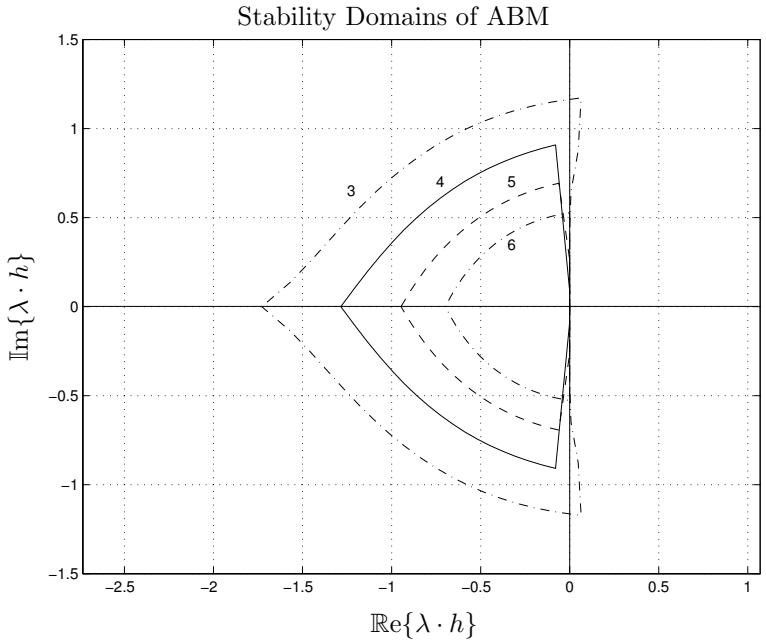

Figura 4.3: Stability domains of predictor–corrector ABM algorithms.

Evidentemente, la idea funcionó, ya que los dominios de estabilidad de ABMi son considerablemente mayores que los de ABi.

4.7. Fórmulas de Diferencias Hacia Atrás (BDF)

Veamos ahora si podemos encontrar una familia de métodos multipaso con dominios de estabilidad que se cierren a la derecha del semiplano $\lambda \cdot h$. Para

esto, escribiremos el polinomio de Newton–Gregory hacia atrás en $\mathbf{x}(t)$ en vez de en $\dot{\mathbf{x}}(t)$, en torno al instante t_{k+1}. Entonces:

$$\mathbf{x}(t) = \mathbf{x}_{k+1} + \binom{s}{1}\nabla\mathbf{x}_{k+1} + \binom{s+1}{2}\nabla^2\mathbf{x}_{k+1} + \binom{s+2}{3}\nabla^3\mathbf{x}_{k+1} + \dots \quad (4.32)$$

o:

$$\mathbf{x}(t) = \mathbf{x}_{k+1} + s\nabla\mathbf{x}_{k+1} + \left(\frac{s^2}{2} + \frac{s}{2}\right)\nabla^2\mathbf{x}_{k+1} + \left(\frac{s^3}{6} + \frac{s^2}{2} + \frac{s}{3}\right)\nabla^3\mathbf{x}_{k+1} + \dots \quad (4.33)$$

Derivando la Ec.(4.33)) respecto al tiempo obtenemos:

$$\dot{\mathbf{x}}(t) = \frac{1}{h}\left[\nabla\mathbf{x}_{k+1} + \left(s + \frac{1}{2}\right)\nabla^2\mathbf{x}_{k+1} + \left(\frac{s^2}{2} + s + \frac{1}{3}\right)\nabla^3\mathbf{x}_{k+1} + \dots\right] \quad (4.34)$$

Evaluando la Ec.(4.34) para $s = 0.0$, resulta:

$$\dot{\mathbf{x}}(t_{k+1}) = \frac{1}{h}\left[\nabla\mathbf{x}_{k+1} + \frac{1}{2}\nabla^2\mathbf{x}_{k+1} + \frac{1}{3}\nabla^3\mathbf{x}_{k+1} + \dots\right] \quad (4.35)$$

Multiplicando la Ec.4.35 por h, truncando tras el término cúbico y expandiendo los operadores ∇, queda:

$$h \cdot \mathbf{f}_{k+1} = \frac{11}{6}\mathbf{x}_{k+1} - 3\mathbf{x}_k + \frac{3}{2}\mathbf{x}_{k-1} - \frac{1}{3}\mathbf{x}_{k-2} \quad (4.36)$$

De la Ec.(4.36) se puede despejar $\mathbf{x}_{k+1}$:

$$\mathbf{x}_{k+1} = \frac{18}{11}\mathbf{x}_k - \frac{9}{11}\mathbf{x}_{k-1} + \frac{2}{11}\mathbf{x}_{k-2} + \frac{6}{11} \cdot h \cdot \mathbf{f}_{k+1} \quad (4.37)$$

que es la fórmula de tercer orden de diferencias hacia atrás (BDF3, por *Backward Difference Formula*).

Podemos obtener algoritmos de BDFi truncando la Ec.(4.35) en diferentes términos. La familia de métodos BDF también puede representarte mediante un vector α y una matriz β:

$$\alpha = \begin{pmatrix} 1 & 2/3 & 6/11 & 12/25 & 60/137 \end{pmatrix}^T \quad (4.38a)$$

$$\beta = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 4/3 & -1/3 & 0 & 0 & 0 \\ 18/11 & -9/11 & 2/11 & 0 & 0 \\ 48/25 & -36/25 & 16/25 & -3/25 & 0 \\ 300/137 & -300/137 & 200/137 & -75/137 & 12/137 \end{pmatrix} \quad (4.38b)$$

Aquí, la fila i representa el algoritmo BDFi. Los coeficientes de la matriz β son los que multiplican los valores pasados del vector de estados $\mathbf{x}$, mientras que los

coeficientes del vector α son los que multiplican el vector de las derivadas del estado $\dot{\mathbf{x}}$ en el instante t_{k+1}.

Los métodos de BDF son algoritmos implícitos. BDF1 es lo mismo que BE. Los dominios de estabilidad de los métodos BDFi se presentan en la Fig.4.4.

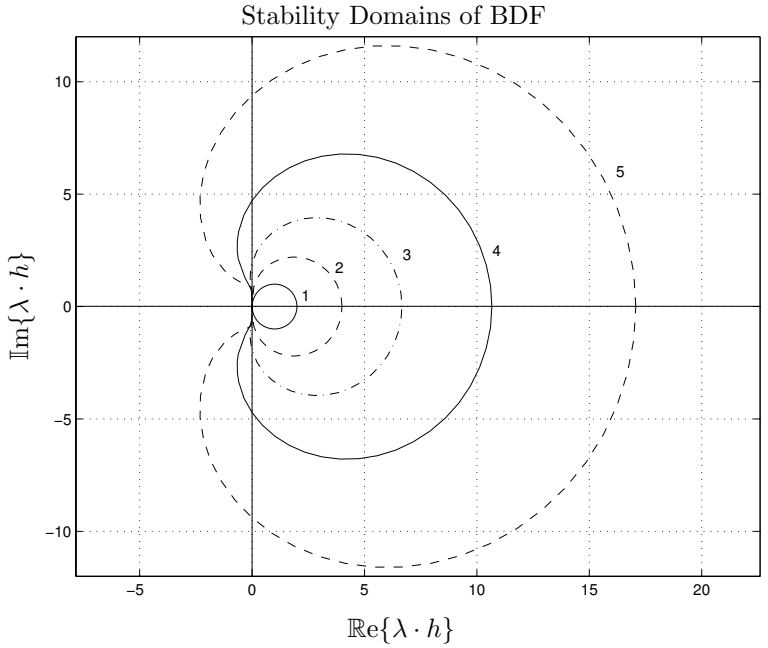

Figura 4.4: Dominios de estabilidad de los métodos implícitos BDF.

Es evidente que al fin encontramos un conjunto de métodos multipaso stiff–estables. Como en todos los otros métodos multipaso, al agrandar el orden de la extrapolación, el método se vuelve cada vez menos estable. En consecuencia, BDF6 tiene sólamente una banda muy angosta de área estable a la izquierda del origen y sólamente es útil cuando los autovalores están sobre el eje real. BDF7 es inestable en todo el plano $\lambda \cdot h$.

De todas formas, debido a su simplicidad, los métodos BDFi son los más utilizados por los paquetes de simulación de propósito general. El código basado en BDF más utilizado es DASSL.

Los métodos de BDF se conocen también como *algoritmos de Gear*, por Bill Gear, quien descubrió las propiedades de los mismos para la simulación de sistemas stiff a principios de la década de 1970.

4.8. Iteración de Newton

Los métodos BDFi pueden escribirse en función de los coeficientes α_i y $\beta_{i,j}$ de la Ec.(4.38) como sigue:

$$\mathbf{x}_{k+1} = \alpha_i \; h \cdot \mathbf{f}_{k+1} + \sum_{j=1}^{i} \beta_{ij} \; \mathbf{x}_{k-j+1} \tag{4.39}$$

Reemplazando para un sistema lineal y estacionario y despejando $\mathbf{x_{k+1}}$, queda:

$$\mathbf{x_{k+1}} = -\left[\alpha_i \cdot (\mathbf{A} \cdot h) - \mathbf{I^{(n)}}\right]^{-1} \sum_{j=1}^{i} \beta_{ij} \; \mathbf{x}_{k-j+1} \tag{4.40}$$

En un problema no lineal, no podemos aplicar inversión matricial. Podemos igualmente reescribir la Ec.(4.39) como:

$$\mathcal{F}(\mathbf{x}_{k+1}) = \alpha_i \; h \cdot \mathbf{f}(\mathbf{x}_{k+1}, t_{k+1}) - \mathbf{x}_{k+1} + \sum_{j=1}^{i} \beta_{ij} \; \mathbf{x}_{k-j+1} = 0.0 \tag{4.41}$$

y entonces se puede utilizar la iteración de Newton sobre la Ec.(4.41):

$$\mathbf{x}_{k+1}^{\ell+1} = \mathbf{x}_{k+1}^{\ell} - [\mathcal{H}^{\ell}]^{-1} \cdot [\mathcal{F}^{\ell}] \tag{4.42}$$

donde el Hessiano $\mathcal{H}$ se calcula como:

$$\mathcal{H} = \alpha_i \cdot (\mathcal{J} \cdot h) - \mathbf{I^{(n)}} \tag{4.43}$$

donde $\mathcal{J}$ es el Jacobiano del sistema. Si reemplazamos la Ec.(4.42) para un sistema lineal y estacionario, puede verse fácilmente que se obtiene nuevamente la Ec.(4.40). Es decir, la iteración de Newton no cambia el dominio de estabilidad.

Como ya mencionamos antes, en general se utiliza la iteración de Newton modificada, que no reevalúa el Jacobiano durante cada iteración. Más aún, en los métodos multipaso muchas veces no se reevalúa el Jacobiano en todos los pasos.

De esta manera, tampoco se reevalúa el Hessiano muy frecuentemente, ya que el mismo sólo debe reevaluarse cuando se reevalua el Jacobiano o cuando el paso de integración se modifica.

El Jacobiano se puede evaluar de dos maneras: numérica o simbólica.

La primera en general se hace mediante una aproximación de primer orden. Cada variable de estado x_i se modifica en un valor pequeño Δx_i, y se calcula

$$\frac{\partial \mathbf{f}(\mathbf{x}, t)}{\partial x_i} \approx \frac{\mathbf{f_{pert}} - \mathbf{f}}{\Delta x_i} \tag{4.44}$$

donde $\mathbf{f}$ es la derivada del vector de estado evaluada en el valor actual del estado, mientras que $\mathbf{f_{pert}}$ es la derivada perturbada, evaluada en $x_i + \Delta x_i$ con el resto de las variables de estado mantenidas en sus valores actuales.

Entonces, un sistema de orden n necesita n evaluaciones adicionales de la función $\mathbf{f}$ para obtener una aproximación completa del Jacobiano. Y aún en ese caso, sólo logramos una aproximación de primer orden del mismo.

Para un sistema lineal es suficiente, ya que obtendremos $\mathcal{J} = \mathbf{A}$ y la iteración de Newton convergerá en sólo un paso. En un caso no lineal en cambio, podemos llegar a tener varias iteraciones.

Por estos motivos, muchas veces es preferible evaluar el jacobiano de manera simbólica. Teniendo en cuenta que muchos paquetes de software (Dymola, Matlab, Mathematica) ofrecen herramientas de cálculo simbólico, esto no agrega dificultad al problema y permite calcular el Jacobiano de manera exacta y sin costo computacional adicional respecto de la evaluación numérica del mismo.

4.9. Control de Paso y de Orden

En los métodos de RK no tiene mucho sentido variar el orden del método durante la simulación, ya que la precisión se puede controlar mucho más fácilmente cambiando el paso de integración.

Sin embargo, en los métodos multipaso el control de orden se utiliza muy a menudo, ya que es muy simple. El orden de un método depende de cuantos valores anteriores de $\mathbf{x_k}$ estamos utilizando en los cálculos. Por lo tanto, es trivial aumentar o disminuir en 1 el orden utilizado. Además, el costo de cambiar el orden es prácticamente nulo.

El control de paso en cambio no es tan económico, ya que las fórmulas multipaso se basan en considerar que los puntos en el tiempo están equiespaciados. Aunque veremos una manera eficiente de utilizar paso variable, es mucho más sencillo el control de orden.

De todas maneras, la variación de orden produce cambios de precisión muy bruscos. Cuando cambiamos en 1 el orden del método, la precisión cambia en general en un factor de 10. Por eso, antes de bajar el orden de la aproximación, el error estimado debe ser muy chico. Además, al disminuir el orden no se ahorra casi nada en cálculos: el número de evaluaciones de la función sigue siendo el mismo.

Como podemos controlar el paso de manera eficiente? La idea no es tan complicada. Vamos a comenzar reconsiderando el polinomio de Newton–Gregor hacia atrás:

$$\mathbf{x}(t) = \mathbf{x_k} + s\nabla\mathbf{x}_k + \left(\frac{s^2}{2} + \frac{s}{2}\right)\nabla^2\mathbf{x}_k + \left(\frac{s^3}{6} + \frac{s^2}{2} + \frac{s}{3}\right)\nabla^3\mathbf{x}_k + \dots \quad (4.45)$$

Derivando respecto al tiempo obtenemos:

$$\dot{\mathbf{x}}(t) = \frac{1}{h}\left[\nabla\mathbf{x}_k + \left(s + \frac{1}{2}\right)\nabla^2\mathbf{x}_k + \left(\frac{s^2}{2} + s + \frac{1}{3}\right)\nabla^3\mathbf{x}_k + \dots\right] \quad (4.46)$$

la segunda derivada es:

$$\ddot{\mathbf{x}}(t) = \frac{1}{h^2}\left[\nabla^2\mathbf{x}_k + (s+1)\nabla^3\mathbf{x}_k + \dots\right] \quad (4.47)$$

etc.

Truncando las Ecs.(4.45)–(4.47) tras el término cúbico, expandiendo los operadores ∇ y evaluando para $t = t_k$ ($s = 0.0$), obtenemos:

$$\begin{pmatrix} x_k \\ h \cdot \dot{x}_k \\ \frac{h^2}{2} \cdot \ddot{x}_k \\ \frac{h^3}{6} \cdot x_k^{(\mathrm{iii})} \end{pmatrix} = \frac{1}{6} \cdot \begin{pmatrix} 6 & 0 & 0 & 0 \\ 11 & -18 & 9 & -2 \\ 6 & -15 & 12 & -3 \\ 1 & -3 & 3 & -1 \end{pmatrix} \cdot \begin{pmatrix} x_k \\ x_{k-1} \\ x_{k-2} \\ x_{k-3} \end{pmatrix} \tag{4.48}$$

El vector del lado izquierdo de la ecuación se denomina *vector de Nordsieck*, (en este caso escribimos el de tercer orden).

Usando esta idea, utilizando una simple multiplicación por una matriz constante, pudimos traducir información de los estados en distintos puntos del tiempo en información del estado y sus derivadas en un mismo instante de tiempo.

En este caso escribimos la transformación para una sóla variable de estado. El caso vectorial funciona de la misma forma, pero la notación es más complicada.

Esto nos permite reslver el problema de control de paso. Si queremos cambiar el paso en el instante t_k, símplemente podemos multiplicar el vector que contiene los valores pasados del estado por la matriz de transformación, tranformando así la historia del estado en un vector de Nordsieck equivalente. En esta nueva forma, el paso se puede cambiar fácilmente premultiplicando el vector de Nordsieck con la matriz diagonal:

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{h_{\mathrm{new}}}{h_{\mathrm{old}}} & 0 & 0 \\ 0 & 0 & \left(\frac{h_{\mathrm{new}}}{h_{\mathrm{old}}}\right)^2 & 0 \\ 0 & 0 & 0 & \left(\frac{h_{\mathrm{new}}}{h_{\mathrm{old}}}\right)^3 \end{pmatrix} \tag{4.49}$$

Tras esto tendremos el vector de Nordsieck expresado en el nuevo paso h_{new}. Si ahora premultiplicamos este vector de Nordsieck modificado por la inversa de la matriz de transformación, obtendremos un vector de historia de los estados modificado, donde los nuevos valores "guardados" $\mathbf{x}$ representan la solución en instantes equidistantes h_{new}.

Este es el método más utilizado para controlar el paso de integración en los métodos multipaso. De todas formas, es algo costoso ya que hay que realizar tres multiplicaciones matriciales cada vez que se quiere cambiar el paso. Además, en presencia de métodos implícitos, hay que evaluar un nuevo Hesiano tras modificar el paso de integración.

Por este motivo, no se suele cambiar muy seguido el paso. Cada vez que se reduce el paso, se suele reducir excesivamente para no tener que reducirlo nuevamente en los siguientes pasos. Asímismo, sólamente se aumenta el paso de integración cuando este aumento es considerable.

4.10. Arranque de los Métodos

Un problema que hay que resolver en los métodos multipaso es el arranque. Normalmente, se conoce el estado inicial en tiempo t_0, pero no hay datos anteriores disponibles, y no se puede entonces comenzar utilizando métodos multipaso de orden mayor que uno (en el caso implícito).

Una solución muy sencilla es utilizar control de orden: se comienza con un método de orden uno para obtener $t_1 = t_0 + h$. Luego, con estos dos puntos se puede utilizar un método de orden dos para obtener $t_2 = t_1 + h$ y así sucesivamente hasta que al cabo de varios pasos se puede obtener un método de orden n.

Este método, si bien funciona, no es muy recomendable debido al problema de la precisión. Si queremos utilizar un método de orden alto, es porque necesitamos precisión. Entonces, al comenzar con un método de orden bajo, deberíamos utilizar un paso demasiado chico en los primeros pasos. Tras esto, además, tendríamos que cambiar el paso de integración para aprovechar el algoritmo de orden alto una vez arrancado.

Otra idea es utilizar algoritmos de Runge–Kutta (del mismo orden de método multipaso a utilizar) durante el arranque, lo que nos libera del problema de usar pasos demasiado chicos.

Esta idea funciona muy bien para los métodos de tipo ABM y AB. El caso de BDF es un poco más problemático ya que al tratar con sistemas stiff, RK deberá utilizar pasos excesivamente pequeños y tendremos nuevamente un problema similar al de control de orden. Una alternativa para esto es usar métodos tipo BRK para no tener limitaciones de estabilidad en el arranque.

4.11. Problemas Propuestos

[P4.1] Nuevos métodos

Utilizando la idea del polinomio de Newton–Gregory hacia atrás, diseñar un conjunto de algoritmos de la forma:

$$\mathbf{x_{k+1}} = \mathbf{x_{k-2}} + \frac{h}{\alpha_i} \cdot \left[\sum_{j=1}^{i} \beta_{ij} \ \mathbf{x}_{k-j+1} \right] \tag{P4.1a}$$

Dibujar sus dominios de estabilidad y compararlos con los de AB.

[P4.2] Costo Versus Precisión – Caso No Stiff Calcular la gráfica de costo vs. precisión de AB4, ABM4 y AM4 para el problema lineal no stiff:

$$\dot{\mathbf{x}} = \begin{pmatrix} 1250 & -25113 & -60050 & -42647 & -23999 \\ 500 & -10068 & -24057 & -17092 & -9613 \\ 250 & -5060 & -12079 & -8586 & -4826 \\ -750 & 15101 & 36086 & 25637 & 14420 \\ 250 & -4963 & -11896 & -8438 & -4756 \end{pmatrix} \cdot \mathbf{x} + \begin{pmatrix} 5 \\ 2 \\ 1 \\ -3 \\ 1 \end{pmatrix} \cdot u \tag{P4.2a}$$

con entrada nula y condición inicial $\mathbf{x_0} = \text{ones}(5, 1)$.

Dicha gráfica debe tener en el eje horizontal la precisión obtenida y en el eje vertical el número de evaluaciones de la función $\mathbf{f}$.

Para esto, tener en cuenta que AB realiza una evaluación por paso, ABM realiza dos y consideraremos (por simplicidad) que AM realiza en promedio cuatro evaluaciones.

Si bien se pide graficar el número de evaluaciones en función de la precisión, es más eficientes variar el número de evaluaciones y calcular la precisión obtenida. Se sugiere seleccionar el número de evaluaciones entre 200 y 4000, por ejemplo, $nbrstp = [200, 500, 1000, 2000, 4000]$. Luego hay que computar el paso de integración. En el caso de AB4 sería $h = 10/nbrstp$, ya que integraremos durante 10 segundos y tenemos una evaluación por paso. En ABM4 usaríamos $h = 20/nbrstp$ y en AM4 $h = 40/nbrstp$.

Dado que el problema es lineal, se puede simular el sistema utilizando las matrices $\mathbf{F}$. Como los métodos no pueden arrancar por sí mismos, se deberá utilizar RK4 durante los 3 primeros pasos.

Para evaluar la precisión obtenida en cada simulación utilizaremos el verdadero error relativo global, calculando primero:

$$\varepsilon_{\text{local}}(t_k) = \frac{\|\mathbf{x}_{\text{anal}}(t_k) - \mathbf{x}_{\text{simul}}(t_k)\|}{\text{máx}(\|\mathbf{x}_{\text{anal}}(t_k)\|, \text{eps})} \tag{P4.2b}$$

donde *eps* es un número muy chico, y luego,

$$\varepsilon_{\text{global}} = \max_k(\varepsilon_{\text{local}}(t_k)) \tag{P4.2c}$$

[P4.3] Costo Versus Precisión – Caso Stiff

Repetiremos la idea del problema anterior pero para el siguiente caso lineal stiff:

$$\dot{\mathbf{x}} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -10001 & -10201 & -201 \end{pmatrix} \cdot \mathbf{x} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \cdot u \tag{P4.3a}$$

Buscamos calcular la respuesta al escalón con condiciones iniciales nulas hasta $t_f = 10$ seg.

En este caso utilizaremos BDF2, BDF3 y BDF4 para comparar su eficiencia relativa para resolver el problema. Consideraremos, como en el caso anterior

de AM4, que cada paso (iteración de Newton) consume 4 evaluaciones de la función.

Como referencia, también computaremos la gráfica de costo vs. precisión de BRK4. Consideraremos que cada paso de BRK4 realiza 4 iteraciones, con 4 evaluaciones de la función en cada una de ellas.

[P4.4] Backward Difference Formulae Programar en Scilab o Matlab una rutina que simule un sistema genérico utilizando BDF4, y que utilice RK4 para el arranque.

Comenzar con el caso de paso fijo, y luego implementar un algoritmo de control de paso.

Capítulo 5

Ecuaciones en Derivadas Parciales

Este capítulo es una traducción resumida del Capítulo 6 del libro *Continuous System Simulation* [2].

5.1. Introducción

La simulación de ecuaciones en derivadas parciales (PDE, por *partial differential equations*) es uno de los temas más difíciles de abordar y presenta una gran cantidad de problemas abiertos.

Si bien la literatura cuenta con numerosos enfoques distintos para la integración numérica de PDEs, en este curso trataremos con sólamente uno de estos enfoques, denominado el *Método de Líneas* (MOL, por *method of lines*).

El MOL se basa en aproximar las PDEs por un sistema grande de ecuaciones diferenciales ordinarias, que luego debe ser simulado con métodos de integración como los que estudiamos en los capítulos anteriores.

Es importante remarcar que el MOL es sólo una entre las tantas técnicas que existen para PDEs. El motivo por el cual estudiaremos el MOL no es porque se trate de la técnica más eficiente, sino que al convertir las PDEs en conjuntos de ODEs nos permitirá hacer uso de todo lo que vimos en el transucrso del curso.

De esta forma, a lo largo de este capítulo vamos a ver cómo el MOL transforma las PDEs en un sistema de ODEs y vamos a estudiar las características particulares de las ODEs resultantes y decidir cuales de los métodos de integración vistos serán convenientes para la simulación de las mismas.

5.2. El Método de Líneas

El Método de Líneas es una técnica que nos permite convertir ecuaciones en derivadas parciales en conjuntos de ecuaciones diferenciales ordinarias que, en

algún sentido, son *equivalentes* a las PDEs originales.

La idea básica es muy sencilla. Tomemos como ejemplo la *ecuación del calor* o *ecuación de difusión* en una sola variable espacial:

$$\frac{\partial u}{\partial t} = \sigma \cdot \frac{\partial^2 u}{\partial x^2} \tag{5.1}$$

En lugar de buscar la solución $u(x,t)$ en todo el espacio bidimensional definido por la variable espacial x y la variable temporal t, podemos discretizar la variable espacial y buscar las soluciones $u_i(t)$ donde el índice i denota un punto particular x_i en el espacio. Para esto, reemplazaremos la derivada parcial de segundo orden de u con respecto a x por una diferencia finita:

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{x=x_i} \approx \frac{u_{i+1} - 2u_i + u_{i-1}}{\delta x^2} \tag{5.2}$$

donde δx es la distancia entre dos puntos vecinos de discretización en el espacio (aquí elegida equidistante). Esta distancia de denomina *ancho de grilla* de la discretización.

Utilizando la Ec.(5.2) en (5.1), resulta:

$$\frac{du_i}{dt} \approx \sigma \cdot \frac{u_{i+1} - 2u_i + u_{i-1}}{\delta x^2} \tag{5.3}$$

y así convertimos la PDE en u en un conjunto de ODEs en u_i.

Como puede verse, la idea principal del MOL es muy sencilla. Lo complicado está en los detalles.

Es razonable utilizar el mismo orden de aproximación para las dos discretizaciones, en el tiempo y en el espacio. Por eso, si queremos integrar el conjunto de IDEs con un método de cuarto orden, deberíamos encontrar una fórmula de discretización para $\partial^2 u/\partial x^2$ que sea de cuarto orden.

Para esto, podemos volver a utilizar los polinomios de Newton–Gregory. Un polinomio de cuarto orden necesita ser ajustado a través de cinco puntos. Preferiremos utilizar *diferencias centradas*, es decir, vamos a ajustar el polinomio a través de los puntos x_{i-2}, x_{i-1}, x_i, x_{i+1}, and x_{i+2}. Al utilizar el polinomio de Newton Gregory hacia atrás, deberemos escribir el mismo en torno al punto ubicado más a la derecha, en nuestro caso, el punto x_{i+2}. Luego, resulta

$$u(x) = u_{i+2} + s\nabla u_{i+2} + \left(\frac{s^2}{2} + \frac{s}{2}\right)\nabla^2 u_{i+2} + \left(\frac{s^3}{6} + \frac{s^2}{2} + \frac{s}{3}\right)\nabla^3 u_{i+2} + \dots \tag{5.4}$$

En consecuencia, la derivada segunda puede escribirse como:

$$\frac{\partial^2 u}{\partial x^2} = \frac{1}{\delta x^2}\left[\nabla^2 u_{i+2} + (s+1)\nabla^3 u_{i+2} + \left(\frac{s^2}{2} + \frac{3s}{2} + \frac{11}{12}\right)\nabla^4 u_{i+2} + \dots\right] \tag{5.5}$$

La Ec.(5.5) debe evaluarse en $x = x_i$, lo que corresponde a $s = -2$. Truncando tras el término de orden cuarto y expandiendo el operador ∇, resulta:

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{x=x_i} \approx \frac{1}{12\delta x^2}\left(-u_{i+2} + 16u_{i+1} - 30u_i + 16u_{i-1} - u_{i-2}\right) \tag{5.6}$$

que es la aproximación en diferencias centrada de cuarto orden de la derivada segunda parcial de $u(x,t)$ respecto de x evaluada en $x = x_i$.

Si hubiéramos querido integrar con un método de segundo orden, deberíamos haber desarrollado el polinomio de Newton Gregory hacia atrás en torno al punto x_{i+1}, truncando la Ec.(5.5) tras el término cuadrático y evaluando en $s = -1$. Esto nos hubiera llevado a:

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{x=x_i} \approx \frac{1}{\delta x^2}\left(u_{i+1} - 2u_i + u_{i-1}\right) \tag{5.7}$$

que es la fórmula de diferencias centrada de segundo orden para $\partial^2 u/\partial x^2$, la misma que utilizamos en Ec.(5.2).

En realidad, puede verse que esta última fórmula es de tercer orden. Por lo tanto, puede utilizarse tanto con métodos de segundo como de tercer orden. De manera similar, la fórmula que deducimos de cuarto orden (5.6) es en realidad de quinto orden. Esto se debe a razones de simetría por los cuales cualquier aproximación en diferencias centradas tiene un orden más que el indicado por el número de puntos utilizado para ajustar el polinomio.

La siguiente dificultad aparece cuando nos aproximamos al borde del dominio espacial. Supongamos que la ecuación del calor corresponde a la distribución de temperatura a lo largo de una barra de longitud $\ell = 1$ m, y supongamos que cortamos dicha barra en 10 segmentos de longitud $\delta\ell = 10$ cm. Si la punta izquierda de la barra corresponde al índice $i = 1$, la punta derecha corresponderá al índice $i = 11$. Supongamos además que queremos integrar mediante un método de orden cuatro. Luego, podremos utilizar la Ec.(5.6) entre los puntos x_3 y x_9. Sin embargo, para los puntos restantes, necesitaremos utilizar un fórmula no centrada, ya que no podemos usar puntos fuera del rango donde la solución $u(x,t)$ está definida.

Para encontrar una fórmula no centrada para x_2, deberemos escribir el polinomio de Newton–Gregory hacia atrás en torno al punto u_5 y evaluarlo para $s = -3$. De manera similar, para obtener una fórmula no centrada para x_1 deberemos escribir el polinomio en torno a u_5 y evaluarlo para $s = -4$. Algo parecido puede hacerse para los puntos x_{10} y x_{11}.

Procediendo de esta manera, obtenemos las siguientes fórmulas no centradas:

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{x=x_1} = \frac{1}{12\delta x^2}\left(11u_5 - 56u_4 + 114u_3 - 104u_2 + 35u_1\right) \tag{5.8a}$$

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{x=x_2} = \frac{1}{12\delta x^2}\left(-u_5 + 4u_4 + 6u_3 - 20u_2 + 11u_1\right) \tag{5.8b}$$

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{x=x_{10}} = \frac{1}{12\delta x^2}\left(11u_{11} - 20u_{10} + 6u_9 + 4u_8 - u_7\right) \tag{5.8c}$$

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{x=x_{11}} = \frac{1}{12\delta x^2}\left(35u_{11} - 104u_{10} + 114u_9 - 56u_8 + 11u_7\right) \tag{5.8d}$$

En el MOL, todas las derivadas con respecto a las variables espaciales se discretizan utilizando aproximaciones en diferencias (centradas o no), mientras que las derivadas con respecto al tiempo se dejan como están. De esta forma, las PDEs se convierten en un conjunto de ODEs que pueden, al menos en teoría, resolverse como cualquier modelo de ODEs mediante métodos de integración numérica estándar.

Tras esto, deberemos discutir que debe hacerse con las condiciones de borde. Cada PDE, además de tener *condiciones iniciales* en el tiempo, tiene *condiciones de borde* (también llamadas condiciones de contorno) en el espacio. Por ejemplo, la ecuación del calor puede tener las siguientes condiciones de borde:

$$u(x = 0.0, t) = 100.0 \tag{5.9a}$$

$$\frac{\partial u}{\partial x}(x = 1.0, t) = 0.0 \tag{5.9b}$$

La Ecuación (5.9a) se denomina *condición de valor de borde*. Este es el caso más simple. Todo lo que debemos hacer es eliminar la ecuación diferencial de $u_1(t)$ y reemplazarla por una ecuación algebraica, en este caso:

$$u_1 = 100.0 \tag{5.10}$$

La condición de borde de la Ec.(5.9b) es también un caso especial. Se denomina *condición de simetría de borde*. Puede tratarse de la siguiente forma. Imaginemos que hay un espejo en $x = 1.0$, que replica la solución $u(x, t)$ en el rango $x \in [1.0, 2.0]$, tal que $u(2.0 - x, t) = u(x, t)$. Obviamente la condición de borde en $x = 2.0$ es la misma que en $x = 0.0$. Luego, no hace falta para nada especificar ninguna condición de borde para $x = 1.0$, ya que por simetría dicha condición de borde será satisfecha. Sabiendo eso, podemos reemplazar las Ecs.(5.8c–d) mediante:

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{x=x_{10}} = \frac{1}{12\delta x^2}\left(-u_{12} + 16u_{11} - 30u_{10} + 16u_9 - u_8\right) \tag{5.11a}$$

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{x=x_{11}} = \frac{1}{12\delta x^2}\left(-u_{13} + 16u_{12} - 30u_{11} + 16u_{10} - u_9\right) \tag{5.11b}$$

o sea, por una aproximación en diferencias centradas.

Debido a la simetría, $u_{12} = u_{10}$ y $u_{13} = u_9$, y podemos reescribir las Ecs.(5.11a–b) como:

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{x=x_{10}} = \frac{1}{12\delta x^2}\left(16u_{11} - 31u_{10} + 16u_9 - u_8\right) \tag{5.12a}$$

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{x=x_{11}} = \frac{1}{12\delta x^2}\left(-30u_{11} + 32u_{10} - 2u_9\right) \tag{5.12b}$$

y ahora podemos olvidarnos de nuestro espejo virtual.

Un tercer tipo especial de condición de borde es la llamada *condición de borde temporal*:

$$\frac{\partial u}{\partial t}(x = 0.0, t) = f(t) \tag{5.13}$$

En este caso, la condición de borde de la PDE es descripta directamente mediante una ODE. Este caso también es simple: directamente reemplazamos la ODE de u_1 por la ODE de borde:

$$\dot{u}_1 = f(t) \tag{5.14}$$

Un condición de borde más general puede tener la forma:

$$g\left(u(x = 1.0, t)\right) + h\left(\frac{\partial u}{\partial x}(x = 1.0, t)\right) = f(t) \tag{5.15}$$

donde f, g, y h son funciones arbitrarias es más complicado. Por ejemplo, podríamos tener una condición:

$$\frac{\partial u}{\partial x}(x = 1.0, t) = -k \cdot \left(u(x = 1.0, t) - u_{\text{amb}}(t)\right) \tag{5.16}$$

donde $u_{\text{amb}}(t)$ es la temperatura ambiente. En este caso, debemos proceder como antes, reemplazando las derivadas espaciales mediante polinomios de Newton–Gregory apropiados:

$$\left.\frac{\partial u}{\partial x}\right|_{x=x_{11}} = \frac{1}{12\delta x}\left(25u_{11} - 48u_{10} + 36u_9 - 16u_8 + 3u_7\right) \tag{5.17}$$

que se trata de la aproximación de diferencias no centradas de cuarto orden. Reemplazando con la Ec.(5.17) en la Ec.(5.16), y resolviendo para u_{11}, encontramos:

$$u_{11} = \frac{12k \cdot \delta x \cdot u_{\text{amb}} + 48u_{10} - 36u_9 + 16u_8 - 3u_7}{12k \cdot \delta x + 25} \tag{5.18}$$

Mediante este proceso, una *condición de borde general* se transformó en una *condición de valor de borde*, y la ODE que define u_{11} puede descartarse.

En algunos casos podemos encontrarnos con *condiciones de borde no lineales*, tales como la condición de radiación:

$$\frac{\partial u}{\partial x}(x = 1.0, t) = -k \cdot \left(u(x = 1.0, t)^4 - u_{\text{amb}}(t)^4\right) \tag{5.19}$$

que conduce a:

$$\begin{aligned}\mathcal{F}(u_{11}) =& 12k \cdot \delta x \cdot u_{11}^4 + 25u_{11} - 12k \cdot \delta x \cdot u_{\text{amb}}^4 - 48u_{10} + 36u_9 \\ & - 16u_8 + 3u_7 = 0.0\end{aligned} \tag{5.20}$$

es decir, una *condición implícita de valor de borde* que puede resolverse mediante la iteración de Newton.

5.3. PDEs Parabólicas

Algunas clases simples de PDEs son tan comunes que reciben nombres especiales. Consideremos por ejemplo la siguiente PDE en dos variables, x e y:

$$a\frac{\partial^2 u}{\partial x^2} + b\frac{\partial^2 u}{\partial x \partial y} + c\frac{\partial^2 u}{\partial y^2} = d \tag{5.21}$$

que es característica de muchos problemas de campos en la física. x e y pueden ser variables espaciales o temporales, y a, b, c, y d pueden ser funciones arbitrarias de x, y, u, $\partial u/\partial x$, and $\partial u/\partial y$. Dicha PDE se denomina *cuasi–lineal*, dado que es lineal en las derivadas superiores.

Dependiendo de la relación numérica entre a, b, y c, la Ec.(5.21) se clasifica como *parabólica*, *hiperbólica*,o *elíptica*. La clasificación es como sigue:

$$b^2 - 4ac > 0 \Longrightarrow \text{PDE es hiperblica} \tag{5.22a}$$

$$b^2 - 4ac = 0 \Longrightarrow \text{PDE es parabólica} \tag{5.22b}$$

$$b^2 - 4ac < 0 \Longrightarrow \text{PDE es elíptica} \tag{5.22c}$$

Las PDEs parabólicas son muy comunes. Por ejemplo, todos los problemas térmicos son de esta naturaleza. El ejemplo más simple de una PDE parabólica es la ecuación de difusión de calor de la Ec.(5.1).

Un ejemplo completo de este problema, incluyendo condiciones iniciales y de borde, es el siguiente:

$$\frac{\partial u}{\partial t} = \frac{1}{10\pi^2} \cdot \frac{\partial^2 u}{\partial x^2} \quad ; \quad x \in [0, 1] \quad ; \quad t \in [0, \infty) \tag{5.23a}$$

$$u(x, t = 0) = \cos(\pi \cdot x) \tag{5.23b}$$

$$u(x = 0, t) = \exp(-t/10) \tag{5.23c}$$

$$\frac{\partial u}{\partial x}(x = 1, t) = 0 \tag{5.23d}$$

Para discretizar este problema mediante el MOL, dividiremos el eje espacial en n segmentos de longitud $\delta x = 1/n$. Utilizaremos la fórmula de diferencias centradas de tercer orden de la Ec.(5.7) para aproximar las derivadas espaciales y utilizaremos el enfoque que presentamos antes para tratar la condición de borde simétrica. De esta forma, obtenemos el conjunto de ODEs:

$$u_1 = \exp(-t/10) \tag{5.24a}$$

$$\dot{u}_2 = \frac{n^2}{10\pi^2} \cdot (u_3 - 2u_2 + u_1) \tag{5.24b}$$

$$\dot{u}_3 = \frac{n^2}{10\pi^2} \cdot (u_4 - 2u_3 + u_2) \tag{5.24c}$$

etc.

$$\dot{u}_n = \frac{n^2}{10\pi^2} \cdot (u_{n+1} - 2u_n + u_{n-1}) \tag{5.24d}$$

$$\dot{u}_{n+1} = \frac{n^2}{5\pi^2} \cdot (-u_{n+1} + u_n) \tag{5.24e}$$

con condiciones iniciales:

$$u_2(0) = \cos\left(\frac{\pi}{n}\right) \tag{5.25a}$$

$$u_3(0) = \cos\left(\frac{2\pi}{n}\right) \tag{5.25b}$$

$$u_4(0) = \cos\left(\frac{3\pi}{n}\right) \tag{5.25c}$$

etc.

$$u_n(0) = \cos\left(\frac{(n-1)\pi}{n}\right) \tag{5.25d}$$

$$u_{n+1}(0) = \cos(\pi) \tag{5.25e}$$

Este es un sistema de la forma:

$$\dot{\mathbf{x}} = \mathbf{A} \cdot \mathbf{x} + \mathbf{b} \cdot u \tag{5.26}$$

donde:

$$\mathbf{A} = \frac{n^2}{10\pi^2} \cdot \begin{pmatrix} -2 & 1 & 0 & 0 & \dots & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & \dots & 0 & 2 & -2 \end{pmatrix} \tag{5.27}$$

$n = 3$	$n = 4$	$n = 5$	$n = 6$	$n = 7$
-0.0244	-0.0247	-0.0248	-0.0249	-0.0249
-0.1824	-0.2002	-0.2088	-0.2137	-0.2166
-0.3403	-0.4483	-0.5066	-0.5407	-0.5621
	-0.6238	-0.9884	-0.9183	-0.9929
		-0.8044	-1.2454	-1.4238
			-1.4342	-1.7693
				-1.9610

Cuadro 5.1: Distribución de autovalores en el modelo de difusión.

A es una matriz con estructura de banda de dimensión $n \times n$. Sus autovalores se encuentran tabulados en la Tabla 5.1.

Todos los autovalores son negativos y reales. Esta es al característica de todos los problemas del dominio térmico y de todas las PDEs parabólicas convertidas en ODEs mediante el MOL.

Puede notarse que mientras que el autovalor más lento permanece casi inalterable, el *cociente de rigidez* (el cociente entre el autovalor más rápido y el más lento, o *stiffness ratio*) aumenta con el número de segmentos.

La Figura 5.1 muestra la gráfica de la raíz cuadrada del cociente de rigidez en función del número de segmentos utilizados.

Figura 5.1: Dependencia del cociente de rigidez con la discretización.

Como vemos, el cociente de rigidez crece en forma cuadrática con el número de segmentos utilizados en la discretización espacial. Cuanto más precisión queremos obtener, la ODE resultante nos queda más stiff.

Debido a las características del sistema de ODEs resultante, resulta que los algoritmos BDF son normalmente los mejores para simularlo.

La PDE que elegimos (5.24) tiene solución analítica conocida:

$$u_c(x,t) = \exp(-t/10) \cdot \cos(\pi \cdot x) \tag{5.28}$$

Luego, podemos comparar esta solución con la solución analítica de la ODE

resultante de aplicar el MOL.

La gráfica de arriba a la izquierda de la Fig.5.2 muestra la solución de la PDE u_c en función del espacio y del tiempo. La gráfica de arriba a la derecha muestra la solución del sistema de ODEs resultante u_d, y la gráfica de abajo a la izquierda muestra la diferencia entre ambos. La gráfica de abajo a la derecha presenta el error máximo en función del número de segmentos utilizado en la discretización.

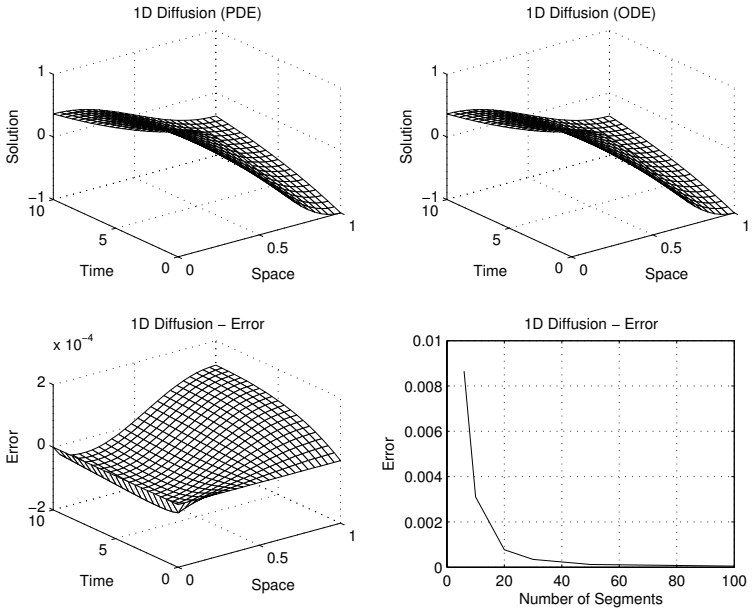

Figura 5.2: Solución del problema de difusión de calor.

Como puede verse, encontramos aquí una nueva clase de error, que llamaremos *error de consistencia*. Este error describe la diferencia entre la PDE original y el conjunto de ODEs resultante de la discretización espacial.

Evidentemente, el error de consistencia no tiene ninguna relación con el método de integración que utilizamos en la integración numérica de las ODEs resultantes.

Podemos disminuir el error de consistencia de dos maneras: aumentando el número de segmentos o incrementando el orden de la aproximación espacial en diferencias. En el primer caso el costo es aumentar la dimensión de las ODEs resultantes y aumentar a la vez el cociente de rigidez.

Veamos que ocurre si aumentamos el orden de la aproximación. Para ver esto, utilizaremos un esquema de quinto orden de diferencias centradas, utilizando diferencias no centradas también de quinto orden para los puntos cercanos a los bordes. Resulta entonces:

$$u_1 = \exp(-t/10) \tag{5.29a}$$

$$\dot{u}_2 = \frac{n^2}{120\pi^2} \cdot (u_6 - 6u_5 + 14u_4 - 4u_3 - 15u_2 + 10u_1) \tag{5.29b}$$

$$\dot{u}_3 = \frac{n^2}{120\pi^2} \cdot (-u_5 + 16u_4 - 30u_3 + 16u_2 - u_1) \tag{5.29c}$$

$$\dot{u}_4 = \frac{n^2}{120\pi^2} \cdot (-u_6 + 16u_5 - 30u_4 + 16u_3 - u_2) \tag{5.29d}$$

$$\text{etc.} nonumber \tag{5.29e}$$

$$\dot{u}_{n-1} = \frac{n^2}{120\pi^2} \cdot (-u_{n+1} + 16u_n - 30u_{n-1} + 16u_{n-2} - u_{n-3}) \tag{5.29f}$$

$$\dot{u}_n = \frac{n^2}{120\pi^2} \cdot (16u_{n+1} - 31u_n + 16u_{n-1} - u_{n-2}) \tag{5.29g}$$

$$\dot{u}_{n+1} = \frac{n^2}{60\pi^2} \cdot (-15u_{n+1} + 16u_n - u_{n-1}) \tag{5.29h}$$

Ahora, la matriz $\mathbf{A}$ resultante tiene la forma:

$$\mathbf{A} = \frac{n^2}{120\pi^2} \cdot \begin{pmatrix} -15 & -4 & 14 & -6 & 1 & \dots & 0 & 0 & 0 & 0 \\ 16 & -30 & 16 & -1 & 0 & \dots & 0 & 0 & 0 & 0 \\ -1 & 16 & -30 & 16 & -1 & \dots & 0 & 0 & 0 & 0 \\ 0 & -1 & 16 & -30 & 16 & \dots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & 16 & -30 & 16 & -1 \\ 0 & 0 & 0 & 0 & 0 & \dots & -1 & 16 & -31 & 16 \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & -2 & 32 & -30 \end{pmatrix} \tag{5.30}$$

Esta matriz sigue teniendo estructura de banda, pero esta banda ahora es más ancha. Los autovalores están tabulados en la Tabla 5.2.

$n = 5$	$n = 6$	$n = 7$	$n = 8$	$n = 9$
-0.0250	-0.0250	-0.0250	-0.0250	-0.0250
-0.2288	-0.2262	-0.2253	-0.2251	-0.2250
-0.5910	-0.6414	-0.6355	-0.6302	-0.6273
-0.7654	-0.9332	-1.1584	-1.2335	-1.2368
-1.2606	-1.3529	-1.4116	-1.6471	-1.9150
	-1.8671	-2.0761	-2.1507	-2.2614
		-2.5770	-2.9084	-3.0571
			-3.3925	-3.8460
				-4.3147

Cuadro 5.2: Distribución de autovalores en el modelo de difusión.

Como vemos, la distribución cambia poco. La Figura 5.3 muestra la raíz cuadrada del cociente de rigidez en función del número de segmentos, y lo compara con lo obtenido para la aproximación de tercer orden.

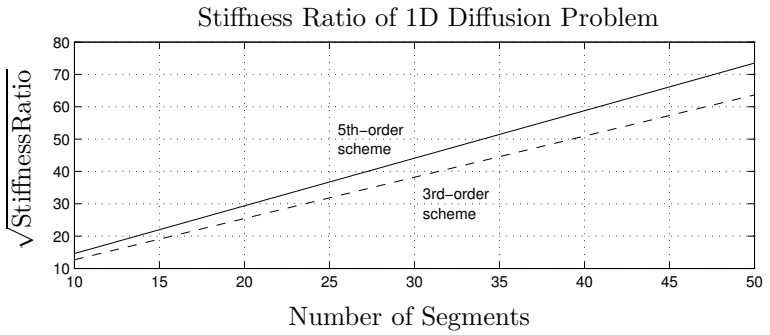

Figura 5.3: Dependencia del cociente de rigidez con la discretización.

Para el mismo número de segmentos, el cociente de rigidez del esquema de quinto orden es algo mayor que el de tercer orden.

En cuanto a la precisión, la Fig.5.4 muestra el error de consistencia en función del número de segmentos utilizados en la discretización para los esquemas de tercer y quinto orden.

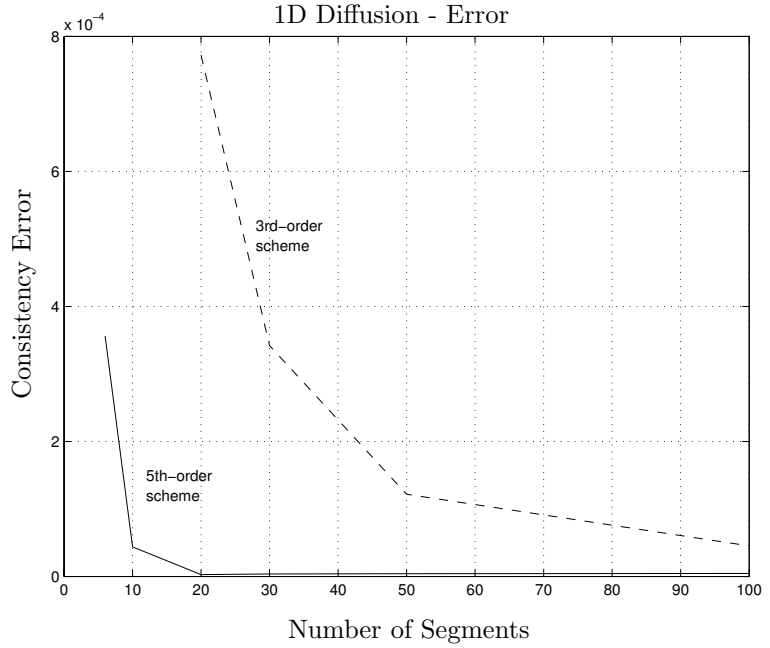

Figura 5.4: Error de consistencia del problema de difusión de calor.

La mejora es muy importante. De todas formas, el costo de simular el sistema de ODEs resultante con una precisión acorde a este error de consistencia será mucho mayor, ya que deberemos utilizar un paso de integración muy pe-

queño.

En síntesis, podemos disminuir el error de consistencia ya sea aumentando el número de segmentos o aumentando el orden de la aproximación, pero de ambas formas aumentaremos el costo computacional.

5.4. PDEs Hiperbólicas

Veamos ahora otra clase de problemas PDE, denominados hiperbólicos. El caso más simple de estos es la *ecuación de ondas* o *ley de conservación lineal*:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \cdot \frac{\partial^2 u}{\partial x^2} \tag{5.31}$$

Podemos transformar fácilmente esta PDE de segundo orden en el tiempo en dos PDEs de primer orden en el tiempo:

$$\frac{\partial u}{\partial t} = v \tag{5.32a}$$

$$\frac{\partial v}{\partial t} = c^2 \cdot \frac{\partial^2 u}{\partial x^2} \tag{5.32b}$$

Ahora, podríamos aproximar las derivadas espaciales mediante diferencias finitas y proceder como antes.

Las Ecuaciones (5.33a–e) constituyen una especificación completa del modelo, incluyendo condiciones iniciales y de borde.

$$\frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2} \quad ; \quad x \in [0,1] \quad ; \quad t \in [0,\infty) \tag{5.33a}$$

$$u(x, t=0) = \sin\left(\frac{\pi}{2}x\right) \tag{5.33b}$$

$$\frac{\partial u}{\partial t}(x, t=0) = 0.0 \tag{5.33c}$$

$$u(x=0, t) = 0.0 \tag{5.33d}$$

$$\frac{\partial u}{\partial x}(x=1, t) = 0.0 \tag{5.33e}$$

Para simular este problema mediante el MOL, dividiremos el eje espacial en n segmentos de ancho $\delta x = 1/n$. Trabajando con la fórmula de diferencias centradas de la Ec.(5.7), y procediendo como antes con la condición de simetría de borde, obtenemos el siguiente conjunto de ODEs:

$$u_1 = 0.0 \tag{5.34a}$$
$$\dot{u}_2 = v_2 \tag{5.34b}$$
etc.
$$\dot{u}_{n+1} = v_{n+1} \tag{5.34c}$$
$$v_1 = 0.0 \tag{5.34d}$$
$$\dot{v}_2 = n^2 \left(u_3 - 2u_2 + u_1\right) \tag{5.34e}$$
$$\dot{v}_3 = n^2 \left(u_4 - 2u_3 + u_2\right) \tag{5.34f}$$
etc.
$$\dot{v}_n = n^2 \left(u_{n+1} - 2u_n + u_{n-1}\right) \tag{5.34g}$$
$$\dot{v}_{n+1} = 2n^2 \left(u_n - u_{n+1}\right) \tag{5.34h}$$

con condiciones iniciales:

$$u_2(0) = \sin\left(\frac{\pi}{2n}\right) \tag{5.35a}$$
$$u_3(0) = \sin\left(\frac{\pi}{n}\right) \tag{5.35b}$$
$$u_4(0) = \sin\left(\frac{3\pi}{2n}\right) \tag{5.35c}$$
etc.
$$u_n(0) = \sin\left(\frac{(n-1)\pi}{2n}\right) \tag{5.35d}$$
$$u_{n+1}(0) = \sin\left(\frac{\pi}{2}\right) \tag{5.35e}$$
$$v_2(0) = 0.0 \tag{5.35f}$$
etc.
$$v_{n+1}(0) = 0.0 \tag{5.35g}$$

Este sistema tiene la forma de la Ec.(5.26), donde:

$$\mathbf{A} = \begin{pmatrix} \mathbf{0^{(n)}} & \mathbf{I^{(n)}} \\ \mathbf{A_{21}} & \mathbf{0^{(n)}} \end{pmatrix} \tag{5.36}$$

donde:

$$\mathbf{A_{21}} = n^2 \begin{pmatrix} -2 & 1 & 0 & 0 & \dots & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & \dots & 0 & 2 & -2 \end{pmatrix} \tag{5.37}$$

A es una matriz con estructura de bandas de dimensión $2n \times 2n$ con dos bandas separadas no nulas. Los autovalores están tabulados en la Tabla 5.3.

$n = 3$	$n = 4$	$n = 5$	$n = 6$
$\pm 1.5529j$	$\pm 1.5607j$	$\pm 1.5643j$	$\pm 1.5663j$
$\pm 4.2426j$	$\pm 4.4446j$	$\pm 4.5399j$	$\pm 4.5922j$
$\pm 5.7956j$	$\pm 6.6518j$	$\pm 7.0711j$	$\pm 7.3051j$
	$\pm 7.8463j$	$\pm 8.9101j$	$\pm 9.5202j$
		$\pm 9.8769j$	$\pm 11.0866j$
			$\pm 11.8973j$

Cuadro 5.3: Distribución de autovalores de la ley de conservación lineal.

Todos los autovalores son estrictamente imaginarios. Las PDEs hiperbólicas convertidas en conjuntos de ODEs utilizando el MOL tienen siempre autovalores complejos. Muchas de ellas los tienen muy cerca del eje imaginario. En el caso de la ley de conservación lineal, los autovalores quedan siempre sobre el eje imaginario.

La Figura 5.5 muestra el *cociente de frecuencia*, es decir, el cociente entre el mayor y el menor valor absoluto de la parte imaginaria de los autovalores en función del número de segmentos utilizados.

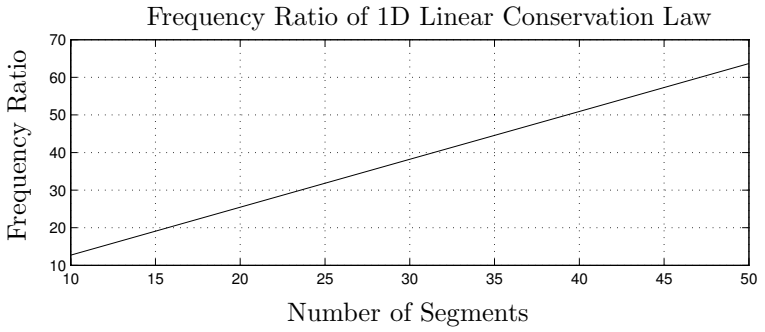

Figura 5.5: Cociente de frecuencia de la ley de conservación lineal.

Evidentemente, el cociente de frecuencia crece linealmente con el número de segmentos utilizados.

El problema numérico es ahora bastante distinto que el del caso parabólico. Ahora no tenemos un problema stiff, y no hay transitorios rápidos que desaparezcan y que permitan aumentar el paso de integración, sino que deberemos utilizar todo el tiempo un paso de integración chico. Cuanto más angosta sea la grilla utilizada, más chico será el paso de integración a utilizar.

Este nuevo ejemplo es muy simple y también se conoce la solución analítica:

$$u(x,t) = \frac{1}{2}\sin\left(\frac{\pi}{2}(x-t)\right) + \frac{1}{2}\sin\left(\frac{\pi}{2}(x+t)\right) \tag{5.38}$$

Procediendo como antes, podemos comparar esta solución con la solución analítica del conjunto de ODEs obtenido y calcular el error de consistencia.

La Figura 5.6 muestra en la gráfica superior izquierda la solución analítica de la PDE original. Arriba a la derecha se grafica la solución de la ODE resultante, abajo a la izquierda el error y por último, abajo a la derecha, el error máximo en función del número de segmentos.

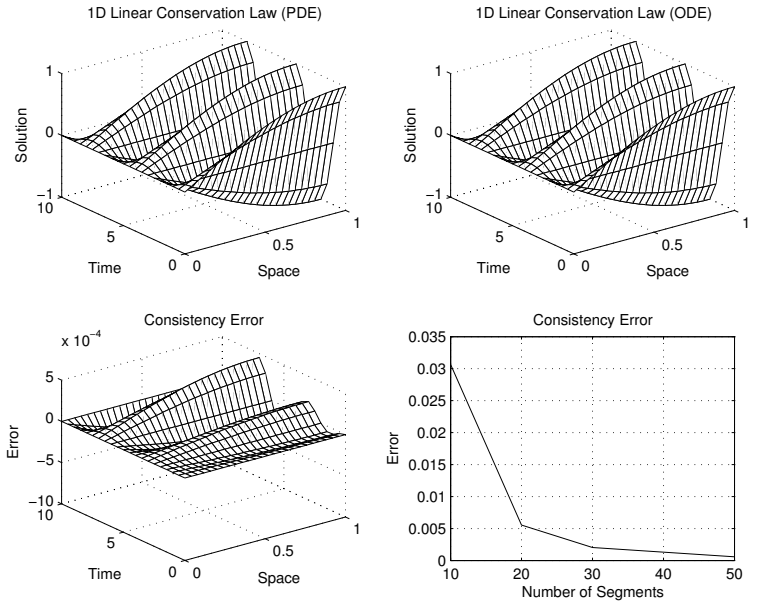

Figura 5.6: Soluciones analíticas de la ecuación de ondas 1D.

El error de consistencia es mucho mayor que en los casos parabólicos anteriores. Para obtener una precisión numérica del 1 %, el error de consistencia debería ser no mayor que el 0.1 %, lo que requiere de utilizar al menos 40 segmentos.

Como antes, podemos utilizar una discretización de orden mayor en el espacio.

Para el caso de quinto orden, la Figura 5.7 muestra el cociente de frecuencias en función del número de segmentos

Como vemos, este es algo mayor que el cociente de frecuencias en la aproximación de tercer orden.

La Figura 5.8 muestra el error de consistencia en función del número de segmentos utilizados en la discretización. Como puede verse, la mejora es muy importante.

En el caso de las PDEs parabólicas, dijimos que los métodos de BDF eran los más apropiados. En el caso hiperbólico sin embargo, nos deberemos inclinar más hacia los métodos F–estables de BI.

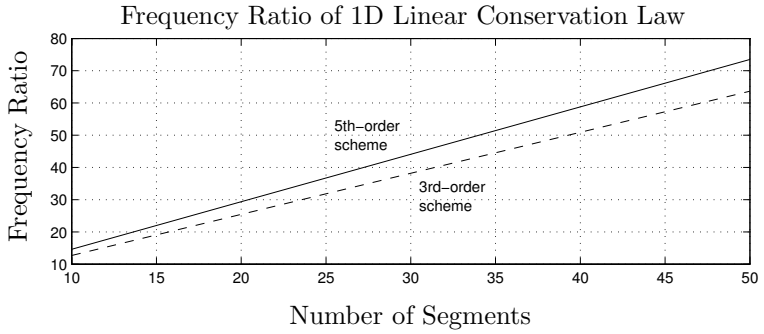

Figura 5.7: Cociente de frecuencias de la ecuación de ondas.

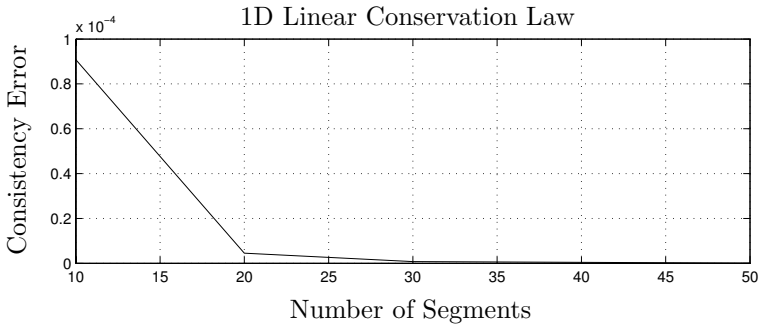

Figura 5.8: Error de consistencia de la ecuación de ondas.

5.5. Problemas Propuestos

[P5.1] Difusión de Calor en el Suelo Los Ingenieros Agrónomos suelen interesarse en conocer la distribución de temperatura en el suelo como función de la temperatura del aire. Como se muestra en la Fig.5.9, asumiremos que la capa del suela es de 50 cm. Bajo el suelo, hay una capa que actúa como un aislante térmico ideal.

El problema del flujo de calor puede escribirse como:

$$\frac{\partial u}{\partial t} = \frac{\lambda}{\rho \cdot c} \cdot \frac{\partial^2 u}{\partial x^2} \tag{P5.1a}$$

donde $\lambda = 0.004$ cal cm^{-1} sec^{-1} K^{-1} es la conductividad térmica específica del suelo, $\rho = 1.335$ g cm^{-3} es la densidad del suelo, y $c = 0.2$ cal g^{-1} K^{-1} es la capacidad térmica específica del suelo.

La temperatura del aire en la superficie fue registrada cada 6 horas y las mediciones en función del tiempo están tabuladas en la Tabla 5.4.

Asumiremos que la temperatura de la superficie del suelo es idéntica a la temperatura del aire de la superficie en todo momento. Más aún, supondremos

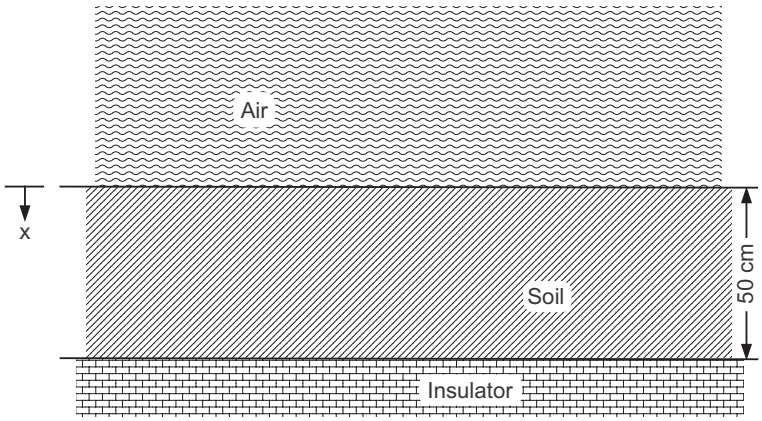

Figura 5.9: Topología del suelo.

t [horas]	u °C
0	6
6	16
12	28
18	21
24	18
36	34
48	18
60	25
66	15
72	4

Cuadro 5.4: Temperatura del aire en la superficie.

que la temperatura inicial del suelo en todos lados es igual a la temperatura inicial de la superficie.

Especificar este problemas utilizando horas como unidad de tiempo y centímetros como unidad espacial. Discretizar el problema mediante diferencias finitas de tercer orden y simular la ODE lineal resultante.

Graficar superpuesta la temperatura en la superficie y en el aislante en función del tiempo. Generar también una gráfica tridimensional que muestre la distribución de la temperatura en el suelo en función del tiempo y el espacio

[P5.2] Calentamiento Eléctrico de una Barra

Consideraremos una ecuación diferencial parcial que describe la distribución de la temperatura en una barra de cobre calentada de forma eléctrica. Este

fenómeno puede modelarse por la siguiente ecuación:

$$\frac{\partial T}{\partial t} = \sigma \left(\frac{\partial^2 T}{\partial r^2} + \frac{1}{r} \cdot \frac{\partial T}{\partial r} + \frac{P_{\text{electr}}}{\lambda \cdot V} \right) \tag{P5.2a}$$

Los dos primeros términos representan la ecuación estándar de difusión en coordinadas polares, y el término constante describe el calor generado por la electricidad.

$$\sigma = \frac{\lambda}{\rho \cdot c} \tag{P5.2b}$$

es el coeficiente de difusión, donde $\lambda = 401.0$ J m^{-1} sec^{-1} K^{-1} es la conductividad térmica específica del cobre, $\rho = 8960.0$ kg m^{-3} es su densidad y $c = 386.0$ J kg^{-1} K^{-1} es la capacidad térmica específica.

$$P_{\text{electr}} = u \cdot i \tag{P5.2c}$$

es la potencia eléctrica disipada, que supondremos constante $P_{\text{electr}} = 500$ W, y

$$V = \pi \cdot R^2 \cdot \ell \tag{P5.2d}$$

es el volumen de la barra de longitud $\ell = 1$ m y radio $R = 0.01$ m. La barra está originalmente en estado de equilibrio a la temperatura ambiente $T_{\text{room}} = 298.0$ K.

Las condiciones de borde son:

$$\left.\frac{\partial T}{\partial r}\right|_{r=0.0} = 0.0 \tag{P5.2e}$$

$$\left.\frac{\partial T}{\partial r}\right|_{r=R} = -k_1 \left(T(R)^4 - T_{\text{room}}^4 \right) - k_2 \left(T(R) - T_{\text{room}} \right) \tag{P5.2f}$$

donde el término de orden cuatro modela la radiación de calor, mientras el término lineal representa el flujo de convección que sale de la barra.

Consideraremos los valores $k_1 = 5.344 \times 10^{-11}$ m^{-1}K^{-3} y $k_2 = 3.1416$ m^{-1}.

Queremos simular este sistema mediante el MOL con 20 segmentos espaciales (en dirección radial), y utilizando diferencias finitas de segundo orden para las derivadas espaciales de primer orden y diferencias de tercer ordan para las de segundo orden.

Para esto, trataremos las condiciones de borde del centro de la barra como una condición de borde general y no como una condición de simetría para evitar las dificultades al computar el término $(\partial T/\partial r)/r$, que da como resultado $0/0$ en $r = 0.0$.

Para los segmentos internos, obtenemos ecuaciones diferenciales de la forma:

$$\frac{dT_i}{dt} \approx \sigma \left(\frac{T_{i+1} - 2T_i + T_{i-1}}{\delta r^2} + \frac{1}{r} \cdot \frac{T_{i+1} - T_{i-1}}{2\delta r} + \frac{P_{\text{electr}}}{\lambda \cdot V} \right) \tag{P5.2g}$$

que son triviales de implementar. Para el segmento de la izquierda, tenemos la condición:

$$\left.\frac{\partial T}{\partial r}\right|_{r=0.0} = 0.0 \approx \frac{1}{2\delta r}\left(-T_3 + 4T_2 - 3T_1\right) \tag{P5.2h}$$

y luego:

$$T_1 \approx \frac{4}{3}T_2 - \frac{1}{3}T_3 \tag{P5.2i}$$

En consecuencia, no hace falta resolver una ecuación diferencial para $r = 0.0$, y evitamos la división $0/0$.

En el segmento de la derecha, obtenemos:

$$\left.\frac{\partial T}{\partial r}\right|_{r=R} = -k_1\left(T_{21}^4 - T_{\text{room}}^4\right) - k_2\left(T_{21} - T_{\text{room}}\right) \approx \frac{1}{2\delta r}\left(3T_{21} - 4T_{20} + T_{19}\right) \tag{P5.2j}$$

Luego, tenemos una ecaución no lineal en T_{21}:

$$\begin{aligned}\mathcal{F}(T_{21}) &= k_1\left(T_{21}^4 - T_{\text{room}}^4\right) + k_2\left(T_{21} - T_{\text{room}}\right) + \frac{1}{2\delta r}\left(3T_{21} - 4T_{20} + T_{19}\right) \\ &\approx 0.0\end{aligned} \tag{P5.2k}$$

que puede resolverse mediante la iteración de Newton:

$$T_{21}^0(t) = T_{21}(t-h) \tag{P5.2l}$$

$$T_{21}^1(t) = T_{21}^0(t) - \frac{\mathcal{F}(T_{21}^0)}{\mathcal{H}(T_{21}^0)} \tag{P5.2m}$$

$$T_{21}^2(t) = T_{21}^1(t) - \frac{\mathcal{F}(T_{21}^1)}{\mathcal{H}(T_{21}^1)} \tag{P5.2n}$$

hasta convergencia

donde:

$$\mathcal{H}(T_{21}) = \frac{\partial \mathcal{F}}{\partial T_{21}} = 4k_1T_{21}^3 + k_2 - \frac{3}{2\delta r} \tag{P5.2ñ}$$

Se pide entonces simular el sistema descripto durante 50 seg utilizando el método de BDF4.

[P5.3] Simulación del Lecho de un Río Un problema muy común en la ingeniería hidráulica es determinar el movimiento de los lechos de los ríos con el tiempo. La dinámica de estos sistemas puede describirse mediante la PDE:

$$\frac{\partial v}{\partial t} + v \cdot \frac{\partial v}{\partial x} + g \cdot \frac{\partial h}{\partial x} + g \cdot \frac{\partial z}{\partial x} = w(v) \tag{P5.3a}$$

donde $v(x,t)$ es el valor absoluto de la velocidad del flujo de agua, $h(x,t)$ es la profundidad del agua, y $z(x,t)$ es la altura relativa del lecho del río relativo a un nivel constante. $g = 9.81$ m sec^{-2} es la constante de la gravedad.

$w(v)$ es la fricción del agua sobre el lecho del río:

$$w(v) = -\frac{g \cdot v^2}{s_k^2 \cdot h^{4/3}} \tag{P5.3b}$$

donde $s_k = 32.0$ m$^{3/4}$ sec^{-1} es la constante de Strickler.

La ecuación de continuidad para el agua puede escribirse como:

$$\frac{\partial h}{\partial t} + v \cdot \frac{\partial h}{\partial x} + h \cdot \frac{\partial v}{\partial x} = 0.0 \tag{P5.3c}$$

y la ecuación de continuidad para el lecho del río puede expresarse como:

$$\frac{\partial z}{\partial t} + \frac{df(v)}{dv} \cdot \frac{\partial v}{\partial x} = 0.0 \tag{P5.3d}$$

donde $f(v)$ es la ecuación de transporte de Meyer–Peter simplificada mediate análisis de regresión:

$$f(v) = f_0 + c_1(v - v_0)^{c_2} \tag{P5.3e}$$

donde $c_1 = 1.272 \cdot 10^{-4}$ m, y $c_2 = 3.5$.

Queremos estudiar el río Rin a la altura de Basilea sobre una distancia de 6.3 km. Simularemos este sistemas durante 20 días, con las condiciones iniciales tabuladas en la Tabla 5.5.

x [m]	v [m/s]	h [m]	z [m]
0.0	2.3630	3.039	59.82
630.0	2.3360	3.073	59.06
1260.0	2.2570	3.181	58.29
1890.0	1.6480	4.357	56.75
2520.0	1.1330	6.337	54.74
3150.0	1.1190	6.416	54.60
3780.0	1.1030	6.509	54.45
4410.0	1.0620	7.001	53.91
5040.0	0.8412	8.536	52.36
5670.0	0.7515	9.554	51.33
6300.0	0.8131	8.830	52.02

Cuadro 5.5: Datos iniciales para la simulación del lecho del rio.

Necesitamos además tres condiciones de borde. Asumiremos que la cantidad de agua amount of water $q = h \cdot v$ que entra en el segmento de rio que estamos simulando es constante. Por simplicidad, asumiremos que h y v son constantes. En el final del segmento, hay una presa. Por lo tanto, podemos asumir que la suma de z y h es constante en la parte mas baja del segmento de río simulado. Dado que el agua se mueve mucho más rápido que el lecho, no tiene mucho sentido aplicar condiciones de borde al lecho.

Este sistema es bastante complicado. Las constantes de tiempo del agua se miden en segundos, mientras que las de la tierra se miden en dias. Si bien

estamos interesados en las constantes de tiempo lentas, son las rápidas las que dictaminan el paso de integración.

Podemos pensar que las dos primeras PDEs generan una función no lineal para la tercera. Modificaremos entonces la Ec.(P5.3d) como sigue:

$$\frac{\partial z}{\partial t} + \beta \cdot \frac{df(v)}{dv} \cdot \frac{\partial v}{\partial x} = 0.0 \tag{P5.3f}$$

Cuanto más grande seleccionamos el parámetro de ajuste, mas rápido se moverá el lecho del rio. Seleccionaremos un valor alrededor de $\beta = 100$ o incluso $\beta = 1000$.

Luego, deberemos analizar el daño que hicimos a la PDE al introducir este parámetro de ajuste. Quizás, podremos extrapolar hacia el comportamiento correcto del sistema en $\beta = 1.0$.

La tercera condición de borde es analíticamente correcta, pero numéricamente no muy efectiva. Podemos entonces eliminar esta condición e introducir otra condición de borde para el tramo inicial:

$$\frac{\partial z}{\partial x} = constante \tag{P5.3g}$$

Sin embargo, dado que no podemos especificar una condición de derivada de borde para una ecuación de primer orden, reformulamos la Ec.(P5.3g) según:

$$z_1 = z_2 + constante \tag{P5.3h}$$

Dibujar la altura del lecho del rio $z(x)$ medida al final de cada período de 5 días superpuesto en una misma gráfica.

Correr luego la simulación para diferentes valores de β. ¿Puede extrapolarse y obtener una aproximación de la solución para $\beta = 1.0$?

Capítulo 6

Integración de Ecuaciones Diferenciales Algebraicas

Este capítulo es principalmente una traducción resumida del capítulo 8 del libro *Continuous System Simulation* [2]. Sin embargo, en este caso hay varios cambios, sobre todo en la introducción.

6.1. Introducción

En el problema [P5.2], tras aplicar el MOL a una Ecuación en Derivadas Parciales, nos encontramos con la siguiente Ecuación Diferencial Algebraica (DAE):

$$\frac{dT_i}{dt} \approx \sigma \left(\frac{T_{i+1} - 2T_i + T_{i-1}}{\delta r^2} + \frac{1}{r} \cdot \frac{T_{i+1} - T_{i-1}}{2\delta r} + \frac{P_{\text{electr}}}{\lambda \cdot V} \right) \tag{6.1}$$

para $i = 2, \ldots, 20$, donde además teníamos

$$T_1 = \frac{4}{3} T_2 - \frac{1}{3} T_3 \tag{6.2}$$

y

$$\begin{aligned} \mathcal{F}(T_{21}) &= k_1 \left(T_{21}^4 - T_{\text{room}}^4 \right) + k_2 \left(T_{21} - T_{\text{room}} \right) + \frac{1}{2\delta r} \left(3T_{21} - 4T_{20} + T_{19} \right) \\ &\approx 0.0 \end{aligned} \tag{6.3}$$

Para simular este problema, lo que hicimos fue implementar una iteración de Newton que encuentre T_{21} de esta última ecuación y luego utilice este valor para calcular las derivadas $\frac{dT_i}{dt}$. Es decir, cada vez que necesitábamos evaluar estas derivadas, debíamos primero invocar la iteración de Newton para calcular T_{21}.

Además, como el problema era stiff (provenía de una ecuación parabólica), debíamos utilizar un método implícito (en el problema sugerimos utilizar BDF4). Por lo tanto, necesitábamos una segunda iteración de Newton sobre muchas

variables en cada paso. Si además hubiéramos utilizado algún tipo de control de paso, habríamos tenido también que rechazar algunos pasos al achicar el paso de integración, lo que constituiría un tercer proceso iterativo. Estos tres procesos iterativos anidadeos se ilustran en la Fig.6.1.

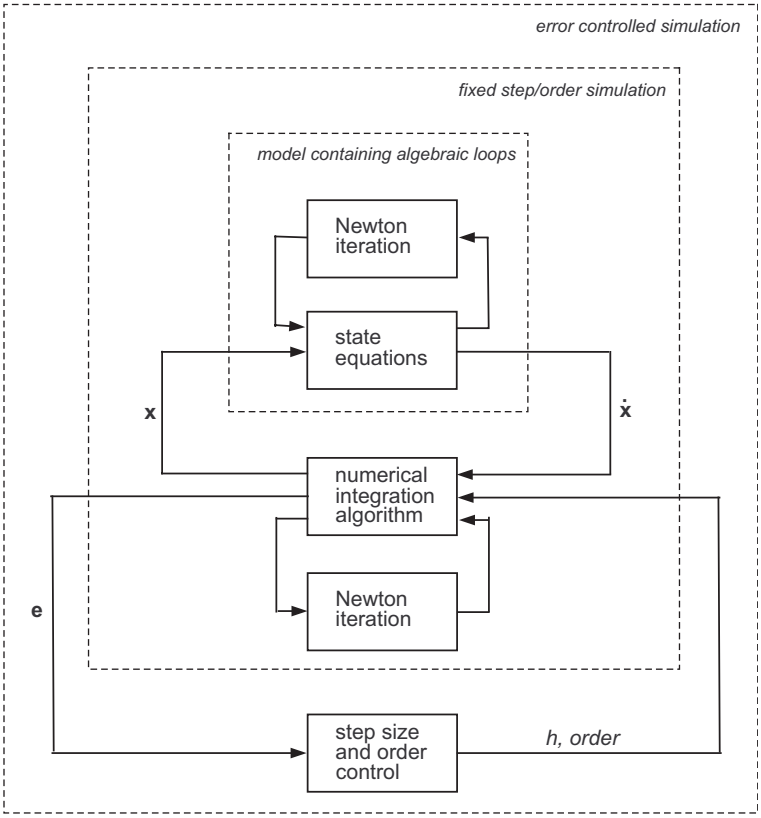

Figura 6.1: Los tres lazos de la simulación.

Ahora bien, ¿es realmente necesario realizar todos estos procesos iterativos?.

La causa de la primer iteración (para obtener T_{21}) es la necesidad de calcular explícitamente las derivadas. Veamos que ocurre si trabajamos directamente sobre una versión implícita del modelo.

En nuestro caso, el modelo tiene la forma:

$$\mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}, t) = 0.0 \tag{6.4}$$

donde $\mathbf{x} = [T_2, \ldots, T_{21}]^T$ (notar que T_{21} no es realmente una variable de estados, y no podemos asignarle una condición inicial arbitraria, sino que debe ser *consistente*). La función $\mathbf{f}$ puede deducirse fácilmente a partir de las Ecs.(6.1)–(6.3).

Si decidimos utilizar el método de Backward Euler:

$$\mathbf{x_{k+1}} = \mathbf{x_k} + h \cdot \dot{\mathbf{x}}_{\mathbf{k+1}} \tag{6.5}$$

podemos primero despejar $\dot{\mathbf{x}}_{\mathbf{k+1}}$:

$$\dot{\mathbf{x}}_{\mathbf{k+1}} = \frac{1}{h}\left[\mathbf{x_{k+1}} - \mathbf{x_k}\right] \tag{6.6}$$

y reemplazar con esta última ecuación en la Ec.(6.4):

$$\mathbf{f}(\mathbf{x_{k+1}}, \frac{1}{h}\left[\mathbf{x_{k+1}} - \mathbf{x_k}\right], t) = 0.0 \tag{6.7}$$

obteniendo así una única ecuación vectorial no lineal con el vector incógnita $\mathbf{x_{k+1}}$, de la forma:

$$\mathcal{F}(\mathbf{x_{k+1}}) = 0.0 \tag{6.8}$$

que puede resolverse directamente mediante la iteración de Newton:

$$\mathbf{x}_{\mathbf{k+1}}^{\boldsymbol{\ell+1}} = \mathbf{x}_{\mathbf{k+1}}^{\boldsymbol{\ell}} - (\mathcal{H}^{\ell})^{-1} \cdot \mathcal{F}^{\ell} \tag{6.9}$$

donde:

$$\mathcal{H} = \frac{\partial \mathcal{F}}{\partial \mathbf{x_{k+1}}} \tag{6.10}$$

De esta manera, pudimos reducir el problema a una única iteración que incluye las iteraciones del método implícito y las iteraciones propias de la definición implícita del modelo. Esta idea, que fue propuesta por primera vez por Bill Gear en 1971, es la que vamos a analizar en más detalle en el resto del capítulo.

6.2. Métodos Monopaso

En principio, la formulación DAE de todos los métodos monopaso es trivial, y no difiere de lo hecho anteriormente para el método de Backward Euler. Por ejemplo, una formulación DAE del método explícito RK4 puede implementarse de la siguiente forma:

$$\begin{aligned}
&\mathbf{f}\left(\mathbf{x_k}, \mathbf{k_1}, t_k\right) = 0.0 \\
&\mathbf{f}\left(\mathbf{x_k} + \frac{h}{2}\mathbf{k_1}, \mathbf{k_2}, t_k + \frac{h}{2}\right) = 0.0 \\
&\mathbf{f}\left(\mathbf{x_k} + \frac{h}{2}\mathbf{k_2}, \mathbf{k_3}, t_k + \frac{h}{2}\right) = 0.0 \\
&\mathbf{f}\left(\mathbf{x_k} + h\mathbf{k_3}, \mathbf{k_4}, t_k + h\right) = 0.0 \\
&\mathbf{x_{k+1}} = \mathbf{x_k} + \frac{h}{6}\left(\mathbf{k_1} + 2\mathbf{k_2} + 2\mathbf{k_3} + \mathbf{k_4}\right)
\end{aligned}$$

Esta es exáctamente la misma fórmula que utilizamos en el Capítulo 3, salvo que ahora la evaluación de los términos $\mathbf{k_i}$ requiere de iteraciones de Newton.

En general, esto es demasiado costoso, y su uso no se justifica. Algo similar ocurre si utilizamos métodos tipo BI, o tipo BRK. La cantidad de evaluaciones de la función a causa de la iteración se hace innecesariamente grande.

Una familia de métodos que son particularmente aptos para DAEs son los denominados *algoritmos de Runge–Kutta completamente implícitos*. Veamos por ejemplo los métodos denominados *Radau IIA*, que pueden representarse por las siguientes tabla de Butcher:

1/3	5/12	-1/12
1	3/4	1/4
x	3/4	1/4

$\frac{4-\sqrt{6}}{10}$	$\frac{88-7\sqrt{6}}{360}$	$\frac{296-169\sqrt{6}}{1800}$	$\frac{-2+3\sqrt{6}}{225}$
$\frac{4+\sqrt{6}}{10}$	$\frac{296+169\sqrt{6}}{1800}$	$\frac{88+7\sqrt{6}}{360}$	$\frac{-2-3\sqrt{6}}{225}$
1	$\frac{16-\sqrt{6}}{36}$	$\frac{16+\sqrt{6}}{36}$	$\frac{1}{9}$
x	$\frac{16-\sqrt{6}}{36}$	$\frac{16+\sqrt{6}}{36}$	$\frac{1}{9}$

El primer método es un método de Runge–Kutta completamente implícito de tercer orden en dos etapas, mientras que el segundo es de quinto orden en tres etapas.

El método de tercer orden puede interpretarse, en el caso de una ODE explícita, de la siguiente forma:

$$\mathbf{k_1} = \mathbf{f}\left(\mathbf{x_k} + \frac{5h}{12}\mathbf{k_1} - \frac{h}{12}\mathbf{k_2}, \mathbf{u}(t_k + \frac{h}{3}), t_k + \frac{h}{3}\right) \tag{6.11a}$$

$$\mathbf{k_2} = \mathbf{f}\left(\mathbf{x_k} + \frac{3h}{4}\mathbf{k_1} + \frac{h}{4}\mathbf{k_2}, \mathbf{u}(t_k + h), t_k + h\right) \tag{6.11b}$$

$$\mathbf{x_{k+1}} = \mathbf{x_k} + \frac{h}{4}\left(3\mathbf{k_1} + \mathbf{k_2}\right) \tag{6.11c}$$

En el caso DAE, el método queda:

$$\mathbf{f}\left(\mathbf{x_k} + \frac{5h}{12}\mathbf{k_1} - \frac{h}{12}\mathbf{k_2}, \mathbf{k_1}, t_k + \frac{h}{3}\right) = 0.0 \tag{6.12a}$$

$$\mathbf{f}\left(\mathbf{x_k} + \frac{3h}{4}\mathbf{k_1} + \frac{h}{4}\mathbf{k_2}, \mathbf{k_2}, t_k + h\right) = 0.0 \tag{6.12b}$$

$$\mathbf{x_{k+1}} = \mathbf{x_k} + \frac{h}{4}\left(3\mathbf{k_1} + \mathbf{k_2}\right) \tag{6.12c}$$

En este caso, tenemos que iterar para obtener las incógnitas: $\mathbf{k_1}$ y $\mathbf{k_2}$. Es decir, tenemos que resolver simultáneamente un conjunto de $2n$ ecuaciones no lineales con $2n$ incógnitas.

Como veremos luego, los métodos de Radau IIA son stiff estables. La ventaja es que el costo computacional que tienen es prácticamente el mismo para el caso ODE que para el caso DAE.

Una consideración importante es que para iterar y obtener $\mathbf{k_i}$, necesitamos comenzar la iteración desde un valor inicial cercano al de $\mathbf{k_i}$. Como los $\mathbf{k_i}$ evalúan la derivada $\dot{\mathbf{x}}$, lo que en realidad necesitamos es estimar la derivada inicial, y comenzar las iteraciones desde la condición inicial:

$$\mathbf{k_1} = \mathbf{k_2} = \dot{\mathbf{x}}(t_0) \tag{6.13}$$

Una forma de estimar $\dot{\mathbf{x}}(t_0)$ es dando un paso con la versión implícita de Backward Euler descripta en la introducción, y calcular $\dot{\mathbf{x}}(t_0) \approx (\mathbf{x_1} - \mathbf{x_0})/h$.

En cuanto a la región de estabilidad de los métodos de Radau IIA, es importante mencionar que la misma no depende de que la formulación sea implícita o explícita. Por lo tanto, podemos analizar directamente el caso ODE dado por las Ecs.(6.11a–c), para la ODE lineal:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} \tag{6.14}$$

de donde obtenemos:

$$\mathbf{k_1} = \mathbf{A}\left(\mathbf{x_k} + \frac{5h}{12}\mathbf{k_1} - \frac{h}{12}\mathbf{k_2}\right) \tag{6.15a}$$

$$\mathbf{k_2} = \mathbf{A}\left(\mathbf{x_k} + \frac{3h}{4}\mathbf{k_1} + \frac{h}{4}\mathbf{k_2}\right) \tag{6.15b}$$

$$\mathbf{x_{k+1}} = \mathbf{x_k} + \frac{h}{4}\left(3\mathbf{k_1} + \mathbf{k_2}\right) \tag{6.15c}$$

y despejando las incóginatas$\mathbf{k_1}$ y $\mathbf{k_2}$:

$$\mathbf{k_1} = \left[\mathbf{I^{(n)}} - \frac{2\mathbf{A}h}{3} + \frac{(\mathbf{A}h)^2}{6}\right]^{-1} \cdot \left(\mathbf{I^{(n)}} - \frac{\mathbf{A}h}{3}\right) \cdot \mathbf{A} \cdot \mathbf{x_k} \tag{6.16a}$$

$$\mathbf{k_2} = \left[\mathbf{I^{(n)}} - \frac{2\mathbf{A}h}{3} + \frac{(\mathbf{A}h)^2}{6}\right]^{-1} \cdot \left(\mathbf{I^{(n)}} + \frac{\mathbf{A}h}{3}\right) \cdot \mathbf{A} \cdot \mathbf{x_k} \tag{6.16b}$$

$$\mathbf{x_{k+1}} = \mathbf{x_k} + \frac{h}{4}\left(3\mathbf{k_1} + \mathbf{k_2}\right) \tag{6.16c}$$

de donde:

$$\mathbf{F} = \mathbf{I^{(n)}} + \left[\mathbf{I^{(n)}} - \frac{2\mathbf{A}h}{3} + \frac{(\mathbf{A}h)^2}{6}\right]^{-1} \cdot \left(\mathbf{I^{(n)}} - \frac{\mathbf{A}h}{6}\right) \cdot (\mathbf{A}h) \tag{6.17}$$

Desarrollando el denominador en series de Taylor en torno a $h = 0.0$, queda:

$$\mathbf{F} \approx \mathbf{I^{(n)}} + \mathbf{A}h + \frac{(\mathbf{A}h)^2}{2} + \frac{(\mathbf{A}h)^3}{6} + \frac{(\mathbf{A}h)^4}{36} \tag{6.18}$$

Es evidente que el método tiene una precisión de tercer orden (al menos para el caso lineal), y el coeficiente de error es:

$$\varepsilon = \frac{1}{72}(\mathbf{A}h)^4 \tag{6.19}$$

Procediendo de manera similar con el método de Radau IIA de 5to orden, obtenemos la matriz $\mathbf{F}$:

$$\mathbf{F} = \mathbf{I^{(n)}} + \left[\mathbf{I^{(n)}} - \frac{3\mathbf{A}h}{5} + \frac{3(\mathbf{A}h)^2}{20} - \frac{(\mathbf{A}h)^3}{60}\right]^{-1} \left(\mathbf{I^{(n)}} - \frac{\mathbf{A}h}{10} + \frac{(\mathbf{A}h)^2}{60}\right) \mathbf{A}h \tag{6.20}$$

Desarrollando el denominador en serie de Taylor en torno a $h = 0.0$, resulta:

$$\mathbf{F} \approx \mathbf{I^{(n)}} + \mathbf{A}h + \frac{(\mathbf{A}h)^2}{2} + \frac{(\mathbf{A}h)^3}{6} + \frac{(\mathbf{A}h)^4}{24} + \frac{(\mathbf{A}h)^5}{120} + \frac{11(\mathbf{A}h)^6}{7200} \tag{6.21}$$

Luego, el método es efectivamente de quinto orden (al menos para casos lineales) y el coeficiente de error es:

$$\varepsilon = \frac{1}{7200}(\mathbf{A}h)^6 \tag{6.22}$$

Otro método que se utiliza frecuentemente es el algoritmo de *Lobatto IIIC*, un método de RK completamente implícito de 4to orden en tres etapas, con tabla de Butcher:

0	1/6	-1/3	1/6
1/2	1/6	5/12	-1/12
1	1/6	2/3	1/6
x	1/6	2/3	1/6

Este método se caracteriza por la matriz $\mathbf{F}$:

$$\mathbf{F} = \mathbf{I^{(n)}} + \left[\mathbf{I^{(n)}} - \frac{3\mathbf{A}h}{4} + \frac{(\mathbf{A}h)^2}{4} - \frac{(\mathbf{A}h)^3}{24}\right]^{-1} \left(\mathbf{I^{(n)}} - \frac{\mathbf{A}h}{4} + \frac{(\mathbf{A}h)^2}{24}\right) \mathbf{A}h \tag{6.23}$$

Desarrollando el denominador en series de Taylor en torno a $h = 0.0$, resulta:

$$\mathbf{F} \approx \mathbf{I^{(n)}} + \mathbf{A}h + \frac{(\mathbf{A}h)^2}{2} + \frac{(\mathbf{A}h)^3}{6} + \frac{(\mathbf{A}h)^4}{24} + \frac{(\mathbf{A}h)^5}{96} \tag{6.24}$$

Luego, el método es efectivamente de cuarto orden, y el coeficiente de error es:

$$\varepsilon = \frac{1}{480}(\mathbf{A}h)^5 \tag{6.25}$$

Las regiones de estabilidad de los tres métodos se muestran en la Fig.6.2.

Los tres métodos tienen muy buenas regiones de estabilidad. En principio tienen un costo relativamente elevado: Radau IIA(3) debe iterar sobre dos evaluaciones de la función. Suponiendo que se necesitan en promedio cuatro iteraciones de Newton para la convergencia, esto implica que haya aproximadamente ocho evaluaciones por paso. Razonando de manera análoga, Lobatto IIIC(4) y Radau IIA(5) necesitan 12 evaluaciones por paso. Sin embargo, los métodos tienen una región de precisión muy buena y permiten utilizar pasos de integración muy grandes.

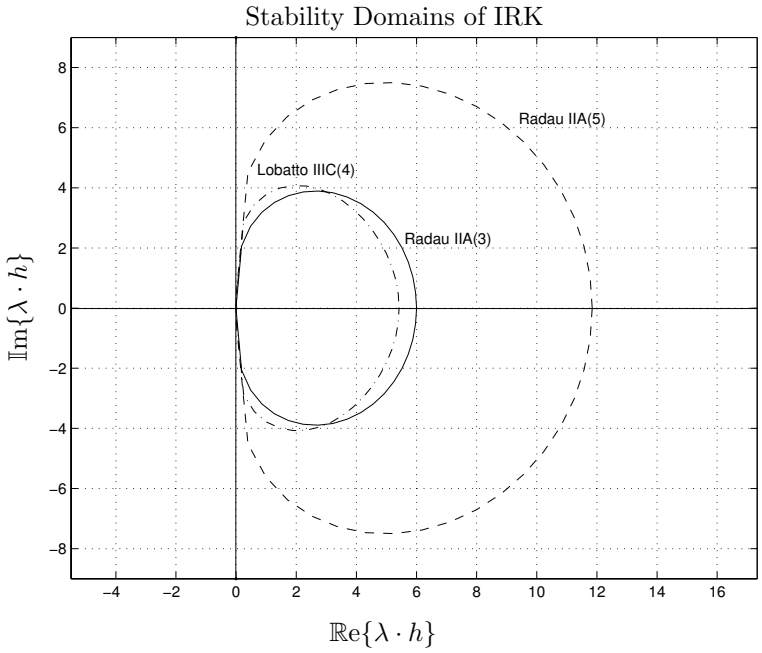

Figura 6.2: Dominios de estabilidad de algoritmos de RK completamente implícitos.

6.3. Fórmulas Multipaso

En los métodos monopaso encontramos ventajas al utilizar la formulación DAE para resolver problemas stiff. El motivo era que podíamos unir en una única iteración las correspondientes al modelo y al método implícito.

Entre los métodos que analizamos, encontramos ventajas con los métodos de RK completamente implícitos, ya que eran stiff estables y tenían pocas evaluaciones de la función $\mathbf{f}$ (y por ende un problema de iteraciones más pequeño).

En el caso de los métodos multipaso, esto nos lleva a considerar directamente los métodos de BDF, ya que requieren sólo una evaluación de la función $\mathbf{f}$ y son stiff estables.

Consideremos por ejemplo el método de BDF3:

$$\mathbf{x_{k+1}} = \frac{6}{11} h \cdot \dot{\mathbf{x}}_{\mathbf{k+1}} + \frac{18}{11}\mathbf{x_k} - \frac{9}{11}\mathbf{x_{k-1}} + \frac{2}{11}\mathbf{x_{k-2}} \tag{6.26}$$

La idea, como antes, es despejar $\dot{\mathbf{x}}_{\mathbf{k+1}}$ y reemplazarla en el modelo implícito (6.4). En este caso, obtenemos:

$$\dot{\mathbf{x}}_{\mathbf{k+1}} = \frac{1}{h}\left[\frac{11}{6} \cdot \mathbf{x_{k+1}} - 3\mathbf{x_k} + \frac{3}{2}\mathbf{x_{k-1}} - \frac{1}{3}\mathbf{x_{k-2}}\right] \tag{6.27}$$

Colocando la Ec.(6.27) dentro de la Ec.(6.4) evaluada para $\mathbf{x_{k+1}}$ obtenemos,

como en caso de Backwar Euler, una ecuación no lineal vectorial en la incógnita $\mathbf{x_{k+1}}$:

$$\mathcal{F}(\mathbf{x_{k+1}}) = 0.0 \tag{6.28}$$

que puede resolverse directamente mediante la iteración de Newton.

La única diferencia con el método de BE en su formulación implícita es que la función implícita $\mathcal{F}$ depende no sólo de $\mathbf{x_k}$ sino también de valores anteriores del estado, pero la resolución tiene prácticamente el mismo costo: hay que iterar sobre n ecuaciones con n incógnitas. Claramente, el costo de un paso con la versión implícita de cualquier método BDFn es mucho menor que el de los métodos de RK completamente implícitos.

Sin embargo, en general, al igual que en el caso ODE, los métodos de RK permiten utilizar pasos más grandes.

También, al igual que en las ODEs, los métodos de BDF no pueden arrancar por sí mismos (ya que requieren de varios valores pasados del estado). Una buena alternativa para arrancar los métodos es utilizar los métodos de RK completamente implícitos que vimos en la sección anterior.

En cuanto a la estabilidad de los métodos de BDF en su formulación DAE, las regiones son idénticas a las de los métodos en su formulación ODE.

6.4. Problemas Propuestos

[**P6.1**] **Calentamiento Eléctrico de una Barra II** Considerar nuevamente el Problema P5.2. En este caso, se pide resolverlo utilizando directamente la formulación DAE del método de BDF4.

Comparar los resultados y los tiempos de ejecución con los obtenidos utilizando la formulación ODE de BDF4 con la iteración de Newton dentro del modelo (como se hizo en el Problema P5.2).

Luego, sobre el mismo modelo, simular utilizando los métodos de Radau IIA(3) y Radau IIA(5) y comparar los resultados y los tiempos de ejecución.

[**P6.2**] **Control de Paso en Radau IIA(5)** Implementar algún algoritmo de control de paso para el método de Radau IIA(5).

Capítulo 7

Simulación de Sistemas Discontinuos

Este capítulo es en parte una traducción resumida del Capítulo 9 del libro *Continuous System Simulation* [2]. En este caso hay varios ejemplos distintos que en el caso del libro.

7.1. Introducción

Todos los algoritmos de integración utilizados se basan, explícita o implícitamente, en expansiones de Taylor. Las trayectorias siempre se aproximan mediante polinomios o mediante funciones racionales en el paso h en torno al tiempo actual t_k.

Esto trae problemas al tratar con modelos discontinuos, ya que los polinomios nunca exhiben discontinuidades, y las funciones racionales sólo tienen polos aislados, pero no discontinuidades finitas. Entonces, si un algoritmo de integración trata de integrar a través de una discontinuidad, sin dudas va a tener problemas.

Dado que el paso h es finito, el algoritmo de integración no reconoce una discontinuidad como tal. Lo único que nota es que la trayectoria de pronto cambia su comportamiento y actúa como si hubiera un gradiente muy grande. Esto es, el algoritmo verá lo mismo que ocurre cuando aparece un autovalor muy rápido (es decir, lejos a la izquierda del semiplano complejo). Si el algoritmo tiene control de paso, actuará reduciendo el paso de integración para atrapar al autovalor dentro de su dominio de estabilidad (o de precisión). Sin embargo, este autovalor virtual nunca quedará dentro del dominio de estabilidad.

Finalmente, ocurrirá o bien que el paso se reducirá al mínimo valor tolerado, o bien que falle el control de error. En consecuencia (si el control de error no falla de manera grosera), la discontiuidad se pasará con un paso muy pequeño y desaparecerá, y a partir de ese momento podrá agrandarse nuevamente el paso de integración.

De esta manera, los algoritmos con control de paso pueden tratar directamente las discontiunidades, aunque muchas veces esto falla.

La Figura 7.1 ilustra como los algoritmos de control de paso manejan las discontinuidades.

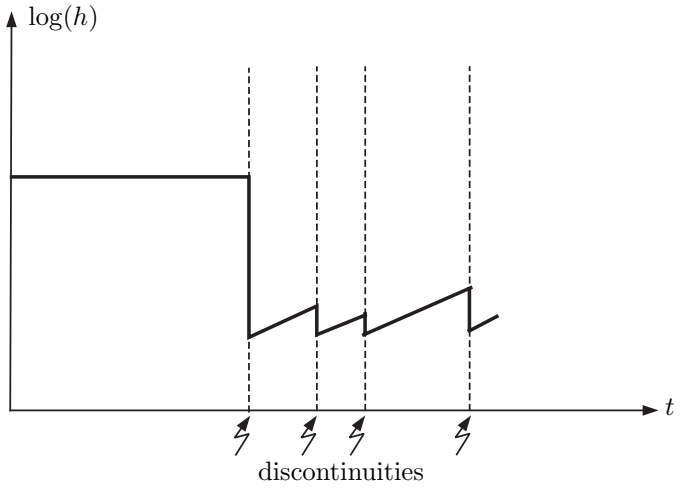

Figura 7.1: Manejo de discontinuidades mediante control de paso.

7.2. Dificultades Básicas

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

$$\dot{x}(t) = v(t) \tag{7.1a}$$

$$\dot{v}(t) = -g - s_w(t) \cdot \frac{1}{m}(k \cdot x(t) + b \cdot v(t)) \tag{7.1b}$$

donde

$$s_w = \begin{cases} 0 & \text{si } x(t) > 0 \\ 1 & \text{en otro caso} \end{cases} \tag{7.2}$$

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 7.2

se muestran los resultados de la simulación con pasos $h = 0.002$, $h = 0.001$ y $h = 0.0005$.

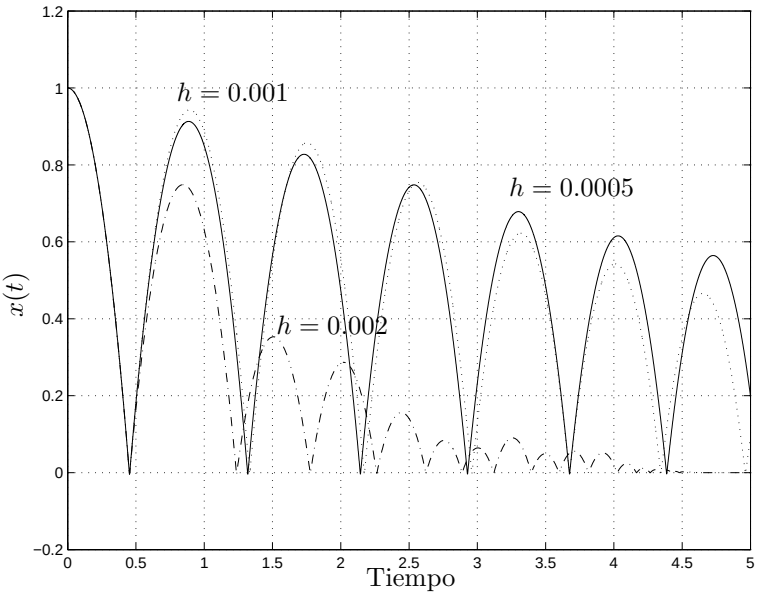

Figura 7.2: Simulación con RK4 de la pelotita rebotando.

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. Para esto, utilizamos un método de RK de segundo orden (la regla explícita del punto medio, que ya vimos en el capítulo 3), y para controlar el paso utilizaremos el mismo método implementado con dos semipasos de $h/2$. El método tendrá orden 2, y el término del error será de orden 3. Por lo tanto, lo llamaremos RK23.

En la Fig 7.3 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.

El problema de las simulaciones que hicimos es que en algunos pasos los métodos integraban a través de una discontinuidad. Es decir, calculábamos como si tuviéramos una función continua entre t_k y $t_k + h$, pero en realidad en el medio (en algún punto t^* en dicho intervalo) ocurría una discontinuidad.

La forma de evitar esto es en principio muy simple: lo que necesitamos es un método de paso variable que de un paso exactamente en el instante t^* en el que ocurre la discontinuidad. De esa forma, estaremos integrando una función

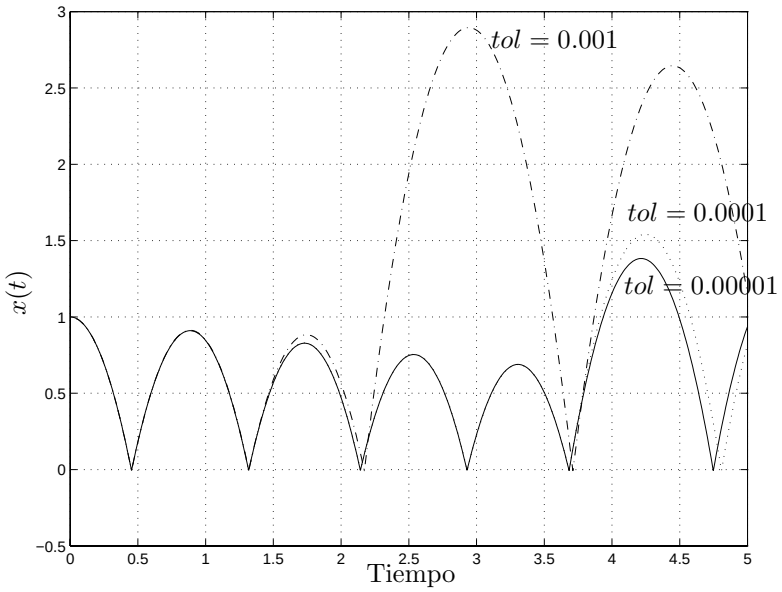

Figura 7.3: Simulación con RK23 de la pelotita rebotando.

continua antes de t^* y otra función continua después de t^*. Este es el principio básico de todos los métodos que realizan *manejo de discontinuidades*.

Comenzaremos entonces distinguiendo los distintos tipos de discontinuidades que pueden ocurrir en un sistema y la manera de tratarlas.

7.3. Eventos Temporales

El circuito de la Fig. 7.4 es un *elevador* de tensión, utilizado para conseguir una tensión continua superior a la de alimentación.

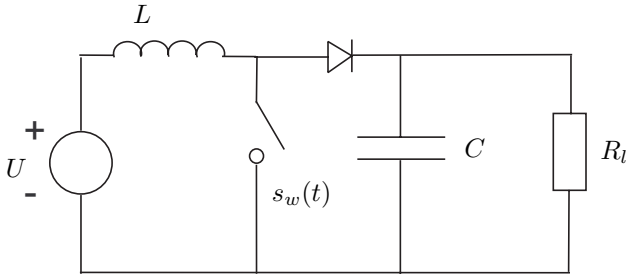

Figura 7.4: Circuito Elevador.

En este circuito, la llave se cierra y se abre periódicamente, y la tensión

sobre la carga R_l depende de la relación entre el tiempo que está cerrada y el tiempo que permanece abierta. Por otro lado, cuanto más alta es la frecuencia de conmutación, mas *continua* resulta la tensión de salida.

Un modelo simplificado de las ecuaciones que rigen la dinámica de este circuito es el siguiente:

$$\dot{i}_L(t) = \frac{1}{L}(U - s_w(t) \cdot v_C(t)) \tag{7.3a}$$

$$\dot{v}_C(t) = \frac{1}{C}(s_w(t) \cdot i_L(t) - \frac{v_C(t)}{R_l}) \tag{7.3b}$$

donde i_L es la corriente por la bobina, v_C es la tensión en el capacitor, U es la tensión continua de alimentación y $s_w(t)$ vale 1 (llave abierta) o 0 (llave cerrada).

Una forma típica de definir la señal $s_w(t)$ es la siguiente:

$$s_w(t) = \begin{cases} 0 & \text{si } t \cdot f - \text{int}(t \cdot f) < \delta \\ 1 & \text{en otro caso} \end{cases} \tag{7.4}$$

donde $0 \leq \delta \leq 1$ es el *ciclo activo* de la llave, es decir, la fracción de período que permanece cerrada y f es la frecuencia de conmutación. Notar que la llave permanece cerrada entre 0 y δ/f, abierta entre δ/f y $1/f$, nuevamente cerrada entre $1/f$ y $(1+\delta)/f$, etc.

En este caso las discontinuidades son debidas a la conmutación de la posición de la llave. En las ecuaciones, las discontinuidades se manifiestan por los saltos en el valor de $s_w(t)$.

Aquí, utilizando la Ec.(10.48) podemos saber de antemano cuando conmutará la llave. Este tipo de discontinuidades en las cuales se conoce con cierta anticipación el tiempo exacto de su ocurrencia se denominan *eventos temporales*.

La forma de tratar eventos temporales es muy sencilla. Dado que conocemos cuando ocurrirán, símplemente debemos avisarle al algoritmo de integración el tiempo de ocurrencia de los mismos. El algoritmo deberá entonces *agendar* dichos eventos y cada vez que de un paso deberá tener cuidado de no saltearse ninguno. Cada vez que el paso de integración h a utilizar sea mayor que el tiempo que falta para el siguiente evento, deberá utilizar un paso de integración que sea exactamente igual al tiempo para dicho evento.

Notar que ninguna discontinuidad tiene lugar mientras el evento es localizado. La discontinuidad no se debe codificar directamente en el modelo, sino sólo la condición para que esta ocurra. Así, las trayectorias vistas por el método de integración serán perfectamente continuas.

Una vez que el tiempo del siguiente evento ha sido localizado, la simulación continua se debe detener por un momento y se debe actualizar la parte discreta del modelo.

De esta manera, una corrida de simulación de un modelo discontinuo puede interpretarse como una secuencia de varias corridas de simulaciones continuas separadas mediante eventos discretos.

7.4. Eventos de Estado

Muy frecuentemente, el tiempo de ocurrencia de una discontinuidad no se conoce de antemano. Por ejemplo, en la pelotita rebotando, no se conoce el tiempo en el que se producen los piques. Todo lo que sabemos es que los piques se dan cuando la altura es cero. Es decir, conocemos la *condición del evento* en lugar del *tiempo del evento*.

Las condiciones de los eventos se suelen especificar como *funciones de cruce por cero*, que son funciones que dependen de las variables de estado del sistema y que se hacen cero cuando ocurre una discontinuidad. En el caso de la pelotita rebotando, una posible función de cruce por cero es $\mathcal{F}_0(x, v) = x$.

Decimos que ocurre un *evento de estado* cada vez que una función de cruce por cero cruza efectivamente por cero. En muchos casos, puede haber varias funciones de cruce por cero.

Las funciones de cruce por cero deben evaluarse continuamente durante la simulación. Las variables que resultan de dichas funciones normalmente se colocan en un vector y deben ser monitoreadas. Si una de ellas pasa a través de cero, debe comenzarse una iteración para determinar el tiempo de ocurrencia del cruce por cero con una precisión predeterminada.

Así, cuando una condición de evento es detectada durante la ejecución de un paso de integración, debe actuarse sobre el mecanismo de control de paso del algoritmo para forzar una iteración hacia el primer instante en el que se produjo el cruce por cero durante el paso actual.

Una vez localizado este tiempo, la idea es muy similar a la del tratamiento de eventos temporales.

7.4.1. Múltiples Cruces por Cero

La Figura 7.5 muestra un proceso de iteración sobre condiciones de eventos, suponiendo que durante el paso tuvieron lugar varios cruces por cero.

En la figura vemos tres funciones distintas de cruce por cero, f_1, f_2, y f_3. En el instante $t_k = 1.0$, f_1 es positiva, mientras f_2 y f_3 son negativas. Hacemos un paso de longitud $h = 3.0$. En $t_{k+1} = 4.0$, f_1 es aún positiva, pero ahora también f_2 y f_3 son positivas, por lo que hubo dos cruces por cero en este paso de integración.

Entonces conectamos los extremos de las funciones que cruzaron por cero con una linea recta y elegimos el menor de dichos instantes como el próximo punto del tiempo. Matemáticamente:

$$t_{\text{next}} = \min_{\forall i} \left[\frac{f_i(t_{k+1}) \cdot t_k - f_i(t_k) \cdot t_{k+1}}{f_i(t_{k+1}) - f_i(t_k)} \right] \tag{7.5}$$

donde i corresponde a cada función con cruce por cero en dicho intervalo. Luego de esto, repetimos el último paso con una longitud de $h = t_{\text{next}} - t_k$.

Si durante el paso desde t_k hasta t_{next} no hubo cruces, utilizaremos t_{next} en lugar de t_k y repetiremos el algoritmo utilizando lo que resta del intervalo.

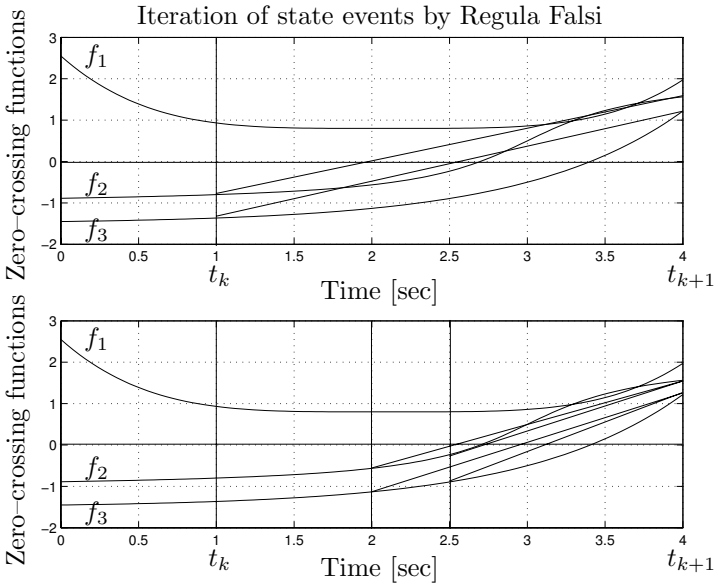

Figura 7.5: Iteración con Regula Falsi.

Si tuvo lugar más de un cruce por cero, lo que haremos es reducir t_{k+1} a t_{next}, y aplicaremos nuevamente el algoritmo sobre el intervalo reducido.

Si tuvo lugar exactamente un cruce por cero podríamos seguir de la misma forma, sin embargo podremos también aplicar métodos para encontrar un único cruce por cero (que veremos luego son más eficientes que este método).

Este algoritmo converge siempre, pero no se puede estimar en principio cuantas iteraciones son necesarias hasta alcanzar cierta precisión. Además, la convergencia suele ser lenta.

Otro algoritmo que se suele usar en lugar de Regula Falsi es el de la *Bisección*. La ventaja de este método es que en cada iteración el intervalo se reduce en una fracción fija (a la mitad). Así, el intervalo se reduce de manera bastante rápida.

Una vez que el algoritmo de iteración se dispara por varios cruces por cero en un paso, el intervalo se subdivide calculando un paso longitud $h/2$, comenzando desde el instante t_k. De esta forma, el intervalo se subdivide en dos subintervalos.

Si no hay cruces en el subintervalo de la izquierda, entonces ese intervalo se puede descartar, o sea, t_k se actualiza a $t_k + h/2$.

Si hay varios cruces por cero en el subintervalo de la izquierda, entonces el subintervalo de la derecha se puede descartar, y t_{k+1} se lleva a $t_k + h/2$.

Una vez actualizado t_k o t_{k+1} se puede repetir el procedimiento. En cada iteración el intervalo se reduce a la mitad.

Por último, cuando hay exactamente un cruce por cero en el subintervalo de la izquierda, entonces t_{k+1} se actualiza a $t_k + h/2$, y se puede continuar con cualquiera de los algorimos para encontrar cruces por cero individuales.

7.4.2. Cruces por Cero Simples. Métodos Monopaso

Cualquiera de los métodos vistos para aislar cruces por cero puede utilizarse para encontrar el tiempo del primer cruce por cero. De todas formas, esto puede ser ineficiente ya que sólo ofrecen velocidad de convergencia lineal.

Todas las variables de la simulación en un modelo en ecuaciones de estado puede ser expresado en términos de las variables de estado y de las entradas. Esto vale también para las funciones de cruce por cero. Por lo tanto, además de los valores de las funciones de cruce por cero podemos conocer sus derivadas.

Un primer algoritmo que explota esta posibilidad es la iteración de Newton. La Figura 7.6 muestra esta idea.

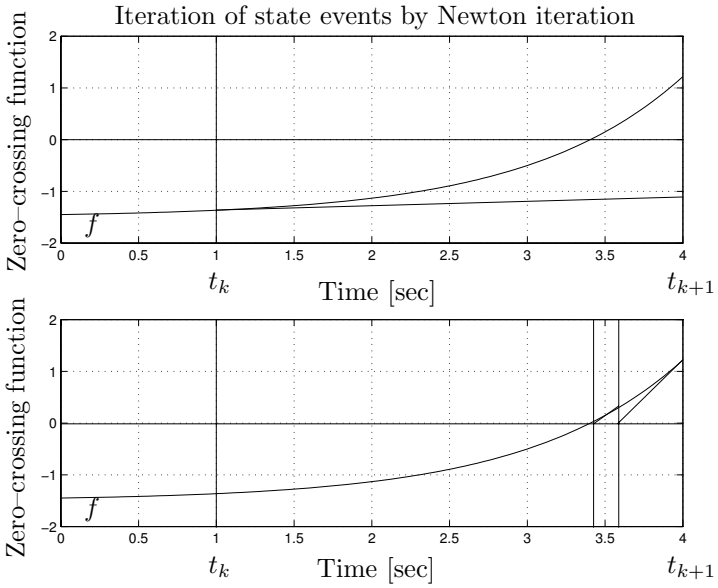

Figura 7.6: Iteración sobre una función de cruce por cero simple mediante Newton.

Una vez que la función queda aislada, se puede utilizar t_k o t_{k+1} como punto inicial de la iteración de Newton.

Lo bueno es que la convergencia tiene velocidad cuadrática, pero lamentablemente no se puede garantizar que la iteración siempre converja. Por este motivo, no se suele utilizar.

Un enfoque mejor podría ser utilizar los valores de las derivadas en ambos extremos del intervalo $[t_k, t_{k+1}]$ simultáneamente. Dado que tenemos acceso a cuatro piezas de información: f_k, df_k/dt, f_{k+1}, y df_{k+1}/dt, podemos ajustar un polinomio de tercer orden sobre las mismas y resolver para encontrar sus raices.

El polinomion de interpolación puede escribirse como:

$$p(t) = a \cdot t^3 + b \cdot t^2 + c \cdot t + d \tag{7.6}$$

y la derivada es:

$$\dot{p}(t) = 3a \cdot t^2 + 2b \cdot t + c \tag{7.7}$$

Luego, podemos utilizar la información que tenemos como sigue:

$$p(t_k) = a \cdot t_k^3 + b \cdot t_k^2 + c \cdot t_k + d = f_k \tag{7.8a}$$

$$p(t_{k+1}) = a \cdot t_{k+1}^3 + b \cdot t_{k+1}^2 + c \cdot t_{k+1} + d = f_{k+1} \tag{7.8b}$$

$$\dot{p}(t_k) = 3a \cdot t_k^2 + 2b \cdot t_k + c = \dot{f}_k = h_k \tag{7.8c}$$

$$\dot{p}(t_{k+1}) = 3a \cdot t_{k+1}^2 + 2b \cdot t_{k+1} + c = \dot{f}_{k+1} = h_{k+1} \tag{7.8d}$$

que puede escribirse en forma matricial como:

$$\begin{pmatrix} f_k \\ f_{k+1} \\ h_k \\ h_{k+1} \end{pmatrix} = \begin{pmatrix} t_k^3 & t_k^2 & t_k & 1 \\ t_{k+1}^3 & t_{k+1}^2 & t_{k+1} & 1 \\ 3t_k^2 & 2t_k & 1 & 0 \\ 3t_{k+1}^2 & 2t_{k+1} & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} \tag{7.9}$$

La Ec. 7.9 puede entonces resolverse para despejar los coeficientes desconocidos a, b, c, y d.

La Figura 7.7 ilustra el método.

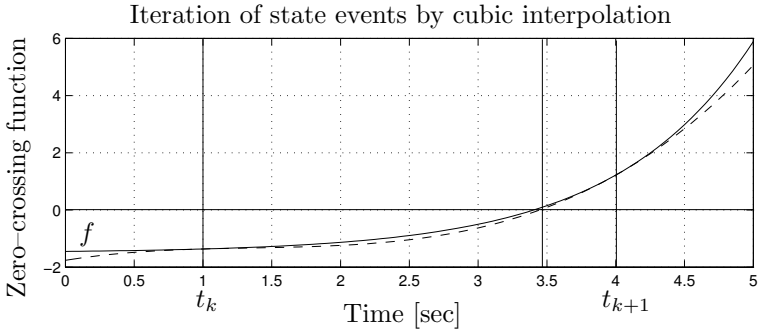

Figura 7.7: Iteración en una función de cruce por cero simple utilizando interpolación cúbica.

Este método converge más rápido que la iteración de Newton, ya que tiene velocidad de convergencia cúbica. Más aún, se puede garantizar que siempre converge.

El polinomio cúbico siempre tiene al menos una solución real en el intervalo $[t_k, t_{k+1}]$. Puede incluso haber tres soluciones reales en el intervalo. En tal caso, cualquiera de las tres puede utilizarse como próximo tiempo de evaluación, t_{next}.

7.4.3. Cruces por Cero Simples, Métodos Multipaso

En el caso de algoritmos multipaso, podemos hacer algo aún mejor. Al final del paso que activó la condición del evento, es decir, en el instante t_{k+1},

conocemos el vector de Nordsieck. Luego, podemos escribir:

$$\begin{aligned}\mathcal{F}(\hat{h}) =& \mathcal{F}_i(t_{\text{next}}) = \mathcal{F}_i(t_{k+1}) + \hat{h}\frac{d\mathcal{F}_i(t_{k+1})}{dt} + \frac{\hat{h}^2}{2}\frac{d^2\mathcal{F}_i(t_{k+1})}{dt^2} \\ &+ \frac{\hat{h}^3}{6}\frac{d^3\mathcal{F}_i(t_{k+1})}{dt^3} + \cdots = 0.0\end{aligned} \tag{7.10}$$

Esto es función de la incógnita $\hat{h}$ que puede resolverse mediante la iteración de Newton. Comenzamos con:

$$\hat{h}^0 = 0.5 \cdot (t_k - t_{k+1}) \tag{7.11}$$

e iteramos:

$$\hat{h}^{\ell+1} = \hat{h}^{\ell} - \frac{\mathcal{F}(\hat{h}^{\ell})}{\mathcal{H}(\hat{h}^{\ell})} \tag{7.12}$$

donde:

$$\mathcal{H}(\hat{h}) = \frac{d\mathcal{F}(\hat{h})}{d\hat{h}} = \frac{d\mathcal{F}_i(t_{k+1})}{dt} + \hat{h}\frac{d^2\mathcal{F}_i(t_{k+1})}{dt^2} + \frac{\hat{h}^2}{2}\frac{d^3\mathcal{F}_i(t_{k+1})}{dt^3} + \ldots \tag{7.13}$$

Mediante esta técnicas, podemos determinar el tiempo del cruce en un sólo paso con la misma precisión que la integración.

7.5. Estados Discretos y Condiciones Iniciales Consistentes

La Figura 7.8 muestra una función seccionalmente lineal con tres segmentos. En la región de la "izquierda", se trata de $y = a_1 \cdot x + b_1$, en la del "centro", $y = a_2 \cdot x + b_2$, y en la de la "derecha", $y = a_3 \cdot x + b_3$.

Tradicionalmente, describiríamos esta función de la siguiente forma:

if $x < x_1$ **then** $y = a_1 \cdot x + b_1$
  else if $x < x_2$ **then** $y = a_2 \cdot x + b_2$
  else $y = a_3 \cdot x + b_3$;

Sin embargo, si utilizamos esta función en un modelo de ecuaciones de estado, tendremos problemas cada vez que x cruce por los valores x_1 y x_2.

Ya sabemos que como primera medida deberemos incorporar las componentes $f_1 = x - x_1$ y $f_2 = x - x_2$ a la función de cruces por cero, para asegurarnos que el algoritmo de integración no integre a través de las discontinuidades.

Lo malo es que con el modelo anterior, aún teniendo en cuenta las funciones de cruce por cero, tendremos problemas: Cuando detectemos una condición de cruce, ya habremos integrado a través de la discontinuidad, y por lo tanto el valor al final del paso no sólo será *incorrecto* sino que probablemente no tendrá ningún

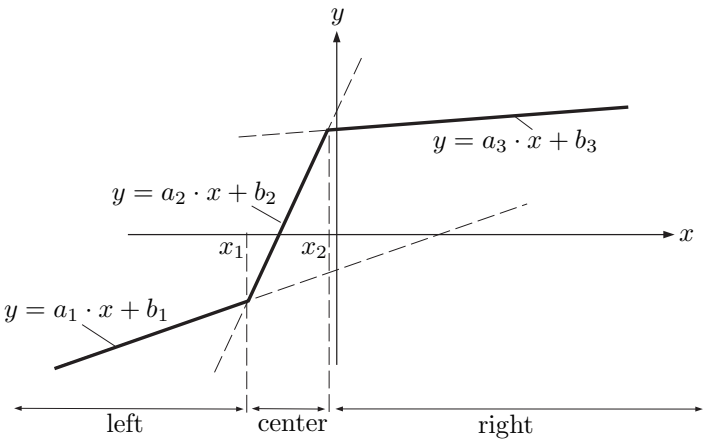

Figura 7.8: Función discontinua.

sentido. Por lo tanto, salvo que utilicemos un método al estilo de la bisección, el valor no nos servirá para realizar iteraciones mediante polinomios.

Más aún, si utilizamos cualquier método que evalúe la función en $t_k + h$ o incluso en $t_k + h/2$ (RK4 por ejemplo), utilizará valores incorrectos de las derivadas, y es posible incluso que este nuevo valor del estado no provoque cruces.

Por lo tanto, es necesario separar la parte continua de la parte discreta. Es decir, debemos agregar una variable discreta que seleccione entre los posibles modelos continuos de la siguiente forma:

case *region*
  $left$: $y = a_1 \cdot x + b_1$;
    schedule *Center* **when** $x - x_1 == 0$;
  $center$: $y = a_2 \cdot x + b_2$;
    schedule $Left$ **when** $x - x_1 == 0$;
    schedule *Right* **when** $x - x_2 == 0$;
  $right$: $y = a_3 \cdot x + b_3$;
    schedule *Center* **when** $x - x_2 == 0$;
end;

En este caso, *region* es la variable discreta, que deberá tener un valor coherente con el valor de x. De todas formas, para la parte continua de la simulación, la variable *region* actua como un parámetro. Por lo tanto, la integración continua siempre ve un modelo continuo y no hay riesgo de pasar por una discontinuidad.

El pseudo comando **schedule** lo estamos utilizando para explicitar las funciones de cruce por cero.

El modelo puede completarse con una descripción de la evolución discreta:

```
event Left
   region := left;
end Left;

event Center
   region := center;
end Center;

event Right
   region := right;
end Right;
```

En principio, el método de simulación al encontrarse con un modelo como este funcionaría así: integramos en la parte continua hasta que detectamos un cruce por cero. Iteramos entonces hasta encontrar con precisión el punto en el que se produce el cruce, damos el paso correspondiente y luego actualizamos la o las variables discretas.

Esto debería funcionar salvo por un pequeño detalle. La variable *region* es parte del *estado discreto* y necesita inicializarse. En algún punto del programa de simulación, necesitaremos unas órdenes del tipo

```
if x < x1 then region := left;
   else if x < x2 then region := center;
   else region := right;
```

¿Funcionará esto siempre?. Desafortunadamente, la respuesta es no. Un problema que no hemos considerado aún es que x podría alcanzar uno de los valores umbrales sin cruzarlo.

Consideremos por ejemplo:

$$x(t) = \frac{x_2 - x_1}{2} \cdot \sin(t) + \frac{x_1 + x_2}{2} \tag{7.14}$$

En este caso, x siempre permanecerá en la región central aunque tocará de vez en cuando los umbrales x_1 y x_2. Sin embargo, el modelo utilizado activará los cambios de región cada vez que el umbral sea alcanzado.

La parte dificil de este problema es determinar en que región quedará el modelo después de cada discontinuidad, es decir, encontrar un conjunto consistentes de condiciones iniciales.

Si bien hay varias soluciones (una alternativa es definir una banda angosta en torno a la condición de cruce y conmutar el modelo sólo cuando al dar un paso de prueba desde el cruce se atraviese dicha banda), las mismas exceden los objetivos de este curso.

7.6. Problemas Propuestos

[P7.1] Simulación con Eventos Temporales Programar algún método de paso variable (RKF4/5 es una alternativa) e incorporarle una rutina para el manejo de eventos temporales. Para esto suponer que la función que define el modelo devuelve no sólo la derivada, sino también el tiempo del próximo evento.

Simular con este método el circuito elevador de la Fig.7.4, considerando condiciones iniciales nulas y los parámetros $C = 220 \times 10^{-6}$; $L = 150 \times 10^{-6}$; $U = 5$; $f = 25000$ y $\delta = 0.63$. Se pide simular durante 0.1 seg. y comparar los resultados con los que se obtienen sin detectar discontinuidades.

[P7.2] Detección de Eventos de Estado – Regula Falsi

Implementar el algoritmo RKF4/5 del Capítulo 3, y agregarle una rutina de detección de discontinuidades basado en el algoritmo de *Regula Falsi*.

Verificar el funcionamiento de la nueva rutina simulando el sistema de la pelotita rebotando.

[P7.3] Detección de Eventos de Estado – Bisección

Repetir el Prob.[P7.2], pero ahora utilizar el método de la bisección para encontrar las raíces de las funciones de cruce por cero.

[P7.4] Detección de Eventos de Estado – Interpolación Cúbica

Repetir el Prob.[P7.2], agregando ahora la iteración por interpolación cúbica.

[P7.5] Eventos Temporales y de Estado

Un modelo más preciso del circuito elevador de la Fig.7.4 debe contemplar la posibilidad de que el diodo deje de conducir.

En tal caso, utilizando la idea de los estados discretos, podemos replantear las ecuaciones utilizando dos variables discretas: *llave* y *diodo*. La variable *llave* puede tomar los valores *abierta* y *cerrada*, mientras que la variable *diodo* puede tomar los valores *conduce* y *cortado*.

Para la situación $llave = abierta$ y $diodo = conduce$, tenemos:

$$\dot{i}_L(t) = \frac{1}{L}(U - v_C(t)) \tag{P7.5aa}$$

$$\dot{v}_C(t) = \frac{1}{C}(i_L(t) - \frac{v_C(t)}{R_l}) \tag{P7.5ab}$$

y deberemos detectar la condición de evento $i_L = 0$, que nos llevará a la situación $diodo = cortado$.

Para la situación $llave = abierta$ y $diodo = cortado$ en tanto, tenemos:

$$\dot{i}_L(t) = 0 \tag{P7.5ba}$$

$$\dot{v}_C(t) = \frac{1}{C}(i_L(t) - \frac{v_C(t)}{R_l}) \tag{P7.5bb}$$

y aquí deberemos detectar la condición de evento $U - v_C(t) = 0$, que nos llevará a la situación $diodo = conduce$.

Para la situación $llave = cerrada$ y $diodo = cortado$, tenemos:

$$\dot{i}_L(t) = \frac{1}{L} U \tag{P7.5ca}$$

$$\dot{v}_C(t) = -\frac{v_C(t)}{R_l \cdot C} \tag{P7.5cb}$$

y en este caso no es necesario detectar ninguna condición de evento.

Por otro lado, al pasar de la situación $llave = cerrada$ a la situación $llave = abierta$ se debe pasar también a $diodo = conduce$.

Finalmente, la situación $llave = cerrada$ y $diodo = conduce$ no puede darse nunca.

Repetir entonces el Prob.[P7.1] utilizando este nuevo modelo y comparar los resultados con los obtenidos utilizando el modelo simplificado.

[P7.6] Modelo de una Neurona Pulsante

Un modelo simple de una *Spiking Neuron*, que se utiliza en algunas aplicaciones de inteligencia artificial, es el siguiente [4]:

$$\dot{v}(t) = 0.04 \cdot v^2 + 5 \cdot v + 140 - u + I(t) \tag{P7.6aa}$$

$$\dot{u}(t) = a \cdot (b \cdot v - u) \tag{P7.6ab}$$

donde $v(t)$ representa el *potencial de membrana* y $u(t)$ modela el fenómeno de restitución de la membrana. La variable $I(t)$ es una corriente de entrada y los restantes parámetros son constantes que dependen del modelo de neurona.

Cuando en un tiempo t la variable $v(t)$ alcanza el valor $v(t) = 30$, se produce el disparo de la neurona, tras el cual las variables de estado se resetean de la siguiente forma:

$$\begin{cases} v(t^+) & = c \\ u(t^+) & = u(t) + d \end{cases} \tag{P7.6b}$$

al igual que a y b, c y d son constantes.

Simular este sistema considerando: $a = 0.02$; $b = 0.2$; $c = -65$; $d = 8$; $I(t) = 30$. Tomar como condición inicial $v(0) = -60$, $u(0) = 0$ y usar un tiempo final de simulación de 300 seg.

Capítulo 8

Simulación en Tiempo Real

Este capítulo es una traducción resumida del Capítulo 3 del libro *Continuous System Simulation* [2].

8.1. Introducción

Muchas aplicaciones muy importantes de la simulación necesitan satisfacer restricciones de tiempo real.

Un *simulador de vuelo* de entrenamiento sólamente es útil si puede reflejar en tiempo real el comportamiento de un avión o un helocóptero, para lo cual además debe ser capaz de procesar los datos de entrada de la persona que lo está utilizando.

En algunas aplicaciones de control se utiliza un modelo idealizado de una planta (denominado planta de referencia), y se trata de lograr que la planta real se comporte lo más parecido posible a la planta de referencia. Esto requiere que el modelo de la plante de referencia se simule en tiempo real, en paralelo con la planta real, de manera que ambos sistemas reciban las mismas señales de entrada.

Una manera de detectar fallas en un sistema consiste en simular un modelo de dicho sistema en tiempo real, y comparar ciertas variables importantes de la simulación con las variables del sistema real. Cuando las discrepancias se tornan grandes, se puede inferir que se está en presencia de una falla en el sistema real. Para esto, también es necesario que la simulación reciba las mismas entradas que la planta.

Conceptualmente, la implementación de un software de simulación en tiempo real es muy sencilla. Hay sólamente cuatro nuevos componentes.

1. El *Reloj de tiempo real* es el responsable de la sincronización entre el tiempo simulado y el tiempo real. Generalmente el reloj de tiempo real se programa para disparar una señal cada h unidades de tiempo real, donde h es el paso del método de integración. El programa que ejecuta la simulación contiene también un ciclo de espera, en el cual se entra tras

haber finalizado los cálculos del paso actual, y del cual se sale tras recibir la señal proveniente del reloj de tiempo real.

2. Los *conversores análogico–digitales (A/D)* se leen al principio de cada paso de integración para actualizar los valores de todas las señales externas que se utilizan en la simulación. Esto corresponde a mecanismos de *muestreo y retención* (S/H, por *Sample and Hold*), es decir, estos valores leidos se mantienen constantes durante todo el paso.

3. Los *conversores digitales–analógicos (D/A)* se actualizan al final de cada paso de integración. Estos brindan la última información necesaria para el mundo exterior a la simulación.

4. Los *Eventos Externos* son eventos temporales que se generan por fuera de la simulación, y que permiten la comunicación asincrónica con el programa de simulación con distintos propósitos. Estos eventos (al estilo de interrupciones) normalmente se posponen para el final del paso actual.

La Figura 8.1 ilustra las diferentes tareas que tienen lugar durante la ejecución de un paso de integración.

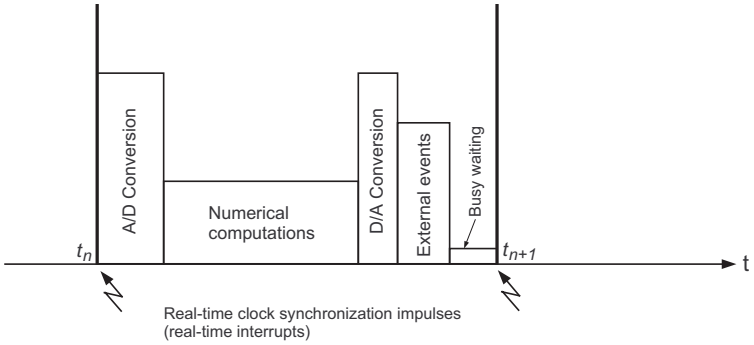

Figura 8.1: Tareas asociadas a un paso de integración.

Una vez que el mensaje del reloj de tiempo real indica que el tiempo real avanzó al instante t_k, el programa de simulación en primer lugar debe leer todos los conversores A/D para actualizar los valores de las señales de entrada. Luego, debe realizar los cálculos numéricos correspondientes al paso utilizando alguna rutina de integración. Tras esto, los resultados se deben pasar a los conversores D/A. En ese momento, el trabajo "normal" asociado con cada paso estaría cumplido. El algoritmo entonces debe fijarse si no ocurrió ninguna interrupción externa, y en tal caso deberá procesarla. Una vez cumplido esto, se deberá entrar en el ciclo de espera hasta que llegue el nuevo mensaje del reloj de tiempo real.

Por supuesto, hay variantes respecto a la manera de hacer esto. Por ejemplo, el ciclo de espera pordría intentar aprovecharse para hacer otras tareas (en ese caso el reloj de tiempo real debería tener capacidad de generar interrupciones).

Las dificultades de la simulación en tiempo real no son de naturaleza conceptual, sino que tienen que ver con garantizar que los cálculos y las tareas finalicen antes del siguiente pulso del reloj. Es decir, la principal dificultad radica en que la simulación debe correr en cada paso una carrera contra el tiempo.

8.2. La Carrera Contra el Tiempo

Hay dos preguntas que debemos hacernos en el contexto de correr contra el tiempo: (1) ¿Cómo garantizamos que todos los cálculos terminen antes que llegue el siguiente pulso del reloj? (ii) ¿Que pasa si no lo conseguimos?

Comenzaremos por la segunda pregunta, ya que es más simple de contestar. En principio, hay cuatro cosas que podríamos hacer si nuestros cálculos demoran demasiado:

1. incrementar el paso, h, para tener más tiempo para terminar las tareas necesarias,
2. hacer más eficiente la evaluación de las funciones, optimizando el programa que representa el modelo en ecuaciones de estado,
3. mejorar la velocidad de los algoritmos de integración, reduciendo por ejemplo el número de evaluaciones de función necesarios en cada paso, y/o
4. comprarnos una computadora más rápida.

En realidad, la última solución no es una mala idea, teniendo en cuenta los costos cada vez más bajos del hardware comparado con el software y la mano de obra.

La primer solución es interesante. Hasta aquí el paso de integración siempre estuvo acotado por encima debido a restricciones de estabilidad y precisión. Ahora, de pronto, el paso está también acotado por debajo debido a la restricción de tiempo real. Obviamente, si este límite inferior es mayor que el límite superior impuesto por la estabilidad y la precisión, estaremos en problemas.

La segunda solución ha sido muy explorada, y si bien es importante, escapa a los propósitos de este curso.

La tercera opción es jústamente la que tiene que ver con nuestros objetivos, ya que involucra directamente a los métodos de integración numérica.

De todas formas, antes de estudiar las maneras de mejorar la velocidad de los algoritmos, analizaremos los distintos métodos para dedicarnos sólamente a aquellos que tengan características útiles para simular en tiempo real.

8.3. Métodos de Integración Numérica Apropiados

En la simulación en tiempo real, no alcanza con obtener una buena aproximación de los valores de las variables de estado. Más aún, estas aproximaciones

son inútiles si se obtienen demasiado tarde. Por lo tanto, debemos estar seguros que todos los cálculos asociados a cada paso finalicen dentro del intervalo de tiempo asignado.

Para esto, el número total de cálculos hecho por cada paso de integración debe acotarse, y todos los procesos iterativos deben cortarse tras un número fijo de iteraciones. Esto, claramente afectará la precisión de los algoritmos. Sin embargo es preferible obtener una solución con algo de error que no poder obtener nada en el tiempo permitido.

En base a estas consideraciones básicas, el siguiente análisis intenta examinar las diferentes características de los métodos que vimos en los capítulos anteriores para discutir sus pros y sus contras en el contexto de la simulación en tiempo real.

- *Métodos Multipaso*. Los métodos multipaso utilizan información de los pasos anteriores para calcular una aproximación de orden alto del siguiente paso. Esta idea se basa en el supuesto de que la ecuación diferencial es suave. Sin embargo, normalmente las simulaciones en tiempo real reciben señales de entrada que no son suaves y por lo tanto, los resultados de los métodos multipaso suelen ser imprecisos.

  Por otro lado, estos métodos reducen el número de evaluaciones de las funciones por paso, lo que es un factor crucial en la simulación en tiempo real. Por esto, pese a la pérdida de precisión, los métodos multipaso explícitos (tales como AB) de orden bajo suelen utilizarse.

- *Métodos explícitos monopaso*. Estos métodos son compatibles con los requerimientos mencionados antes. El esfuerzo computacional es relativamente bajo y constante. El número de cálculos por paso puede estimarse fácilmente. Más aún, los métodos pueden tratar bastante bien con señales de entrada discontinuas ya que no utilizan información del pasado. Por esto, para ecuaciones diferenciales ordinarias no stiff, los métodos explícitos monopaso suelen ser la mejor opción.

  En el caso de sistemas no muy stiff, estos métodos pueden aún utilizarse con un paso chico (una fracción del intervalo de muestreo). Pero esta idea empieza a fallar a medida que aumenta la rigidez de los modelos.

- *Métodos implícitos monopaso*. Como sabemos, los métodos implícitos requieren resolver un sistema de ecuaciones no lineal en cada paso, lo que implica utilizar métodos de iteración. Por lo tanto, no puede predecirse exactamente el costo computacional ya que el mismo depende del número de iteraciones, que en principio es no acotado. Por esto, los métodos implícitos no son apropiados para tiempo real. De todas formas, en base a estos métodos se pueden plantear algoritmos útiles que mantengan acotado por un valor fijo el número de iteraciones.

- *Métodos de orden alto*. En la mayor parte de las aplicaciones en tiempo real, los intervalos de muestreo son pequeños en comparación con las escalas de tiempo de interés (para poder tomar adecuadamente las señales

de entrada), y la precisión requerida suele ser bastante baja. Teniendo en cuenta además que es crucial reducir el número de cálculos, es muy raro encontrar una simulación de tiempo real que utilice métodos de integración de orden mayor que dos ó tres.

- *Métodos de paso variable*. En las simulaciones en tiempo real no podemos darnos el lujo de cambiar el paso, ya que el mismo debe estar sincronizado con la frecuencia de muestreo y muy restringido con la especificación de tiempo real del problema.

Tras aeste análisis, sólamente los métodos explícitos de bajo orden parecen ser apropiados para la simulación en tiempo real. De hecho, en ausencia de rigidez, discontinuidades o ecuaciones implícitas muy alineales, estos métodos funcionarán muy bien.

Desafortunadamente, la mayor parte de los sistemas de la ingeniería son de hecho stiff. Veremos entonces algunos nuevos algoritmos y algunas ideas para tratar estos problemas en el contexto de tiempo real.

8.4. Métodos Linealmente Implícitos

Los métodos linealmente implícitos o semi–implícitos explotan el hecho de que los métodos implícitos aplicados a sistemas lineales no requieren un número no acotado de iteraciones, sino que pueden implementarse directamente mediante inversión matricial.

Uno de los métodos de este tipo más utilizados es la fórmula de Euler semi–implícita:

$$\mathbf{x_{k+1}} = \mathbf{x_k} + h \cdot [\mathbf{f}(\mathbf{x_k}, t_k) + \mathcal{J}_{\mathbf{x_k},t_k} \cdot (\mathbf{x_{k+1}} - \mathbf{x_k})] \tag{8.1}$$

donde

$$\mathcal{J}_{\mathbf{x_k},t_k} = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\mathbf{x_k},t_k} \tag{8.2}$$

es la matriz Jacobiana evaluada ent $(\mathbf{x_k}, t_k)$.

Notar que

$$\mathcal{J}_{\mathbf{x_k},t_k} \cdot (\mathbf{x_{k+1}} - \mathbf{x_k}) \approx \mathbf{f}(\mathbf{x_{k+1}}, t_{k+1}) - \mathbf{f}(\mathbf{x_k}, t_k) \tag{8.3}$$

y entonces:

$$\mathbf{f}(\mathbf{x_k}, t_k) + \mathcal{J}_{\mathbf{x_k},t_k} \cdot (\mathbf{x_{k+1}} - \mathbf{x_k}) \approx \mathbf{f}(\mathbf{x_{k+1}}, t_{k+1}) \tag{8.4}$$

Es decir, el método de Euler linealmente implícito se aproxima al método de Backward Euler. Más aún, en el caso lineal:

$$\dot{\mathbf{x}} = \mathbf{A} \cdot \mathbf{x} \tag{8.5}$$

tenemos:

$$\mathbf{x_{k+1}} = \mathbf{x_k} + h \cdot [\mathbf{A} \cdot \mathbf{x_k} + \mathcal{J}_{\mathbf{x_k},t_k} \cdot (\mathbf{x_{k+1}} - \mathbf{x_k})] = \mathbf{x_k} + h \cdot \mathbf{A} \cdot \mathbf{x_{k+1}} \tag{8.6}$$

que coincide exactamente con Backward Euler. Esto implica que el dominio de estabilidad del método de Euler semi–implícito coincide con el de BE.

La Ecuación (8.1) puede reescribirse como:

$$(I - h \cdot \mathcal{J}_{\mathbf{x_k},t_k}) \cdot \mathbf{x_{k+1}} = (I - h \cdot \mathcal{J}_{\mathbf{x_k},t_k}) \cdot \mathbf{x_k} + h \cdot \mathbf{f}(\mathbf{x_k}, t_k) \tag{8.7}$$

lo que muestra que $\mathbf{x_{k+1}}$ puede obtenerse resolviendo un sistema lineal de ecuaciones.

El valor de $\mathbf{x_{k+1}}$ puede también obtenerse como:

$$\mathbf{x_{k+1}} = \mathbf{x_k} + h \cdot (I - h \cdot \mathcal{J}_{\mathbf{x_k},t_k})^{-1} \cdot \mathbf{f}(\mathbf{x_k}, t_k) \tag{8.8}$$

Esta fórmula es similar a la de Forward Euler, pero difiere en la presencia del término $(I - h \cdot \mathcal{J}_{\mathbf{x_k},t_k})^{-1}$.

Este término implica que el algoritmo debe calcular el Jacobiano en cada paso e invertir una matriz.

Aunque estos cálculos puedan ser algo caros, el esfuerzo computacional es predecible, lo que hace que el método sea adecuado para tiempo real.

Teniendo en cuenta que el dominio de estabilidad coincide con el de BE, este método resulta apropiado para simular sistemas stiff y sistemas de DAEs. De hecho, los métodos linealmente implícitos de bajo orden son a menudo la mejor opción para la simuación en tiempo real. Sin embargo, cuando la dimensión del problema es grande, el costo de la inversión matricial puede resultar demasiado caro.

Afortunadamente, la rigidez en sistemas grandes está frecuentemente relacionada con la presencia de algunos subsistemas lentos y otros rápidos que se pueden identificar. En estos casos, podemos sacar provecho de esta información y utilizar distintos pasos de integración e incluso distintos algoritmos de integración en cada parte. Estas ideas nos llevan a los conceptos de integración multipaso e integración de modo mixto.

8.5. Integración Multitasa

Hay muchos casos en los que la rigidez se debe a la presencia de un subsistema con dinámica muy rápida comparada con el resto del sistema. Esto ocurre típicamente en los sistemas físicos multidominio, ya que los componentes de los distintos dominios de la física generalmente tienen asociadas constantes de tiempo muy diferentes.

Por ejemplo, si queremos estudiar las propiedades térmicas de un circuito integrado, debemos tener en cuenta que las constantes de tiempo eléctricas del dispositivo son mucho más rápidas que las constantes de tiempo térmicas. Sin embargo, no podemos ignorar las constantes de tiempo rápidas de la parte eléctrica, ya que son la causa del calentamiento del dispositivo.

Veamos en más detalle esta idea en un ejemplo. La Figura 8.2 muestra un modelo concentrado de una línea de transmisión alimentada por un oscilador de Van–der–Pol.

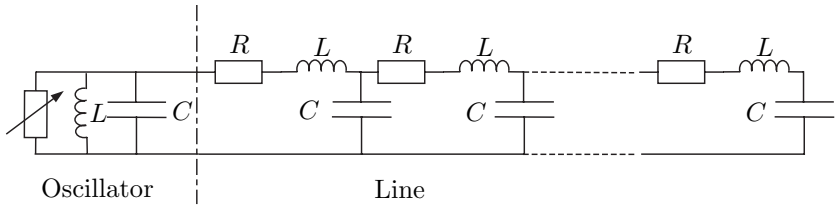

Figura 8.2: Oscilador de Van–der–Pol y línea de transmisión.

Supondremos que el resistor no lineal del circuito oscilador satisface la ley:

$$i_R = k \cdot u_R^3 - u_R \tag{8.9}$$

Luego, el sistema puede describirse mediante las siguientes ecuaciones:

$$\begin{aligned}
\frac{\mathrm{d}i_L}{\mathrm{d}t} &= \frac{1}{L}u_C && \text{(8.10a)}\\
\frac{\mathrm{d}u_C}{\mathrm{d}t} &= \frac{1}{C}(u_C - k \cdot u_C^3 - i_L - i_1) && \text{(8.10b)}\\
\frac{\mathrm{d}i_1}{\mathrm{d}t} &= \frac{1}{L}u_C - \frac{R}{L}i_1 - \frac{1}{L}u_1 && \text{(8.10c)}\\
\frac{\mathrm{d}u_1}{\mathrm{d}t} &= \frac{1}{C}i_1 - \frac{1}{C}i_2 && \text{(8.10d)}\\
\frac{\mathrm{d}i_2}{\mathrm{d}t} &= \frac{1}{L}u_1 - \frac{R}{L}i_2 - \frac{1}{L}u_2 && \text{(8.10e)}\\
\frac{\mathrm{d}u_2}{\mathrm{d}t} &= \frac{1}{C}i_2 - \frac{1}{C}i_3 && \text{(8.10f)}\\
&\vdots\\
\frac{\mathrm{d}i_n}{\mathrm{d}t} &= \frac{1}{L}u_{n-1} - \frac{R}{L}i_n - \frac{1}{L}u_n && \text{(8.10g)}\\
\frac{\mathrm{d}u_n}{\mathrm{d}t} &= \frac{1}{C}i_n && \text{(8.10h)}
\end{aligned}$$

Aquí, u_C e i_L son el voltaje y la corriente del capacitor y la inductancia en el oscilador. De manera similar, u_j e i_j son el voltaje y la corriente de los capacitores e inductancias en la etapa j de la línea.

Supondremos que la línea de transmisión tiene 5 etapas (es decir., $n = 5$), y que los parámetros son $L = 10$ mH, $C = 1$ mF, $R = 10\Omega$ y $k = 0.04$.

Si queremos simular el sistema utilizando el método de Forward Euler, necesitaremos utilizar un paso no mayor que $h = 10^{-4}$ segundos. De otra forma, la salida del oscilador (u_C) tendrá un error inaceptable.

Sin embargo, utilizando la señal de entrada generada por el oscilador, la linea de transmisión sola puede simularse con un paso 10 veces mayor.

Entonces, decidimos dividir el sistema en dos partes, el circuito oscilador y la linea de transmisión, con dos pasos de integración distintosL 10^{-4} segundos para el primero y 10^{-3} segundos para el segundo.

De esta forma, integramos el subsistema rápido pero pequeño con un paso pequeño, e integramos el subsistema lento pero grande con un paso mayor.

En consecuencia, durante cada milisegundo de tiempo real,la computadore debe evaluar diez veces las dos funciones escalares corresponientes a las dos primeras ecuaciones de estado, mientras que sólo debe evaluar una vez las restantes 10 funciones escalares. Por esto, el número de operaciones de punto flotante se reduce en un factor de cuatro aproximadamente, comparado con la utilización del mismo paso pequeño sobre todo el sistema.

Los resultados de simuación se muestran en las Figs.8.3–8.4.

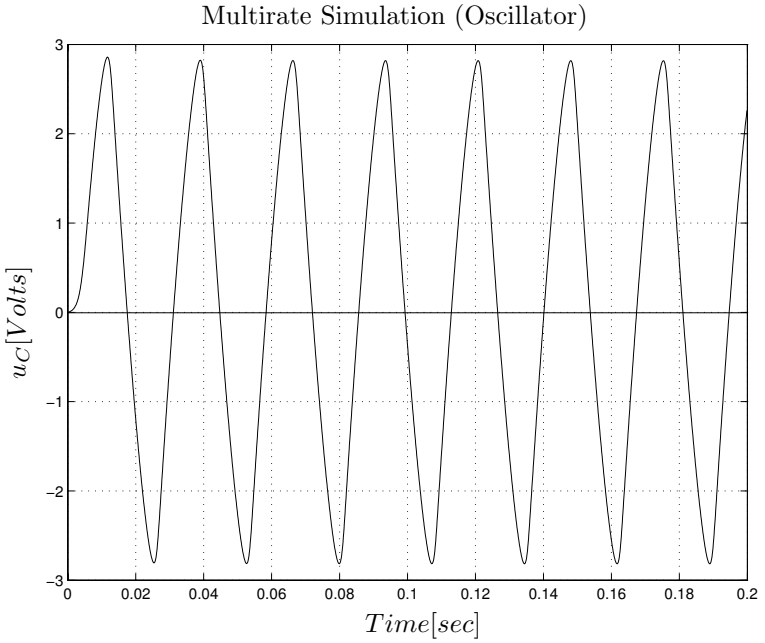

Figura 8.3: Voltaje del oscilador de Van–der–Pol.

Podemos generalizar esta idea para sistemas de la forma:

$$\dot{\mathbf{x}}_\mathbf{f}(t) = \mathbf{f}_\mathbf{f}(\mathbf{x}_\mathbf{f}, \mathbf{x}_\mathbf{s}, t) \tag{8.11a}$$

$$\dot{\mathbf{x}}_\mathbf{s}(t) = \mathbf{f}_\mathbf{s}(\mathbf{x}_\mathbf{f}, \mathbf{x}_\mathbf{s}, t) \tag{8.11b}$$

donde los sub–índices f y s corresponden a "rápido"(fast) y "lento" (slow) respectivamente.

Luego, el uso de la versión multitasa (multirate) de FE nos llevará a unas ecuaciones en diferencias de la forma:

$$\begin{aligned} \mathbf{x}_\mathbf{f}(t_i + (j+1)\cdot h) = {} & \mathbf{x}_\mathbf{f}(t_i + j\cdot h) + h \cdot \mathbf{f}_\mathbf{f}(\mathbf{x}_\mathbf{f}(t_i + j\cdot h), \\ & \mathbf{x}_\mathbf{s}(t_i + j\cdot h), t_i + j\cdot h) \end{aligned} \tag{8.12a}$$

$$\mathbf{x}_\mathbf{s}(t_i + k\cdot h) = \mathbf{x}_\mathbf{s}(t_i) + h \cdot \mathbf{f}_\mathbf{s}(\mathbf{x}_\mathbf{f}(t_i), \mathbf{x}_\mathbf{s}(t_i), t_i) \tag{8.12b}$$

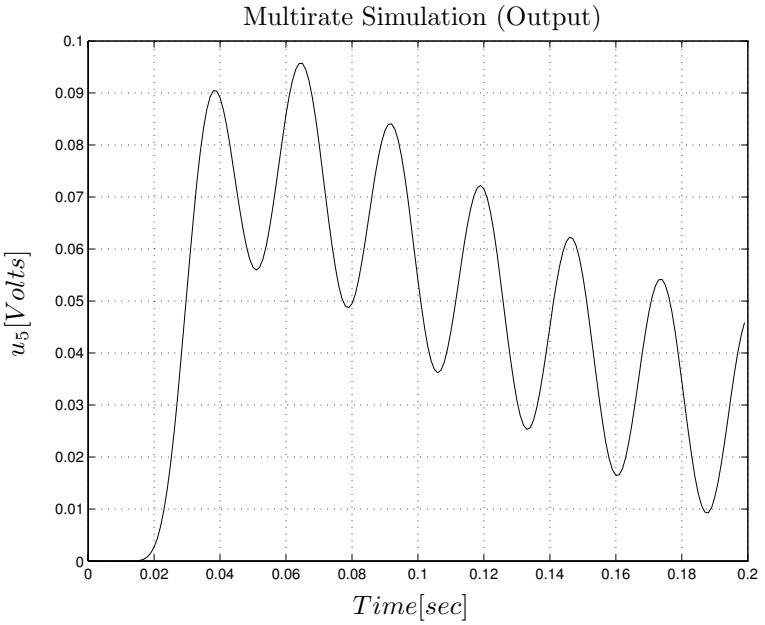

Figura 8.4: Voltaje a la salida de la línea de transmisión.

donde k es la razón (entera) entre ambos pasos, $j = 0 \ldots k-1$, y $h = t_{i+1} - t_i$ es el paso del subsistema lento.

Las Ecuaciones (8.12a–b) no especifican cómo calcular $\mathbf{x_s}(t_i + j \cdot h)$, ya que las variables del subsistema lento no se evalúan en los instantes intermedios.

En nuestro ejemplo, pusimos $\mathbf{x_s}(t_i + j \cdot h) = \mathbf{x_s}(t_i)$, es decir, utilizamos el último valor calculado.

A pesar de la mejora alcanzada, no debemos olvidar que el ejemplo no era muy rígido. Ya sabemos que los métodos explícitos no funcionarán con sistemas más stiff, porque nos obligarán a utilizar un paso excesivamente chico en el subsistema rápido.

En estos casos, como ya discutimos, los métodos semi–implícitos pueden ser una mejor opción. Sin embargo, también mencionamos que en los sistemas de gran dimensión tendremos problemas porque deberemos invertir una matriz muy grande.

Una solución que combina las dos ideas, de integración multitasa y semi–implícita, consiste en separar el sistema en un subsistema rápido y otro lento, aplicando un método explícito a la parte lenta y un método linealmente implícito a la parte rápida. Estos métodos se denominan algoritmos de integración de *modo mixto*.

8.6. Integración de Modo Mixto

La Figura 8.5 muestra el mismo circuito de la Fig. 8.2 con la inclusión de una carga RC adicional al final de la línea de transmisión.

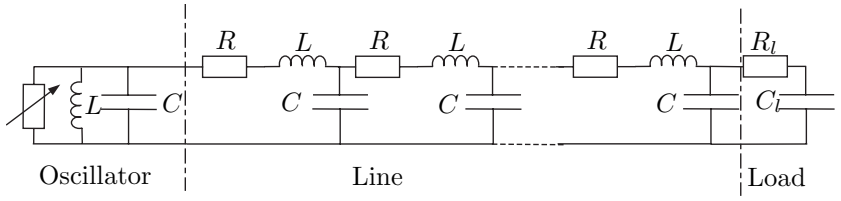

Figura 8.5: Oscilador de Van–der-Pol y línea de transmisión.

Las ecuaciones de estado son similares a las de antes, pero ahora también aparece el voltaje en el capacitor de la carga:

$$\frac{\mathrm{d}i_L}{\mathrm{d}t} = \frac{1}{L}u_C \tag{8.13a}$$
$$\frac{\mathrm{d}u_C}{\mathrm{d}t} = \frac{1}{C}(u_C - k\cdot u_C^3 - i_L - i_1) \tag{8.13b}$$
$$\frac{\mathrm{d}i_1}{\mathrm{d}t} = \frac{1}{L}u_C - \frac{R}{L}i_1 - \frac{1}{L}u_1 \tag{8.13c}$$
$$\frac{\mathrm{d}u_1}{\mathrm{d}t} = \frac{1}{C}i_1 - \frac{1}{C}i_2 \tag{8.13d}$$
$$\frac{\mathrm{d}i_2}{\mathrm{d}t} = \frac{1}{L}u_1 - \frac{R}{L}i_2 - \frac{1}{L}u_2 \tag{8.13e}$$
$$\frac{\mathrm{d}u_2}{\mathrm{d}t} = \frac{1}{C}i_2 - \frac{1}{C}i_3 \tag{8.13f}$$
$$\vdots$$
$$\frac{\mathrm{d}i_n}{\mathrm{d}t} = \frac{1}{L}u_{n-1} - \frac{R}{L}i_n - \frac{1}{L}u_n \tag{8.13g}$$
$$\frac{\mathrm{d}u_n}{\mathrm{d}t} = \frac{1}{C}i_n - \frac{1}{R_l\cdot C}(u_n - u_l) \tag{8.13h}$$
$$\frac{\mathrm{d}u_l}{\mathrm{d}t} = \frac{1}{R_l\cdot C_l}(u_n - u_l) \tag{8.13i}$$

Supongamos que los parámetros de la carga son $R_l = 1$ kΩ and $C_l = 1$ nF.

Ya que la resistencia de carga es mucho mayor que los de la linea, el nuevo término de la Ec.(8.13h) no va a influenciar la dinámica de la línea de transmisión en forma significativa, y podemos esperar que el sub–sistema (8.13a–h) exhiba un comportamiento similar al del Sistema (8.10).

Sin embargo, la última ecuación, Ec.(8.13i), introduce un autovalor rápido. La posición del mismo es

$$\lambda_l \approx -\frac{1}{R_l\cdot C_l} = -10^6 \text{ sec}^{-1} \tag{8.14}$$

sobre el eje real negativo del plano complejo.

Esto quiere decir que deberemos reducir el paso de integración por un factor de 1000 con respecto al ejemplo anterior para mantener la estabilidad numérica.

Desafortunadamente, esto es completamente inaceptable en el contexto de una simulación en tiempo real.

Una primer alternativa podría ser reemplazar el método de FE por el método de Euler semi–implícito. Sin embargo, estamos en presencia de un sistema de orden 13, y, dejando las supersticiones de lado, probablemente no tendremos tiempo para invertir una matriz de 13×13 en cada paso.

Si miramos un poco más cuidadosamente el sistema, veremos que no hay una gran interacción entre los subsistemas de las Ecs (8.13a–h) y de la Ec.(8.13i). De hecho, la dinámica rápida se puede explicar mirando a esta última ecuación sola.

Luego, podría ser razonable usar el método de Euler semi–implícito sólamente en la última ecuación.

Si hacemos esto, en la ecuación en diferencias resultante tendremos sólamente una ecuación implícita (lineal, ya que usamos el método semi–implícito), de la que deberemos despejar el valor de $u_l(t_{k+1})$. En definitiva, tendremos que calcular un Jacobiano de 1×1. Como además esta última ecuación es lineal en u_l, el Jacobiano será constante e igual a $-1/(R_l \cdot C_l)$. Por lo tanto, tampoco deberemos invertir ninguna matriz ni recalcular el Jacobiano en cada paso.

En definitiva, el costo será exactamente el mismo que el de usar FE con un paso de 1×10^{-3}... pero con la gran diferencia de que el resultado ahora será estable.

Tras simular este sistema utilizando el mismo enfoque que antes, es decir, usando un paso de 10^{-4} segundos en las ecuaciones del oscilador y un paso de 10^{-3} segundos en todas las otras ecuaciones, incluyendo la ecuación "semi–implícita" de la carga, obtuvimos los resultaos de la Figura 8.6.

En términos más generales, dado un sistema como el de las Ecs.(8.11a–b), el método mixto de Backward–Forward Euler estará dado por la siguiente fórmula:

$$\mathbf{x_s}(t_{k+1}) = \mathbf{x_s}(t_k) + h \cdot \mathbf{f_s}(\mathbf{x_f}(t_k), \mathbf{x_s}(t_k), t_k) \tag{8.15a}$$

$$\mathbf{x_f}(t_{k+1}) = \mathbf{x_f}(t_k) + h \cdot \mathbf{f_f}(\mathbf{x_f}(t_{k+1}), \mathbf{x_s}(t_{k+1}), t_{k+1}) \tag{8.15b}$$

Luego, el algoritmo comienza cada paso calculando explícitamente el valor de $\mathbf{x_s}(t_{k+1})$. Luego, utiliza este valor para calcular $\mathbf{x_f}(t_{k+1})$ en forma implícita (o mejor aún, semi–implícita).

La ventaja de resolver la ecuación implícita sólamente para las componentes $\mathbf{x_f}(t_{k+1})$ puede resultar muy importante en sistemas como el visto aquí, donde la longitud del vector $\mathbf{x_f}$ es considerablemente menor que la de $\mathbf{x_s}$.

En nuestro ejemplo (bastante académico), la reducción del número de cálculos es enorme.

Es importante observar que en realidad utilizamos una mezcla de simulación multi–tasa y de modo mixto, ya que usamos un paso de integración diez veces menor para las ecuaciones del oscilador.

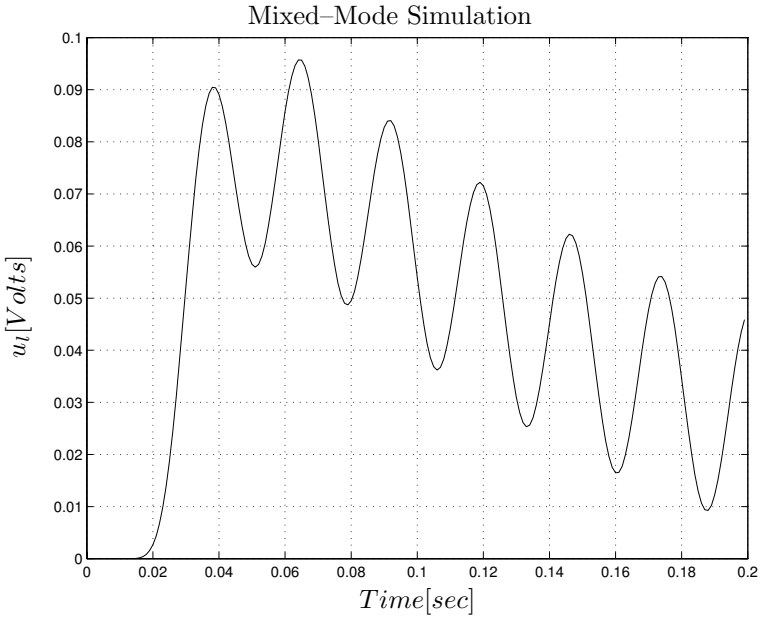

Figura 8.6: Voltaje de salida.

Tanto la integración multi–tasa como la de modo mixto suponen que existen dos (o más) subsistemas con dinámicas distintas que pueden distinguirse. Sin embargo, no siempre pasa esto. En el caso de las PDE parabólicas semidiscretizadas mediante el método de líneas, por ejemplo, no es posible realizar esta separación. Aquí los autovalores rápidos y lentos se encuentran distribuidos en todo el sistema.

8.7. Problemas Propuestos

[P8.1] Regla Trapezoidal Semi–implícita

Obtener una versión semi–implícita de la regla trapezoidal

$$\mathbf{x}(t_{k+1}) = \mathbf{x}(t_k) + \frac{h}{2}[\mathbf{f}(\mathbf{x}(t_k), t_k) + \mathbf{f}(\mathbf{x}(t_{k+1}), t_{k+1})] \tag{P8.1a}$$

Ayuda: Aproximar $\mathbf{f}(\mathbf{x}(t_{k+1}), t_{k+1})$ usando las ideas explicadas en la Sección 8.4.

Demostrar que el método es F–estable

[P8.2] Péndulo

Utilizando la fórmula trapezoidal semi–implícita, simular el modelo del péndulo:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\sin(x_1) - b \cdot x_2 \end{aligned}$$

suponiendo que el parámetro de fricción es $b = 0.02$.

Comenzar de la condición inicial $x_0 = (0.5\ 0.5)^T$ usando un paso de $h = 0.5$, y simular hasta $t_f = 500$.

Repetir el experimento utilizando la regla trapezoidal común.

Comparar los resultados y el número de evaluaciones de función requeridas por las dos simulaciones.

[P8.3] Péndulo sin Fricción

Repetir el problema P8.2 con $b = 0$ (o sea, sin fricción).

Comparar los resultados y explicar las diferencias.

[P8.4] Integración Multitasa

Los resultados de simulación mostrados en las Figs.8.3–8.4 tienen en realidad un error (había un pequeño error en el modelo que no fue advertido hasta la edición del libro).

Pedimos entonces implementar nuevamente la simulación, utilizando la idea expresada en las Ecs.(8.12a–b) y los parámetros indicados en la Sección 8.5.

Además, se pide repetir la simulación utilizando FE de manera normal, con $h = 1 \times 10^{-4}$ en todo el sistema y comparar los resultados y el tiempo que consume la simulación en cada caso.

[P8.5] Integración de Modo Mixto

Los resultados mostrados en la Fig.8.6 arrastran el error explicado en el problema anterior.

Pedimos también implementar nuevamente la simulación, utilizando la idea expresada en las Ecs.(8.15a–b), y los parámetros indicados en la Sección 8.6.

Además, se pide repetir la simulación utilizando BE de manera normal, con $h = 1 \times 10^{-4}$ en todo el sistema y comparar los resultados y el tiempo que consume la simulación en cada caso.

Capítulo 9

Simulación por Eventos Discretos

Este capítulo es una traducción resumida del capítulo 11 del libro *Continuous System Simulation* [2].

9.1. Introducción

En los capítulos anteriores estudiamos muchos métodos distintos para simular sistemas continuos. Pese a las diferencias (vimos métodos explícitos, implícitos, de paso fijo, de paso variable, monopaso, multipaso, etc.), todos los algoritmos vistos tienen algo en común: dado el instante t_{k+1}, se realiza una extrapolación polinomial para determinar el valor de las variables de estado en dicho instante.

Ahora veremos una idea completamente distinta. En lugar de preguntarnos que valor tendrán los estados en un determinado instante de tiempo, nos preguntaremos cuando una variable de estado se desviará de su valor actual en más de ΔQ. Es decir, buscaremos el menor paso h que hará que $x(t_k + h) = x(t_k) \pm \Delta Q$.

Obviamente, esta idea resultará en una estrategia de paso variable. Más aún, cuando $\mathbf{x}$ sea un vector, el valor resultante de h será distinto para cada componente del estado. Tendremos entonces dos posibilidades: o elegimos el menor de estos valores como paso central h, o podemos elegir diferentes valores de h_i para las diferentes componentes, lo que nos llevará a una simulación asincrónica, en la cual cada variable de estado tiene su propio tiempo de simulación.

La primera de las dos alternativas no sería nada más que una forma novedosa de controlar el paso de integración. La segunda alternativa, en cambio, es mucho más revolucionaria. A primera vista, el método resultante consistiría en una especie de combinación de simulación multi–tasa y de paso variable.

Hasta aquí, todos los métodos que vimos se podían describir mediante *ecuaciones en diferencias*. Tal representación no tendrá ningún sentido con esta nueva idea, donde cada componente evoluciona siguiendo sus propios valores de

h_i. En otras palabras, no tendremos más una aproximación en *tiempo discreto*, sino que, como veremos más adelante, obtendremos aproximaciones de *eventos discretos*

En este capítulo entonces, estudiaremos una nueva familia de métodos que resulta de esta idea de reemplazar la discretización temporal por una discretización del espacio de estado.

9.2. Discretización Espacial: Un Ejemplo Simple

Consideremos el sistema de primer orden:

$$\dot{x}_a(t) = -x_a(t) + 10 \cdot \varepsilon(t - 1.76) \tag{9.1a}$$

donde $\varepsilon(t)$ representa el escalón unitario, por lo que $\varepsilon(t - 1.76)$ es un escalón que tiene lugar en $t = 1.76$.

Supongamos que la tenemos la condición inicial

$$x_a(t_0 = 0) = 10 \tag{9.1b}$$

y veamos que pasa con este *sistema continuo*

$$\dot{x}(t) = -\text{floor}[x(t)] + 10 \cdot \varepsilon(t - 1.76) \tag{9.2a}$$

o:

$$\dot{x}(t) = -q(t) + 10 \cdot \varepsilon(t - 1.76) \tag{9.2b}$$

donde $q(t) \triangleq \text{floor}[x(t)]$ es la parte entera de la variable $x(t)$.

Aunque el sistema de la Ec. (9.2) es no lineal y discontinuo, podemos resolverlo fácilmente.

Cuando $0 < t < 1/9$, tenemos $q(t) = 9$ y $\dot{x}(t) = -9$. Durante este intervalo, $x(t)$ decrece linealmente desde 10.0 hasta 9.0. Luego, en el intervalo $1/9 < t < 1/9 + 1/8$, tenemos $q(t) = 8$ y $\dot{x}(t) = -8$. Aquí, $x(t)$ decrece linealmente desde 9.0 hasta 8.0.

Siguiendo con este análisis, encontramos que $x(t)$ alcanza el valor 3.0 en $t = 1.329$. Si no ocurriese ningún evento temporal, $x(t)$ alcanzaría el valor de 2.0 en $t = 1.829$. Sin embargo, cuando $t = 1.76$, cuando $x = 2.138$, la entrada cambia, y desde ese momento tendremos $\dot{x}(t) = 8$. La variable $x(t)$ incrementa su valor nuevamente de manera lineal con el tiempo, hasta que alcanza el valor de 3.0 en $t = 1.8678$.

De esta forma, $x(t)$ continua creciendo hasta que se llega a $x(t) = q(t) = 10$, a partir de donde la derivada $\dot{x}(t)$ se hace cero y el sistema no cambia su estado nunca más.

La Figura 9.1 muestra las trayectorias de $x(t)$ y $q(t)$.

Esta simulación se completó en 17 pasos y, despreciando algún problema de redondeo, obtuvimos la *solución exacta* de la Ec.(9.2). Todos los cálculos para obtener esta solución fueron triviales porque la derivada del estado permanece

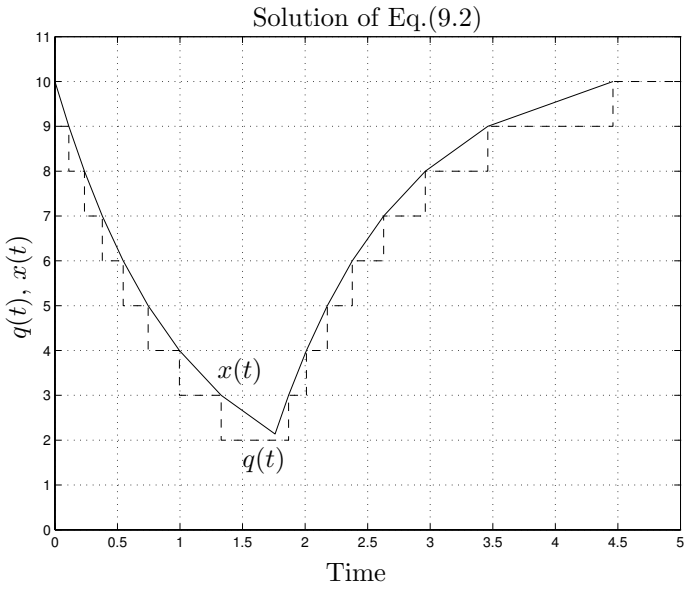

Figura 9.1: Trayectorias de las variables del sistema de la Ec.(9.2).

constante entre los instantes de los eventos, lo que nos permite calcular $x(t)$ de manera analítica.

La solución $x(t)$ y la del sistema original de la Ec.(9.1), $x_a(t)$, se comparan en la Fig.9.2.

Sin dudas, ambas soluciones son similares. Aparentemente, reemplazando $x(t)$ por $q(t) = \text{floor}[x(t)]$ en el lado derecho de la ecuación de una ecuación diferencial de primer orden encontramos un método explícito para simular dicha ecuación.

La primer pregunta que surge es si podemos generalizar esta idea para un sistema de orden n. Sin embargo, antes de poder contestar esta pregunta, necesitaremos analizar y estudiar la naturaleza discreta del sistema de la Ec.(9.2), para lo cual introduciremos algunas herramientas para su representación y simulación.

9.3. Sistemas de Eventos Discretos y DEVS

Todos los métodos vistos antes de este capítulo, aproximaban las ecuaciones diferenciales por sistemas de tiempo discreto (ecuaciones en diferencia), de la forma

$$\mathbf{x}(t_{k+1}) = \mathbf{f}(\mathbf{x}(t_k), t_k) \tag{9.3}$$

Con la nueva idea de cuantificar las variables de estado, sin embargo, estamos frente a sistemas que son discretos (ya que realizan un número finito de cambios),

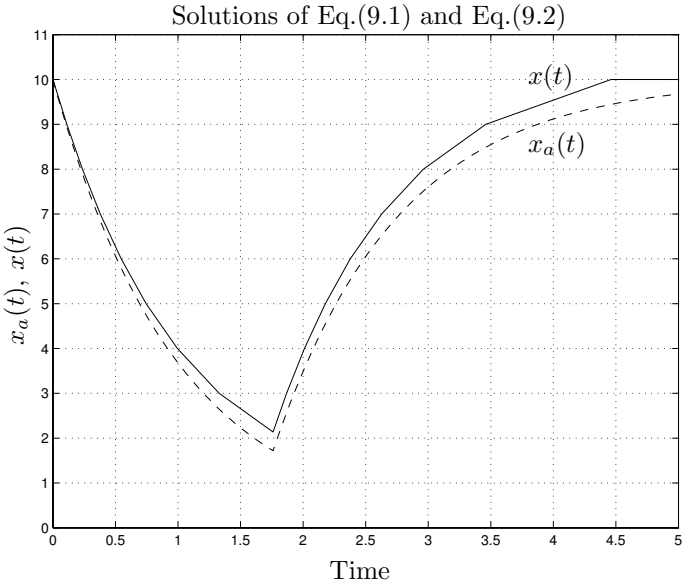

Figura 9.2: Trayectorias de los Sistemas (9.1) y (9.2).

pero no son de tiempo discreto.

Como veremos pronto, nuesta nueva manera de aproximar las ecuaciones nos lleva a *sistemas de eventos discretos*, dentro del formalismo DEVS.

DEVS, cuyas siglas provienen de *Discrete EVent System specification*, fue introducido por Bernard Zeigler a mediados de la década de 1970. DEVS permite representar todos los sistemas cuyo comportamiento entrada/salida pueda describirse mediante secuencias de eventos.

En este contexto, un *evento* es la representación de un cambio instantáneo en alguna parte del sistema. Como tal, puede caracterizarse mediante un valor y un tiempo de ocurrencia. El valor puede ser un número, un vector, una palabra, o en general, un elemento de algún conjunto.

La trayectoria definida por una secuencia de eventos toma el valor ϕ (o *No Event*) en casi todos los instantes de tiempo, excepto en los instantes en los que hay eventos. En estos instantes, la trayectoria toma el valor correspondiente al evento. La Figura 9.3 muestra una trayectoria de eventos que toma los valores x_2 en el tiempo t_1, luego toma el valor x_3 en t_2, etc.

Un modelo DEVS procesa una trayectoria de eventos y, de acuerdo a dicha trayectoria y a sus propias condiciones iniciales, provoca una trayectoria de eventos de salida. Este comportamiento entrada/salida se muestra en la Fig.9.4.

El comportamiento de un modelo DEVS *atómico* se explicita mediante distintas funciones y conjuntos, definidos en la siguiente estructura:

$$M = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta)$$

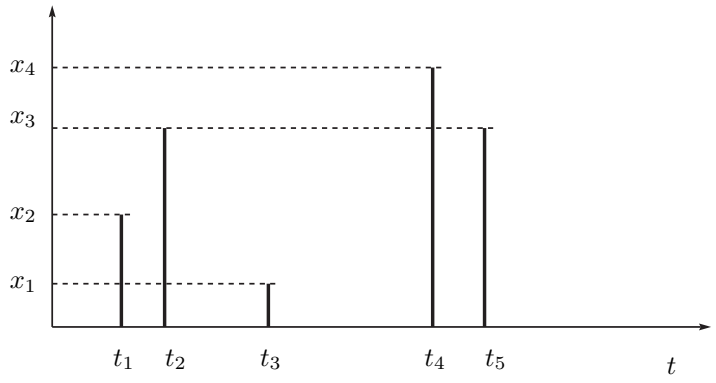

Figura 9.3: Trayectoria de eventos.

Figura 9.4: Comportamiento Entrada/Salida de un modelo DEVS.

donde:

- X es el conjunto de los valores de los eventos de entrada, es decir el conjunto de todos los valores que un evento de entrada puede tomar;
- Y es el conjunto de valores de los eventos de salida;
- S es el conjunto de los valores del estado;
- δ_{int}, δ_{ext}, λ y ta son funciones que definen la dinámica del sistema.

La Figura 9.5 nos muestra el comportamiento de un modelo DEVS.

Cada estado posible s ($s \in S$) tiene un *tiempo de avance* asociado calculado por la *función de avance de tiempo* $ta(s)$ ($ta(s) : S \rightarrow \Re_0^+$). El avance de tiempo es un número real no negativo, que determina cuanto tiempo debe permanecer el sistema en un estado en ausencia de eventos de entrada.

Luego, si el estado toma el valor s_1 en el instante t_1, tras $ta(s_1)$ unidades de tiempo (o sea, en el instante $t_1+ta(s_1)$), el sistema realiza una *transición interna*, alcanzando el nuevo estado s_2. El nuevo estado se calcula como $s_2 = \delta_{\text{int}}(s_1)$. La Función δ_{int} ($\delta_{\text{int}} : S \rightarrow S$) se denomina *función de transición interna*.

Cuando el estado cambia su valor desde s_1 a s_2, un evento de salida se produce con el valor $y_1 = \lambda(s_1)$. La función λ ($\lambda : S \rightarrow Y$) se denomina *función*

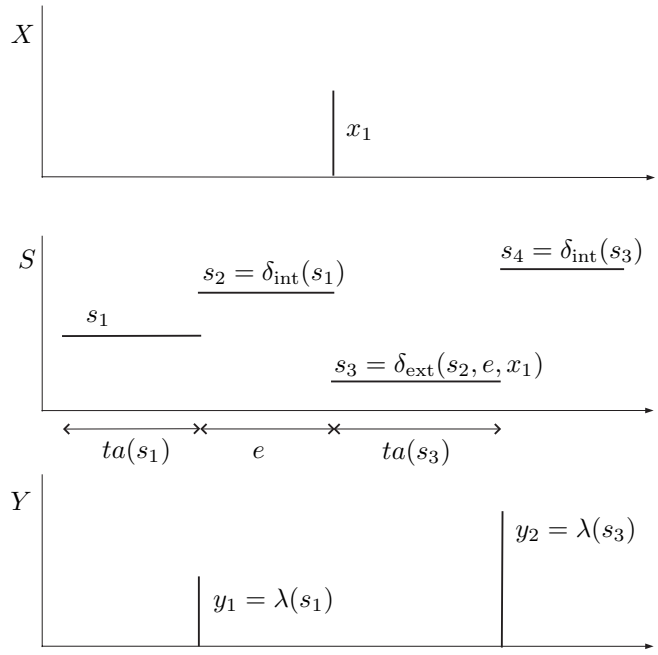

Figura 9.5: Trayectorias de un modelo DEVS.

de salida. De esta manera, las funciones ta, δ_{int} y λ definen el comportamiento autónomo de un modelo DEVS.

Cuando llega un evento de entrada, el estado cambia instantáneamente. El nuevo valor del estado depende no sólo del valor del evento de entrada, sino también del valor anterior del estado y del tiempo transcurrido desde la última transición. Si el sistema toma el valor s_2 en el instante t_2, y llega un evento en el instante $t_2 + e < ta(s_2)$ con valor x_1, el nuevo estado se calcula como $s_3 = \delta_{\text{ext}}(s_2, e, x_1)$. En este caso, decimos que el sistema realiza una *transición externa*. La función δ_{ext} ($\delta_{\text{ext}} : S \times \Re_0^+ \times X \rightarrow S$) se denomina *función de transición externa*. Durante la transición externa, no se produce ningún evento de salida.

En muchos casos, además, los conjuntos X e Y (donde toman los valores los eventos de entrada y salida) se forman por el producto cartesiano de un conjunto arbitrario (que contiene los valores de los eventos propiamente dichos) y un conjunto que contiene un número finito de *puertos* de entrada y salida respectivamente (luego veremos que estos puertos se utilizarán para acoplar modelos DEVS atómicos).

Consideremos por ejemplo un sistema que calcula una función estática $f(u_0, u_1)$, donde u_0 y u_1 son trayectorias seccionalmente constantes.

Podemos representar una trayectoria seccionalmente constante mediante una

secuencia de eventos si relacionamos cada evento con un cambio en el valor de la trayectoria. Utilizando esta idea, podemos construir el siguiente modelo DEVS atómico:

$$
\begin{aligned}
&M_F = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta), \text{ donde} \\
&X = \Re \times \{0, 1\} \\
&Y = \Re \times \{0\} \\
&S = \Re^2 \times \Re_0^+ \\
&\delta_{\text{int}}(s) = \delta_{\text{int}}(u_0, u_1, \sigma) = (u_0, u_1, \infty) \\
&\delta_{\text{ext}}(s, e, x) = \delta_{\text{ext}}(u_0, u_1, \sigma, e, x_v, p) = \tilde{s} \\
&\lambda(s) = \lambda(u_0, u_1, \sigma) = (f(u_0, u_1), 0) \\
&ta(s) = ta(u_0, u_1, \sigma) = \sigma
\end{aligned}
$$

con:

$$
\tilde{s} = \begin{cases} (x_v, u_1, 0) & \text{si } p = 0 \\ (u_0, x_v, 0) & \text{en otro caso} \end{cases}
$$

En este modelo, incluimos una variable σ en el estado s que coincide con la función ta. Esto se suele hacer siempre, ya que así se hace más facil la obtención de un modelo DEVS.

Como dijimos antes, cada evento de entrada y de salida incluye un número entero que indica el correspondiente puerto de entrada o salida. En los eventos de entrada, el puerto p puede ser 0 o 1 (es decir, hay dos puertos de entrada, uno para u_0y el otro para u_1). En los eventos de salida, hay un sólo puerto de salida (0).

En este caso, utilizamos números desde 0 para indicar los puertos. En principio podríamos haber utilizado palabras u otros objetos. Sin embargo, en este curso numeraremos siempre los puertos de esta forma ya que utilizaremos una herramienta de software que utiliza esta nomenclatura.

9.4. Modelos DEVS Acoplados

DEVS es un formalismo muy general, que puede utilizarse para describir sistemas muy complejos. Sin embargo, representar un sistema muy complejo utilizando funciones de transición y avance de tiempo es muy dificil ya que para describir dichas funciones debemos tener en cuenta todas las posibles situaciones en el sistema.

Los sistemas complejos generalmente se piensan como el acoplamiento de sistemas más simples. A través del acoplamiento, los eventos de salida de unos subsistemas se convierten en eventos de entrada de otros subsistemas. La teoría de DEVS garantiza que el modelo resultante de acoplar varios modelos DEVS atómicos es equivalente a un nuevo modelo DEVS atómico, es decir, DEVS es cerrado frente al acoplamiento (clausura del acoplamiento). Esto permite

el acoplamiento jerárquico de modelos DEVS, o sea, la utilización de modelos acoplados como si fueran modelos atómicos que a su vez pueden acoplarse con otros modelos atómicos o acoplados.

Si bien existen distintas maneras de acoplar modelos DEVS, en este curso utilizaremos siempre el acoplamiento mediante puertos de entrada/salida.

Cada *conexión interna* involucra un puerto de entrada y un puerto de salida de dos subsistemas. Además, en el contexto del acoplamiento jerárquico, también pueden existir conexiones desde los puertos de salida de un subsistema hacia los puertos de salida del sistema que lo contiene. Dichas conexiones se denominan *conexiones externas de salida*. Similarmente, hay conexiones desde los puertos de entrada de un sistema hacia los puertos de entrada de los subsistemas contenidos en dicho sistema. Tales conexiones se denominan *conexiones externas de entrada*.

La Figura 9.6 muestra un modelo DEVS acoplado N, resultado de acoplar los modelos M_a y M_b. En dicho modelo, el puerto de salida 1 de M_a se conecta al puerto de entrada 0 de M_b. Esta conexión puede representarse por el par ordenado $[(M_a, 1), (M_b, 0)]$. Otras conexiones son $[(M_b, 0), (M_a, 0)]$, $[(N, 0), (M_a, 0)]$, $[(M_b, 0), (N, 1)]$, etc. De acuerdo a la propiedad de clausura de DEVS, el modelo N puede usarse de la misma forma que si fuera un modelo atómico, y puede ser acoplado con otros modelos atómicos y/o acoplados.

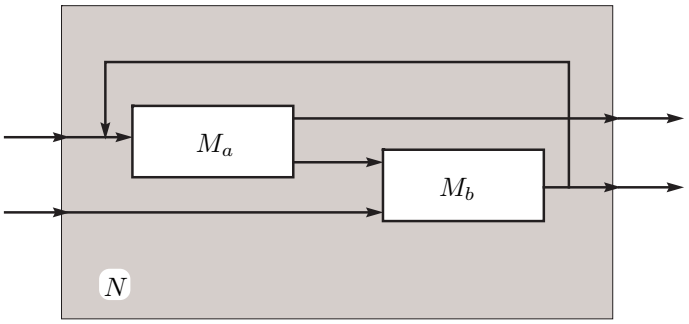

Figura 9.6: Modelo DEVS acoplado.

Volviendo al ejemplo anterior donde describimos el modelo DEVS M_F de una función estática, haciendo uso del acoplamiento podemos reutilizar dicho sistema para formar un modelo que calcule la función $f_2(u_0, u_1, u_2) = f[f(u_0, u_1), u_2]$. Para esto, podemos acoplar dos modelos de la clase M_F. Si llamamos A y B a dichos subsistemas, y N al sistema acoplado resultante, el acoplamiento correspondiente estaría dado por las conexiones: $[(A, 0), (B, 0)]$, $[(N, 0), (A, 0)]$, $[(N, 1), (A, 1)]$, $[(N, 2), (B, 1)]$, y $[(B, 0), (N, 0)]$.

En general, más que describir las conexiones de esta forma, lo que haremos será dibujarlas en forma de *Diagrama de Bloques* similares al mostrado en la Fig.9.6.

9.5. Simulación de Modelos DEVS

Una de las características más importantes de DEVS es su facilidad para implementar simulaciones. Un modelo DEVS puede simularse con un programa ad–hoc escrito en cualquier lenguaje. De hecho, la simulación de un modelo DEVS no es mucho más complicada que la de un modelo de tiempo discreto.

Un algoritmo básico que puede utilizarse para simular un modelo DEVS acoplado puede describirse por los siguientes pasos:

1. Identificamos el modelo atómico que, de acuerdo a su tiempo de avance y al tiempo transcurrido, deba ser el próximo en realizar la transición interna. Llamamos d^* a este sistema y t_n al tiempo de dicha transición.

2. Avanzamos el reloj de la simulación t a $t = t_n$, y ejecutamos la función de transición interna del modelo d^*.

3. Propagamos el evento de salida provocado por d^* a todos los modelos atómicos conectados al puerto de salida y ejecutamos las funciones de transición externas correspondientes. Luego, volvemos al paso 1.

Una de las maneras más simples de implementar estos pasos es escribiendo un programa con una estructura jerárquica equivalente a la del modelo a ser simulado.

Aunque, como ya dijimos, la implementación de una simulación de DEVS es muy simple, en general los modelos que se utilizan en la práctica están compuestos por muchos subsistemas y, hacer un programa ad–hoc de todos estos modelos suele tornarse muy trabajoso.

Por esto, hay disponibles varias herramientas de software para la simulación de modelos DEVS que simplifican mucho la tarea de modelado y de simulación.

En este curso, utilizaremos PowerDEVS. Este software, a pesar de ser un simulador de DEVS de propósito general, fue concebido especialmente para facilitar la simulación de sistemas continuos.

PowerDEVS tiene una interface gráfica que permite crear modelos DEVS acoplados con las herramientas típicas de drag and drop. Cuenta también con librerías que tienen varios modelos atómicos ya definidos (muchos de ellos para realizar simulaciones con métodos de integración que veremos luego).

Además, los modelos atómicos pueden crearse y modificarse fácilmente utilizando el *editor de modelos atómicos*, donde el usuario sólo debe definir las funciones de transición, de salida y de avance de tiempo utilizando sintaxix C++.

Lo que hace PowerDEVS para ejecutar una simulación es *armar* y compilar un programa en C++, utilizando para esto la información del acoplamiento del modelo gráfico, el código C++ contenido en cada modelo atómico y ciertas rutinas preestablecidas (también programadas en C++) encargadas de llevar a cabo la simulación.

9.6. DEVS y Simulación de Sistemas Continuos

En el primer ejemplo de la Sección 9.3, vimos que las trayectorias seccionalmente constantes podían representarse mediante secuencias de eventos. Esta simple idea constituye en realidad la base del uso de DEVS en la simulación de sistemas continuos.

En dicho ejemplo, vimos que un modelo DEVS puee representar el comportamiento de una función estática con una entrada seccionalmente constante. En los sistemas continuos las trayectorias no tienen esta forma, pero podríamos alterarlos para que las trayectorias sean así. De hecho, esto es lo que hicimos con el sistema de la Ec.(9.1), donde utilizamos la función "floor" para convertirlo en el sistema de la Ec.(9.2).

Volviendo a este ejemplo, podemos dividir la Ec.(9.2) de la siguiente forma:

$$\dot{x}(t) = d_x(t) \tag{9.4a}$$
$$q(t) = \text{floor}[x(t)] \tag{9.4b}$$

y:

$$d_x(t) = -q(t) + u(t) \tag{9.5}$$

donde $u(t) = 10 \cdot \varepsilon(t - 1.76)$.

Podemos representar este sistema usando el Diagrama de Bloques de la Fig.9.7.

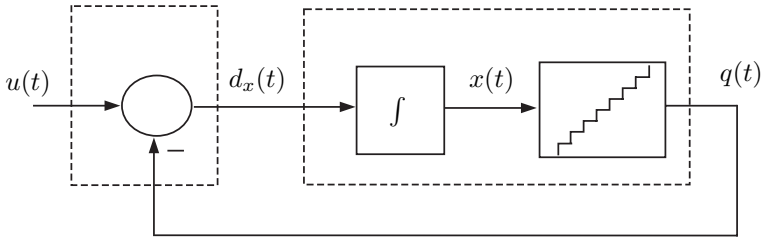

Figura 9.7: Diagrama de Bloques de las Ecs.(9.4–9.5).

Como dijimos antes, el subsistema de la Ec.(9.5) –siendo una funcion estática– puede representarse por el modelo DEVS M_F de la Sección 9.3.

El subsistema de la Ec.(9.4) is un sistema dinámico que tiene una entrada seccionalmente constante $d_x(t)$ y una salida seccionalmente constante $q(t)$. Podemos representar este sistema exactamente utilizando el siguiente modelo DEVS:

$$
\begin{aligned}
&M_{QI} = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta), \text{ donde} \\
&X = Y = \Re \times \mathbb{N} \\
&S = \Re^2 \times \mathbb{Z} \times \Re_0^+ \\
&\delta_{\text{int}}(s) = \delta_{\text{int}}(x, d_x, q, \sigma) = (x + \sigma \cdot d_x, d_x, q + \text{sign}(d_x), \frac{1}{|d_x|}) \\
&\delta_{\text{ext}}(s, e, x) = \delta_{\text{ext}}(x, d_x, q, \sigma, e, x_v, p) = (x + e \cdot d_x, x_v, q, \tilde{\sigma}) \\
&\lambda(s) = \lambda(x, d_x, q, \sigma) = (q + \text{sign}(d_x), 0) \\
&ta(s) = ta(x, d_x, q, \sigma) = \sigma
\end{aligned}
$$

con:

$$
\tilde{\sigma} = \begin{cases} \frac{q + 1 - x}{x_v} & \text{si } x_v > 0 \\ \frac{q - x}{x_v} & \text{si } x_v < 0 \\ \infty & \text{en otro caso} \end{cases}
$$

Ahora, si queremos simular el sistema de las Ecs.(9.4–9.5) con PowerDEVS, podemos traducir los modelos DEVS atómicos M_F y M_{QI}, en modelos atómicos PowerDEVS.

Un modelo atómico de PowerDEVS correspondiente a M_F tendrá, en el editor de modelos atómicos, la siguiente forma:

ATOMIC MODEL STATIC1
State Variables and Parameters:

```
double u[2],sigma; //states
double y; //output
double inf; //parameters
```

Init Function:

```
inf = 1e10;
u[0] = 0;
u[1] = 0;
sigma = inf;
y = 0;
```

Time Advance Function:

```
return sigma;
```

Internal Transition Function:

```
sigma = inf;
```

External Transition Function:

```
double xv;
xv = *(double*)(x.value);
u[x.port] = xv;
```

$sigma = 0$;

Output Function:

$y = u[0] - u[1]$;
return Event(&y,0);

La conversión del modelo DEVS M_F en el correspondiente modelo de PowerDEVS STATIC1 es trivial.

De manera similar, el modelo DEVS M_{QI} puede representarse en PowerDEVS como:

ATOMIC MODEL NHINTEGRATOR

State Variables and Parameters:

double X, dX, q, $sigma$; //states
//we use capital X because x is reserved
double y; //output
double inf; //parameters

Init Function:

va_list $parameters$;
va_start($parameters$, t);
X = va_arg($parameters$, double);
$dX = 0$;
q =floor(X);
$inf = 1e10$;
$sigma = 0$;
$y = 0$;

Time Advance Function:

return $sigma$;

Internal Transition Function:

```
X = X + sigma * dX;
if (dX > 0) {
   sigma = 1/dX;
   q = q + 1;
   }
else {
   if (dX < 0) {
      sigma = -1/dX;
      q = q - 1;
      }
   else {
      sigma = inf;
   };
};
```

External Transition Function:

```
double xv;
xv = *(double*)(x.value);
X = X + dX * e;
if (xv > 0) {
    sigma = (q + 1 - X)/xv;
    }
else {
    if (xv < 0) {
        sigma = (q - X)/xv;
        }
    else {
        sigma = inf;
    };
};
dX = xv;
```

Output Function:

```
if (dX == 0) {y = q;} else {y = q + dX/fabs(dX);};
return Event(&y,0);
```

Nuevamente, la traducción es directa. Sin embargo, en este último caso agregamos algunos nuevos elementos a la función de inicialización. Las dos primeras líneas las incluye automáticamente el editor de modelos atómicos cuando se edita un nuevo modelo. Estas declaran la variable *parameters*, donde la interface gráfica coloca los parámetros del modelo. En nuestro caso, definimos la condición inicial en la variable X como un parámetro. Luego, la tercera linea de la función de inicialización símplemente toma el primer parámetro del bloque y lo coloca en X.

De esta manera, en la interface gráfica, podemos símplemente hacer doble click en el bloque y cambiar el valor de dicho parámetro (o sea, podemos cambiar la condición inicial sin cambiar la definición del modelo atómico).

El otro cambio con respecto al modelo M_F tiene también que ver con la inclusión de las condiciones iniciales. Al principio de la simulación, forzamos al modelo a provocar un evento, de manera tal que el valor inicial de la variable cuantificada correspondiente q sea conocida por el resto del sistema.

El modelo PowerDEVS generado utilizando el modelo gráfico del editor se muestra en la Fig.9.8.

En este modelo, además de los modelos atómicos STATIC1 y NHINTEGRATOR, incluimos tres bloques más: un bloque STEP que produce un evento con valor 10 en el instante $t = 1.76$, y dos modelos adicionales que graban y muestran los resultados de simulación. Estos últimos tres modelos están incluidos en las librerías estándares de la distribución de PowerDEVS.

Utilizando el modelo PowerDEVS de la Fig.9.8, obtuvimos los datos mostrados en la Fig.9.1. La gráfica de dicha figura fue generada por MATLAB, utilizando los datos grabados en un archivo por el bloque “iss2dsk”.

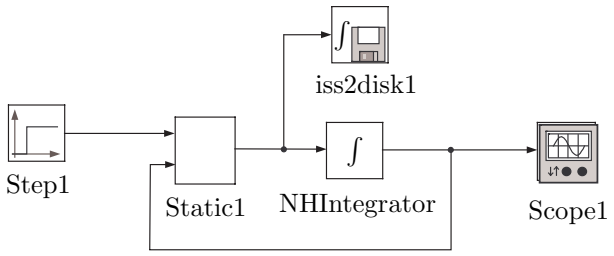

Figura 9.8: Modelo PowerDEVS Acoplado de las Eqs.(9.4–9.5).

El subsistema de la Ec.(9.4) corresponde al integrador junto con el bloque en forma de escalera del diagrama de la Fig.9.7. Dicho subsistema es equivalente al modelo DEVS M_{QI} representado por el modelo NHINTEGRATOR en PowerDEVS.

Esto es lo que Zeilger llamó *integrador cuantificado*. Aquí, la función "floor" actúa como una *función de cuantificación*. En general, una función de cuantificación mapea un dominio continuo de números reales en un conjunto discreto de los reales.

Un sistema que relacione su entrada y su salida mediante cualquier tipo de función de cuantificación será entonces llamado *cuantificador*. Luego, nuestro bloque con forma de escalera es un caso particular de cuatificador con cuantificación uniforme.

Un integrador cuantificado es entonces un integrador concatenado con un cuantificador.

De manera similar, el modelo M_F representa una función estática con su vector de entrada en $\Re^2$. El modelo STATIC1 de PowerDEVS implementa un caso particular de dicha función estática, esto es, la función $f(u_0, u_1) = u_0 - u_1$. Un modelo DEVS para una función estática genérica con su vector de entrada en $\Re^n$ puede fácilmente construirse y programarse en PowerDEVS.

Asímismo, podemos obtener un modelo DEVS que represente un integrador cuantificado general (con cualquier función de cuantificación).

Luego, si tenemos un sistema general invariante en el tiempo[1]:

$$\begin{array}{rcl} \dot{x}_{a_1} & = & f_1(x_{a_1}, x_{a_2}, \cdots, x_{a_n}, u_1, \cdots, u_m) \\ \dot{x}_{a_2} & = & f_2(x_{a_1}, x_{a_2}, \cdots, x_{a_n}, u_1, \cdots, u_m) \\ & \vdots & \\ \dot{x}_{a_n} & = & f_n(x_{a_1}, x_{a_2}, \cdots, x_{a_n}, u_1, \cdots, u_m) \end{array} \tag{9.6}$$

con trayectorias de entrada seccionalmente constantes $u_j(t)$, podemos transfor-

[1] Utilizaremos x_a para referirnos a las variables de estado del sistema original, tal que $x_a(t)$ sea la solución analítica.

marlo en:

$$\begin{array}{rcl} \dot{x}_1 & = & f_1(q_1, q_2, \cdots, q_n, u_1, \cdots, u_m) \\ \dot{x}_2 & = & f_2(q_1, q_2, \cdots, q_n, u_1, \cdots, u_m) \\ & \vdots & \\ \dot{x}_n & = & f_n(q_1, q_2, \cdots, q_n, u_1, \cdots, u_m) \end{array} \tag{9.7}$$

donde $q_i(t)$ se relaciona con $x_i(t)$ mediante alguna función de cuantificación.

Las variables q_i se llaman *variables cuantificadas*. Este sistema de ecuaciones puede representarse mediante el diagrama de bloques de la Fig.9.9, donde **q** y **u** son los vectores formados por las variables cuantificadas y por las variables de entrada respectivamente.

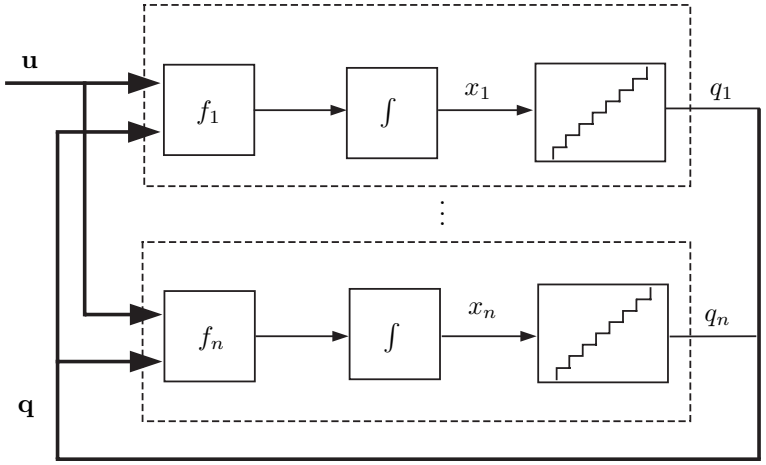

Figura 9.9: Diagrama de Bloques de la Ec.(9.7).

Cada subsistema en la Fig.9.9 puede representarse exactamente por un modelo DEVS, ya que todos los subsistemas son funciones estáticas o integradores cuantificados. Estos modelos DEVS pueden acoplarse, y, debido a la clausura bajo acoplamiento, el sistema completo formará también un modelo DEVS.

Entonces, cuando un sistema se modifica agregando cuantificadores en la salida de todos los integradores, el sistema resultante es equivalente a un modelo DEVS acoplado que puede simularse, siempre y cuando todas las entradas sean seccionalmente constantes.

Esta idea constituye la primer aproximación a un método para simular sistemas continuos mediante eventos discretos.

Desafortunadamente, esta idea no funciona ....

Este método nos llevará en la mayor parte de los casos a un modelo DEVS *ilegítimo*, es decir, un modelo que realiza un número infinito de transiciones en un intervalo acotado de tiempo. De tal forma, la simulación se trabará luego de cierto tiempo.

Podemos observar este problema en el sistema de las Ecs.(9.4–9.5) símplemente cambiando la entrada a $u(t) = 10.5 \cdot \varepsilon(t - 1.76)$ (en realidad el problema

aparece utilizando cualquier valor constante no entero de $u(t)$). Las trayectorias hasta $t = 1.76$ son las mismas que antes (Fig.9.1). Tras aplicar el escalor, las trayectorias crecen un poco más rápido que antes, pero cuando $x(t) = q(t) = 10$, la derivada del estado ya no se hace 0. En lugar de eso, continúa creociendo con una pendiente $\dot{x}(t) = 0.5$. Luego, tras 2 unidades de tiempo, tenemos $x(t) = q(t) = 11$. Entonces, la pendiente se torna negativa. y $x(t)$ decrece con una pendiente $\dot{x}(t) = -0.5$. Luego, $q(t)$ inmediatamente retorna a 10, la derivada del estado se torna positiva nuevamente y $x(t)$ comienza a crecer de nuevo. Todo esto ocurre en realidad en tiempo 0, por lo que la frecuencia de las oscilaciones es infinita.

Como veremos en a continuación, este problema se puede resolver con una simple modificación en la estrategia de cuantificación. Esta nueva estrategia es lo que da lugar al primer método numérico de cuantificación, denominado método de los *sistemas de estados cuantificados*, y abreviado como método de QSS (por *Quantized State Systems*).

9.7. Método de los Sistemas de Estados Cuantificados (QSS)

Si analizamos las oscilaciones infinitamente rápidas en el sistema de las Ecs.(9.4–9.5), podemos ver que las mismas se deben a los cambios en $q(t)$. Un cambio infinitesimal en $x(t)$ puede producir, debido a la cuantificación, una oscilación importante con una frecuencia infinita en $q(t)$.

Una solución a esto es utilizar histéresis en la cuantificación. Agregando histéresis a la relación entre $x(t)$ y $q(t)$, las oscilaciones en esta última pueden ser sólo debidas a oscilaciones grandes en $x(t)$. Si la derivada $\dot{x}(t)$ es finita, una oscilación grande en $x(t)$ no puede ocurrir instantaneamente sino que tiene una frecuencia máxima acotada.

Entonces, antes de presentar formalmente el método de QSS, definiremos el concepto de *función de cuantificación con histéresis*.

Supongamos que $x(t) : \Re \to \Re$ es una trayectoria continua en el tiempo, y supongamos que $q(t) : \Re \to \Re$ es una trayectoria seccionalmente constante. Luego, diremos que $x(t)$ y $q(t)$ se relacionan mediante una función de cuantificación con histéresis uniforme si se cumple que:

$$q(t) = \begin{cases} \text{floor}[x(t_0)/\Delta Q] \cdot \Delta Q & \text{si } t = t_0 \\ x(t) & \text{si } |q(t^-) - x(t)| \geq \Delta Q \\ q(t^-) & \text{en otro caso} \end{cases} \tag{9.8}$$

El parámetro de la cuantificación ΔQ se denomina *quantum*.

Las Figuras 9.10–9.11 muestran una función de cuantificación con histéresis y dos variables relacionadas mediante dicha función

Ahora, podemos definir el método de QSS:

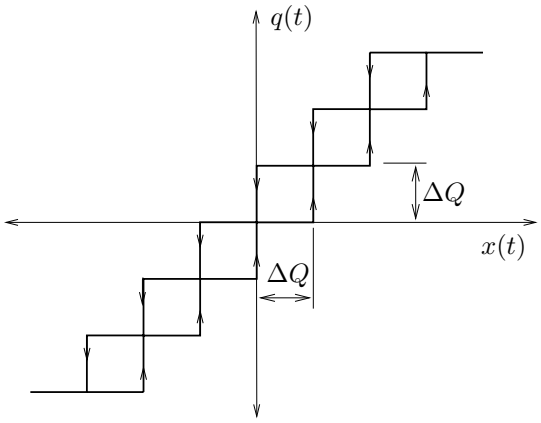

Figura 9.10: Función de Cuantificación con Histéresis.

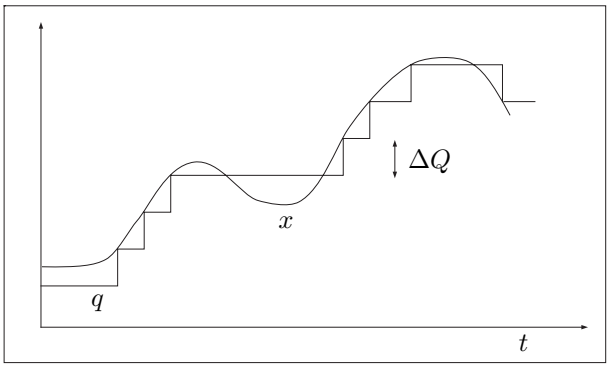

Figura 9.11: Variables Relacionadas con una Función de Cuantificación con Histéresis.

> Dado un sistema como el de la Ec.(9.6), el método de QSS lo transforma en un *sistema de estados cuantificados* de la forma de la Ec.(9.7), donde cada par de variables $x_i(t)$ y $q_i(t)$ se relaciona mediante funciones de cuantificación con histéresis.

Es decir, para utilizar el método de QSS símplemente deberemos elegir el *quantum* a utilizar ΔQ_i para cada variable de estado.

Se puede demostrar que las variables cuantificadas y de estado son siempre seccionalmente constantes y seccionalmente lineales respectivamente. Por esto, la simulación con QSS no puede trabarse como ocurría con la cuantificación sin histéresis.

Para llevar a cabo la simulación, al igual que antes, podemos construir un modelo DEVS acoplando los subsistemas correspondientes a las funciones estáticas y a los *integradores cuantificados con histéresis*.

Un integrador cuantificado con histéresis es un integrador cuantificado donde la función de cuantificación sin memoria es reemplazada por una con histéresis. Esto es equivalente a cambiar la Ec.(9.4b) por la Ec.(9.8) en el sistema de la Ec.(9.4).

Con esta modificación, el integrador cuantificado con histéresis de la Ec.(9.4a) y la Ec.(9.8) puede representarse por el modelo DEVS:

$$
\begin{aligned}
) \quad & M_{HQI} = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta), \text{ donde} \\
& X = Y = \Re \times \{0\} \\
& S = \Re^3 \times \Re_0^+ \\
& \delta_{\text{int}}(s) = \delta_{\text{int}}(x, d_x, q, \sigma) = (x + \sigma \cdot d_x, d_x, x + \sigma \cdot d_x, \sigma_1) \\
& \delta_{\text{ext}}(s, e, x_u) = \delta_{\text{ext}}(x, d_x, q, \sigma, e, x_v, p) = (x + e \cdot d_x, x_v, q, \sigma_2) \\
& \lambda(s) = \lambda(x, d_x, q, \sigma) = (x + \sigma \cdot d_x, 0) \\
& ta(s) = ta(x, d_x, q, \sigma) = \sigma
\end{aligned}
$$

con:

$$\sigma_1 = \frac{\Delta Q}{|d_x|}$$

y:

$$\sigma_2 = \begin{cases} \dfrac{q + \Delta Q - (x + e \cdot d_x)}{x_v} & \text{if } x_v > 0 \\ \dfrac{(x + e \cdot d_x) - (q - \Delta Q)}{|x_v|} & \text{if } x_v < 0 \\ \infty & \text{si } x_v = 0 \end{cases}$$

Como dijimos antes, el método de QSS consiste en elegir el quantum en cada variable de estado. Esto define automáticamente el modelo DEVS M_{HQI}. Representando las funciones estáticas $f_1, \ldots, f_n$ con modelos DEVS similares a los de M_F y acoplándolos, el sistema de la Ec.(9.7) puede simularse exactamente (ignorando los problemas de redondeo).

La Figura 9.12 nos muestra una representación de un QSS genérico.

El integrador cuantificado con histéresis M_{HQI} puede implementarse en PowerDEVS como sigue:

ATOMIC MODEL HINTEGRATOR
State Variables and Parameters:
double X, dX, q, $sigma$; //states
double y; //output
double dq, inf; //parameters

Init Function:
va_list $parameters$;
va_start($parameters$, t);

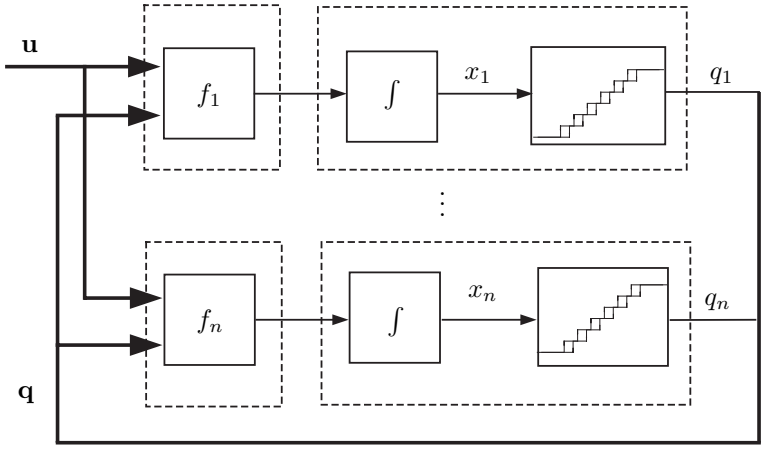

Figura 9.12: Diagrama de Bloques de un QSS.

$dq =$ va_arg($parameters$, double);
$X =$ va_arg($parameters$, double);
$dX = 0$;
$q =$floor$(X/dq) * dq$;
$inf = 1e10$;
$sigma = 0$;

Time Advance Function:
return $sigma$;

Internal Transition Function:
$X = X + sigma * dX$;
if $(dX > 0)$ {
$sigma = dq/dX$;
$q = q + dq$;
}
else {
if $(dX < 0)$ {
$sigma = -dq/dX$;
$q = q - dq$;
}
else {
$sigma = inf$;
};
};

External Transition Function:
double xv;
$xv =$ *(double*)$(x.value)$;
$X = X + dX * e$;

```
if (xv > 0) {
   sigma = (q + dq − X)/xv;
   }
else {
   if (xv < 0) {
      sigma = (q − dq − X)/xv;
      }
   else {
      sigma = inf;
   };
};
dX = xv;
```

Output Function:

```
if (dX == 0) {y = q;} else {y = q + dq ∗ dX/fabs(dX);};
return Event(&y,0);
```

A continuación, veremos como funciona el método con un ejemplo muy simple, lineal, y de segundo orden:

$$\begin{aligned} \dot{x}_{a_1}(t) &= x_{a_2}(t) \\ \dot{x}_{a_2}(t) &= 1 - x_{a_1}(t) - x_{a_2}(t) \end{aligned} \tag{9.9}$$

con condiciones iniciales:

$$x_{a_1}(0) = 0, \quad x_{a_2}(0) = 0 \tag{9.10}$$

Utilizaremos un quantum $\Delta Q = 0.05$ en ambas variables. Luego, el QSS resultante es:

$$\begin{aligned} \dot{x_1}(t) &= q_2(t) \\ \dot{x_2}(t) &= 1 - q_1(t) - q_2(t) \end{aligned} \tag{9.11}$$

Este sistema puede simularse utilizando un modelo DEVS acoplado, compuesto por dos modelos atómicos del tipo M_{HQI} (integradores cuantificados) y dos modelos atómicos similares a M_F que calculen las funciones estáticas $f_1(q_1, q_2) = q_2$ y $f_2(q_1, q_2) = 1 - q_1 - q_2$. La Figura 9.13 representa el sistema acoplado.

En realidad, como f_1 no depende de q_1, esta conexión no es necesaria. Más aún, como $f_1(q_1, q_2) = q_2$ el subsistema F_1 puede reemplazarse por una conexión directa desde QI_2 hacia QI_1. Estas simplificaciones pueden reducir considerablemente el costo computacional de la simulación.

De hecho, al dibujar el diagrama de bloques en PowerDEVS, automáticamente hacemos estas simplificaciones (ver Fig.9.14).

Los resultados de la simulación se muestran en la Fig.9.15. Esta simulación se completó tras 30 transiciones internas en cada integrador cuantificado, lo que da un total de 60 pasos. En la Fig.9.15 pueden verse las trayectorias seccionalmente lineales de $x_1(t)$ y $x_2(t)$, y las trayectorias seccionalmente constantes de $q_1(t)$ y $q_2(t)$.

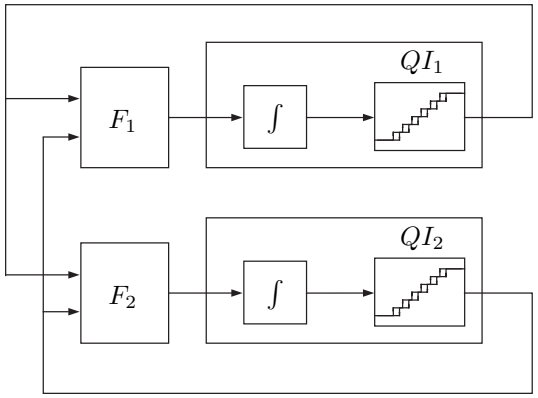

Figura 9.13: Diagrama de Bloques de la Ec.(9.11).

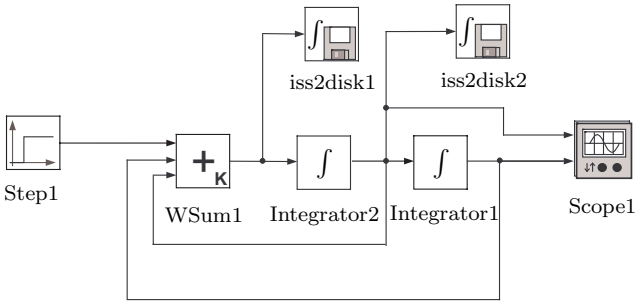

Figura 9.14: Modelo de PowerDEVS.

La presencia de la histéresis se puede notar cuando las pendientes de las variables de estado cambian de signo, ya que cerca de estos puntos tenemos distintos valores de q para el mismo valor de x.

Las simplificaciones que mencionamos respecto a las conexiones pueden aplicarse a sistemas generales, donde muchas de las funciones pueden depender de sólo algunas variables de estado. De esta manera, QSS puede explotar las propiedades estructurales de los sistemas para reducir el costo computacional. Cuando el sistema es *ralo*, las simulaciones suelen ser muy eficientes porque cada paso involucra cálculos en pocos integradores y en pocas funciones.

Los métodos de tiempo discreto (que vimos en los capítulos anteriores) también pueden explotar las propiedades de raleza. Sin embargo, para esto, hay que aplicar técnicas específicas de manejo de matrices ralas. En el caso de QSS, el aprovechamiento de la raleza es intrínseco al método.

9.8. Problemas Propuestos

[**P9.1**] **Aquiles y la Tortuga** Considerar el sistema de segundo orden:

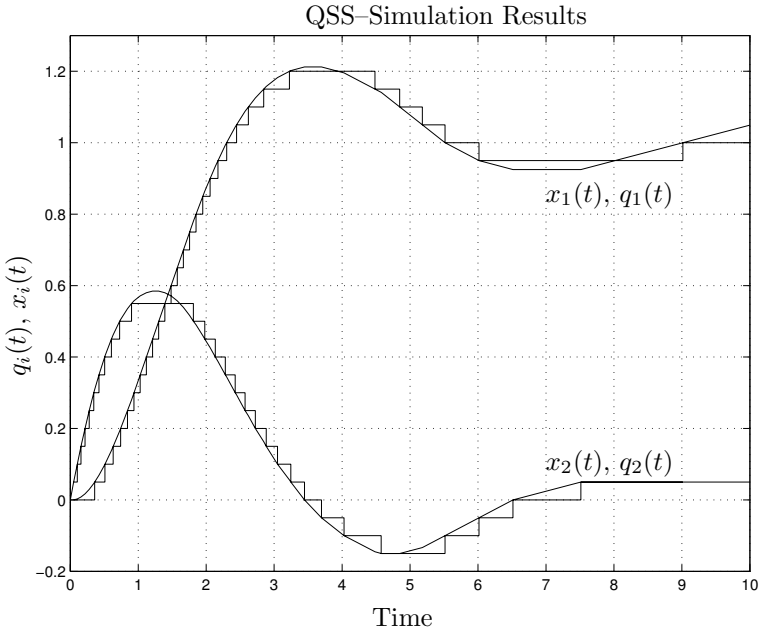

Figura 9.15: Trayectorias del sistema de la Ec.(9.11).

$$\begin{aligned} \dot{x}_{a_1} &= -0.5 \cdot x_{a_1} + 1.5 \cdot x_{a_2} \\ \dot{x}_{a_2} &= -x_{a_1} \end{aligned} \tag{P9.1a}$$

Aplicar la función de cuantificación sin memoria:

$$q_i = 2 \cdot \text{floor}(\frac{x_i - 1}{2}) + 1 \tag{P9.1b}$$

en ambas variables de estado, y estudiar las soluciones del sistema cuantificado:

$$\begin{aligned} \dot{x}_1 &= -0.5 \cdot q_1 + 1.5 \cdot q_2 \\ \dot{x}_2 &= -q_1 \end{aligned} \tag{P9.1c}$$

desde la condición inicial $x_{a_1} = 0$, $x_{a_2} = 2$.

1. Demostrar que el tiempo de simulación no puede avanzar más de 5 segundos.

2. Dibujar la trayectoria de estado $x_1(t)$ *vs.* $x_2(t)$.

[P9.2] Función Estática Lineal Obtener un modelo DEVS atómico de una función estática lineal $f_i : \Re^n \to \Re$ definida como

$$f(u_0, u_1, \cdots, u_{n-1}) = \sum_{k=0}^{n-1} a_k \cdot u_k \tag{P9.2a}$$

donde $a_0, \cdots, a_{n-1}$ son constantes conocidas.

Luego, programar el modelo en PowerDEVS de manera tal que las constantes y el número de entradas sean parámetros.

[P9.3] Función de retarde DEVS Considerar una función que represente un retarde fijo de tiempo T.

$$f(u(t)) = u(t - T) \tag{P9.3a}$$

Suponiendo que la entrada $u(t)$ es seccionalmente constante, obtener un modelo DEVS de esta función.

Crear un bloque PowerDEVS de esta función, donde el tiempo de retardo T sea un parámetro.

Ayuda: Suponer que el número máximo de cambios en $u(t)$ está acotado en el período T por un número fijo (10000 por ejemplo).

[P9.4] Aquiles y la Tortuga Revisitados Modificar el modelo atómico PowerDEVS NHINTEGRATOR, tal que el cuantificador verifique la Ec.(P9.1b), y corroborar por simulación lo que se predice en el Problema [P9.1].

[P9.5] Varying Quantum and Hysteresis Obtener la solución exacta del sistema de la Ec.(9.9), y luego repetir las simulaciones con QSS utilizando los siguiente valores de quantum:

1. $\Delta Q_1 = \Delta Q_2 = 0.01$
2. $\Delta Q_1 = \Delta q_2 = 0.05$
3. $\Delta Q_1 = \Delta q_2 = 0.1$
4. $\Delta Q_1 = \Delta Q_2 = 1$

Comparar los resultados, y conjeturar sobre los efectos del tamaño de la cuantificación sobre el error, la estabilidad y el costo computacional (número de pasos).

Capítulo 10

Métodos de Integración por Cuantificación

Este capítulo es una traducción resumida del Capítulo 12 del libro *Continuous System Simulation* [2]. De todas maneras, en estas notas hay varios ejemplos diferentes, sobre todo en lo referente al manejo de discontinuidades.

10.1. Introducción

En los primeros capítulos del curso, vimos dos propiedades básicas de los métodos numéricos: la *precisión de la aproximación* y la *estabilidad numérica*. Para estudiar estos problemas, lo que hacíamos era tomar un sistema lineal y analizar la relación entre los autovalores de la matriz $\mathbf{F}$ de la ecuación en diferencias resultantes y los de la matriz $\mathbf{A}$ del sistema continuo original.

En el contexto del método de QSS que vimos al final del capítulo anterior, esto no tiene ningún sentido ya que ni siquiera tenemos una ecuación en diferencias. En este caso, para analizar las propiedades mencionadas, es conveniente comparar directamente los sistemas (9.6) y (9.7).

Para esto, comenzaremos escribiendo ambas representaciones en notación vectorial. El sistema continuo original tiene la forma:

$$\dot{\mathbf{x}}_\mathbf{a}(t) = \mathbf{f}(\mathbf{x}_\mathbf{a}(t), \mathbf{u}(t)) \tag{10.1}$$

y el sistema de estados cuantificados que simula el método de QSS es:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{q}(t), \mathbf{u}(t)) \tag{10.2}$$

donde $\mathbf{x}(t)$ y $\mathbf{q}(t)$ están relacionados componente a componente mediante funciones de cuantificación con histéresis.

Definiendo $\mathbf{\Delta x}(t) = \mathbf{q}(t) - \mathbf{x}(t)$, la Ec.(10.2) puede reescribirse como:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t) + \mathbf{\Delta x}(t), \mathbf{u}(t)) \tag{10.3}$$

y ahora, el modelo de simulación de la Ec.(10.2) puede verse como una *versión perturbada* del sistema original.

Una función de cuantificación con histéresis uniforme tiene una propiedad: si dos variables $q_i(t)$ y $x_i(t)$ están relacionadas por dicha función, las mismas no pueden diferir entre sí en más del quantum. Es decir:

$$|q_i(t) - x_i(t)| \leq \Delta Q \tag{10.4}$$

Esta propiedad implica que cada componente de la perturbación $\mathbf{\Delta x}$ está acotada por el quantum. En virtud de esto, el análisis de estabilidad y de precisión puede basarse en estudiar los efectos de perturbaciones acotadas. Como veremos luego, esto nos permitirá no sólo estudiar la estabilidad, sino también establecer cotas de error global.

A pesar de esta ventaja, aparecerá un nuevo problema en QSS, que veremos a continuación con un ejemplo. Consideremos el sistema de primer orden:

$$\dot{x}_a(t) = -x_a(t) + 9.5 \tag{10.5}$$

con la condición inicial $x_a(0) = 0$.

El resultado de simular este sistema con el método de QSS usando un quantum $\Delta Q = 1$ se muestra en la Fig.10.1

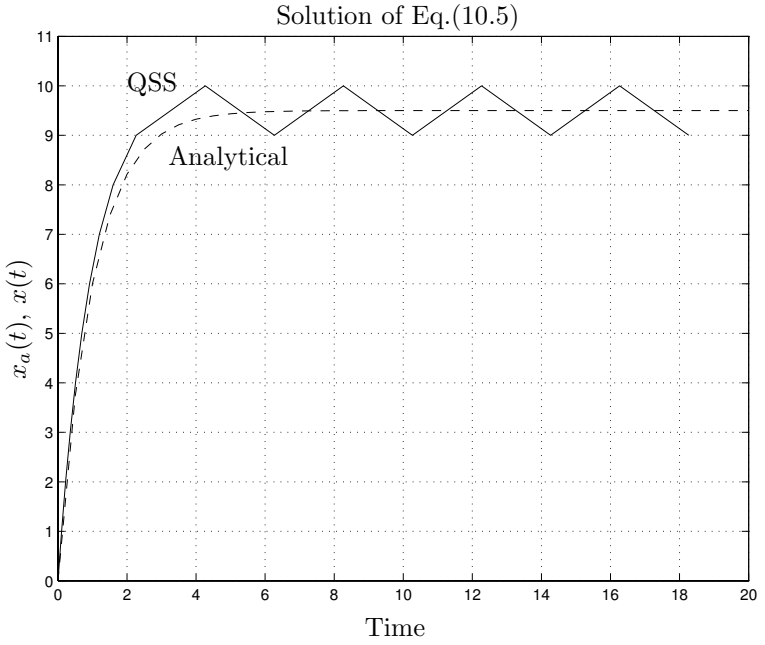

Figura 10.1: Simulación con QSS de la Ec.(10.5).

Aunque el sistema de la Ec.(10.5) es asintóticamente estable, la simulación con QSS finaliza en un ciclo límite. El punto de equilibrio $\bar{x} = 9.5$ no es más un

punto de equilibrio estable y luego no podemos afirmar que el método conserva la estabilidad del sistema.

Sin embargo, la solución que da QSS nunca se desvía lejos de la solución exacta. Incluso, esta finaliza con oscilaciones en torno al punto de equilibrio. Teniendo en cuenta que nuestra meta era simular el sistema y obtener trayectorias aproximadas, el resultado obtenido no está mal.

La trayectoria encontrada por el método de QSS se denomina *finalmente acotada*. En general, los métodos de integración por cuantificación no aseguran estabilidad de acuerdo a la definición clásica. por lo tanto, hablaremos de estabilidad en términos de cota final de las soluciones obtenidas.

10.2. Precisión y Estabilidad en QSS

Para estudiar las propiedades fundamentales de Precisión y Estabilidad del método de QSS, introduciremos primero algo de notación.

Utilizaremos el símbolo $|\cdot|$ para denotar el módulo por componentes de un vector o matriz. Por ejemplo, si $\mathbf{G}$ es una matriz de $j \times k$ con componentes complejos $g_{1,1}, \ldots, g_{j,k}$, entonces $|\mathbf{G}|$ será también una matriz de $j \times k$ con componentes reales positivos $|g_{1,1}|, \ldots, |g_{j,k}|$.

De manera similar, utilizaremos el símbolo "$\leq$" para realizar comparaciones por componentes entre vectores reales de idéntica longitud. Así, la expresión $\mathbf{x} \leq \mathbf{y}$ implica que $x_1 \leq y_1, \ldots, x_n \leq y_n$.

Con estas definiciones, llamemos $\mathbf{x_a}(t)$ a la solución del sistema lineal y estacionario (SLE):

$$\dot{\mathbf{x}}_\mathbf{a}(t) = \mathbf{A} \cdot \mathbf{x}_\mathbf{a}(t) + \mathbf{B} \cdot \mathbf{u}(t) \tag{10.6}$$

Sea $\mathbf{x}(t)$ la solución, comenzando de la misma condición inicial, de su QSS asociado:

$$\dot{\mathbf{x}}(t) = \mathbf{A} \cdot \mathbf{q}(t) + \mathbf{B} \cdot \mathbf{u}(t) \tag{10.7}$$

que puede reescribirse como:

$$\dot{\mathbf{x}}(t) = \mathbf{A} \cdot [\mathbf{x}(t) + \mathbf{\Delta x}(t)] + \mathbf{B} \cdot \mathbf{u}(t) \tag{10.8}$$

Definamos el error $\mathbf{e}(t) \triangleq \mathbf{x}(t) - \mathbf{x_a}(t)$. Restando la Ec.(10.6) de la Ec. (10.8), resulta que $\mathbf{e}(t)$ satisface la ecuación:

$$\dot{\mathbf{e}}(t) = \mathbf{A} \cdot [\mathbf{e}(t) + \mathbf{\Delta x}(t)] \tag{10.9}$$

con $\mathbf{e}(t_0) = 0$ ya que ambas trayectorias, $\mathbf{x_a}(t)$ y $\mathbf{x}(t)$, comienzan desde condiciones iniciales idénticas.

Comencemos analizando el caso escalar:

$$\dot{e}(t) = a \cdot [e(t) + \Delta x(t)] \tag{10.10}$$

Por motivos que pronto se aclararán, asumiremos que a, e, y Δx pertenecen a $\mathbb{C}$. Requeriremos también que $\mathbb{R}\text{e}\{a\}$ sea negativo.

También supondremos que $|\Delta x| \leq w$, siendo w alguna constante positiva.

$e(t)$ puede escribirse en notación polar como:

$$e(t) = \rho(t) \cdot e^{j\theta(t)} \tag{10.11}$$

donde $\rho(t) = |e(t)|$.

Luego la Ec.(10.10) queda:

$$\dot{\rho}(t) \cdot e^{j\theta(t)} + \rho(t) \cdot e^{j\theta(t)} \cdot j \cdot \dot{\theta}(t) = a \cdot [\rho(t) \cdot e^{j\theta(t)} + \Delta x(t)] \tag{10.12}$$

o:

$$\dot{\rho}(t) + \rho(t) \cdot j \cdot \dot{\theta}(t) = a \cdot [\rho(t) + \Delta x(t) \cdot e^{-j\theta(t)}] \tag{10.13}$$

Tomando sólo la parte real de la última ecuación:

$$\begin{aligned} \dot{\rho}(t) &= \mathbb{R}\text{e}\{a\} \cdot \rho(t) + \mathbb{R}\text{e}\{a \cdot \Delta x(t) \cdot e^{-j\theta(t)}\} \\ &\leq \mathbb{R}\text{e}\{a\} \cdot \rho(t) + |a| \cdot |\Delta x(t)| \\ &\leq \mathbb{R}\text{e}\{a\} \cdot \left[\rho(t) - \left|\frac{a}{\mathbb{R}\text{e}\{a\}}\right| \cdot w\right] \end{aligned} \tag{10.14}$$

Entonces, como $\mathbb{R}\text{e}\{a\}$ es negativo y $\rho(0) = 0$, será siempre cierto que:

$$|e(t)| = \rho(t) \leq \left|\frac{a}{\mathbb{R}\text{e}\{a\}}\right| \cdot w \tag{10.15}$$

ya que, cuando ρ alcanza el límite superior, la derivada ser torna negativa (o cero).

Veamos ahora el caso desacoplado. Supongamos que la matriz $\mathbf{A}$ en la Ec.(10.9) es diagonal, con componentes $a_{i,i}$ complejos con parte real negativa. Supongamos también que

$$|\mathbf{\Delta x}(t)| \leq \mathbf{w} \tag{10.16}$$

Repitiendo el análisis escalar para cada componente de $\mathbf{e}(t)$, resulta que:

$$|\mathbf{e}| \leq |\mathbb{R}\text{e}\{\mathbf{A}\}^{-1} \cdot \mathbf{A}| \cdot \mathbf{w} \tag{10.17}$$

Finalmente, volvamos nuevamente a la Ec.(10.9), esta vez suponiendo que la matriz $\mathbf{A}$ es Hurwitz y diagonalizable. Asumamos que el quantum es tal que:

$$|\mathbf{\Delta x}| = |\mathbf{q} - \mathbf{x}| \leq \mathbf{\Delta Q} \tag{10.18}$$

Llamemos $\mathbf{\Lambda}$ a la matriz diagonal de autovalores de $\mathbf{A}$, y sea $\mathbf{V}$ una matriz de autovectores correspondiente. Luego

$$\mathbf{A} = \mathbf{V} \cdot \mathbf{\Lambda} \cdot \mathbf{V}^{-1} \tag{10.19}$$

es la descomposición espectral de la matriz $\mathbf{A}$.

Consideremos el cambio de variables:

$$\mathbf{z}(t) = \mathbf{V}^{-1} \cdot \mathbf{e}(t) \tag{10.20}$$

Usando la nueva variable $\mathbf{z}$, la Ec.(10.9) puede reescribirse como:

$$\mathbf{V} \cdot \dot{\mathbf{z}}(t) = \mathbf{A} \cdot [\mathbf{V} \cdot \mathbf{z}(t) + \mathbf{\Delta x}(t)] \tag{10.21}$$

y luego:

$$\dot{\mathbf{z}}(t) = \mathbf{V}^{-1} \cdot \mathbf{A} \cdot [\mathbf{V} \cdot \mathbf{z}(t) + \mathbf{\Delta x}(t)] \tag{10.22}$$

$$= \mathbf{\Lambda} \cdot [\mathbf{z}(t) + \mathbf{V}^{-1} \cdot \mathbf{\Delta x}(t)] \tag{10.23}$$

De la Ec.(10.18), resulta que:

$$|\mathbf{V}^{-1} \cdot \mathbf{\Delta x}| \leq |\mathbf{V}^{-1}| \cdot \mathbf{\Delta Q} \tag{10.24}$$

Teniendo en cuneta que la matriz $\mathbf{\Lambda}$ es diagonal, resulta que la Ec.(10.23) es el caso diagonal que analizamos antes. Luego, de la Ec.(10.17), resulta que:

$$|\mathbf{z}(t)| \leq |\mathbb{R}\mathrm{e}\{\mathbf{\Lambda}\}^{-1} \cdot \mathbf{\Lambda}| \cdot |\mathbf{V}^{-1}| \cdot \mathbf{\Delta Q} \tag{10.25}$$

y entonces:

$$|\mathbf{e}(t)| \leq |\mathbf{V}| \cdot |\mathbf{z}(t)| \leq |\mathbf{V}| \cdot |\mathbb{R}\mathrm{e}\{\mathbf{\Lambda}\}^{-1} \cdot \mathbf{\Lambda}| \cdot |\mathbf{V}^{-1}| \cdot \mathbf{\Delta Q} \tag{10.26}$$

Podemos entonces concluir que:

$$|\mathbf{x}(t) - \mathbf{x_a}(t)| \leq |\mathbf{V}| \cdot |\mathbb{R}\mathrm{e}\{\mathbf{\Lambda}\}^{-1} \cdot \mathbf{\Lambda}| \cdot |\mathbf{V}^{-1}| \cdot \mathbf{\Delta Q} \tag{10.27}$$

La desigualdad de la Ec.(10.27) tiene implicancias teóricas y prácticas muy importantes.

Puede verse que la cota de error es proporcional al quantum y, para cualquier quantum utilizado, el error está siempre acotado. Es también importante notar que la desigualdad Ec.(10.27) es la expresión analítica de la *cota de error global*. Los métodos de tiempo discreto no tienen tales fórmulas. El hecho que la Ec.(10.27) sea independiente de las condiciones iniciales y las trayectorias de entrada tendrá también aparejada sus ventahas.

De alguna manera, el método de QSS ofrece un control de error intríseco, sin necesidad de ninguna regla de adaptación de paso. Mas aún:

> El método de QSS es siempre estable, sin utilizar ninguna fórmula implícita.

10.3. Elección del Quantum

El método de QSS requiere elegir un quantum apropiado. Aunque los resultados sean siempre estables, el quantum debe ser el adecuado para obtener resultados de simulación decentes.

En sistemas lineales y estacionarios, la desigualdad de la Ec.(10.27) puede usarse para diseñar la cuantificación. Dado un error máximo admitido, puede encontrarse fácilmente el valor de $\mathbf{\Delta Q}$ que satisface dicha desigualdad.

De todas formas, para evitar calcular autovalores y autovectores, y teniendo en cuenta la posibilidad de simular sistemas no lineales, podemos dar una regla empírica para elegir el quantum. Una buena opción es tomarlo proporcional (por ejemplo 100 veces menor) a la máxima amplitud que tomará la variable correspondiente. Esto obviamente requiere conocer de antemano el orden de magnitud de las variables de la simulación.

10.4. Señales de Entrada en el Método de QSS

Ya mencionamos que el método de QSS permite simular sistemas estacionarios con entradas seccionalmente constantes.

Para incorporar dichas señales a una simulación, todo lo que debemos hacer es agregar un modelo DEVS que genere secuencias de eventos que representen dichas señales seccionalmente constantes, y luego acoplar estos *generadores* con las funciones estáticas que calculan las derivadas.

Supongamos que tenemos una señal de entrada escalar seccionalmente constante $u(t)$ que toma los valores v_1, v_2, ..., v_j, ... en los instantes t_1, t_2, ..., t_j, ..., respectivamente. Un modelo DEVS que produce eventos de acuerdo a esta señal de entrada puede especificarse como sigue:

$$
\begin{aligned}
&M_{IN} = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta), \text{ donde} \\
&X = \emptyset \\
&Y = \Re \times \mathbb{N} \\
&S = \mathbb{N} \times \Re_0^+ \\
&\delta_{\text{int}}(s) = \delta_{\text{int}}(j, \sigma) = (j+1, t_{j+1} - t_j) \\
&\lambda(s) = \lambda(j, \sigma) = (v_j, 0) \\
&ta(s) = ta(j, \sigma) = \sigma
\end{aligned}
$$

Una ventaja interesante del método de QSS es que trata las trayectorias de entrada de manera asincrónica. El evento que indica un cambio en una señal siempre se procesa en el instante de tiempo correcto, provocando cambios en las pendientes de las variables de estado afectadas directamente.

Esta es una característica intrínseca del método, que se obtiene sin modificar para nada los modelos DEVS de los integradores cuantificados ni de las funciones estáticas. En otras palabras, los *eventos temporales* se resuelven de

forma automática sin necesidad de agregar nada al código (a diferencia de los métodos de tiempo discreto donde teníamos que ajustar el paso).

Hasta aquí consideramos sólamente entradas seccionalmente constantes. Sin embargo, en la mayor parte de las aplicaciones las señales de entrada toman formas más generales. En estos casos, podemos aproximar estas señales por trayectorias seccionalmente constantes y luego generarlas mediante modelos DEVS.

Al hacer esto, debemos tener en cuenta que estamos agregando una nueva fuente de error. En el caso particular de los SLE, la presencia de este nuevo error de cuantificación de la entrada transforma la Ec.(10.27) en:

$$\begin{aligned} |\mathbf{x}(t) - \mathbf{x_a}(t)| \quad \leq \quad & |\mathbf{V}| \cdot |\mathbb{R}\mathrm{e}\{\mathbf{\Lambda}\}^{-1} \cdot \mathbf{\Lambda}| \cdot |\mathbf{V}^{-1}| \cdot \mathbf{\Delta Q} \\ & + |\mathbf{V}| \cdot |\mathbb{R}\mathrm{e}\{\mathbf{\Lambda}\}^{-1} \cdot \mathbf{V}^{-1} \cdot \mathbf{B}| \cdot \mathbf{\Delta u} \end{aligned} \tag{10.28}$$

donde $\mathbf{\Delta u}$ es el vector con el máximo error de cuentificación en cada señal de entrada (ver Prob.[P10.2]).

10.5. Arranque e Interpolación

El arranque del método de QSS consiste en asignar condiciones iniciales apropiadas a los modelos atómicos DEVS que realizan la simulación.

El estado del integrador cuantificado puede escribirse como $s = (x, d_x, q, \sigma)$ (ver modelo M_{HQI} en la página 146), donde x es la variable de estado, d_x su derivada, q es la variable cuantificada y σ el avance de tiempo.

Es claro de debemos elegir $x = x_i(t_0)$ y $q = |DeltaQ_i = \mathrm{floor}(x_i(t_0)/\Delta Q_i)$.

Una manera sencilla de inicializar todas las demás variables (incluyendo las de las funciones estáticas), es colocando $\sigma = 0$, de modo tal que todos los integradores cuantificados realicen un paso en $t = t_0$ en el cual *muestren* el valor de salida que tienen, de forma tal que las funciones estáticas calculen correctamente las derivadas iniciales.

Esto, de todas formas, puede conllevar un problema. Como todos los integradores intentarán provocar un evento al mismo tiempo, el simulador elegirá uno de ellos para comenzar (de acuerdo a las *prioridades*). Tras esto, algunas funciones estáticas recibirán el primer evento y también colocarán su σ en cero. Por lo tanto, es posible que el simulador escoja alguna de estas funciones para producir la salida y entonces algunos integradores recibirán un evento antes de sacar el suyo correspondiente. Si ocurriera esto, tras las transiciones externas de dichos integrador, los *sigma* correspondientes se modificarán y ya no serán 0 y nadie verá cuáles eran los estados iniciales.

Por lo tanto, hay que verificar en la transición externa de los integradores la condición $\sigma = 0$. En tal caso, se deberá dejar $\sigma = 0$ para forzar el evento de salida.

Por último, los generadores de señales también deberán tener iniciamente su valor de $sigma = 0$ para que las funciones estáticas calculen las derivadas correctamente desde el instante inicial.

En cuanto a la interpolación de valores de salida, como sabemos, las trayectorias de estado en QSS son seccionalmente lineales y continuas. Por lo tanto, el problema es trivial: podemos conocer el valor del estado en cualquier instante de tiempo.

Es importante tener en cuenta que las Ecs.(10.27) y (10.28) son válidas para todo t, por lo que los valores interpolados se mantienen también dentro de la cota teórica de error global.

10.6. Método de QSS de Segundo Orden

Más allá de las propiedades teóricas importantes del método de QSS, el principal problema del mismo es que realiza sólo una aproximación de primer orden y no puede obtenerse una buena precisión.

De hecho, la Ec.(10.27) nos muestra que la cota de error crece linealmente con el quantum y entonces para disminuir 100 veces el error debemos disminuir 100 veces el quantum, lo que aumentará aproximadamente en 100 veces el número de pasos.

Para mejorar esta situación, existe un método de segundo orden denominado *método de cuantificación de estados de segundo orden*, o símplemente QSS2, que como veremos, tendrá propiedades muy similares a QSS.

La idea básica de QSS2 es la utilización de *cuantificadores de primer orden*, que reemplazan las funciones de cuantificación con histéresis de QSS.

En un cuantificador con histéresis, la variable seccionalmente constante $q(t)$ cambia cada vez que difiere de $x(t)$ en el quantum ΔQ. La idea del cuantificador de primer orden es la misma, sólo que ahora la salida del cuantificador $q(t)$ es seccionalmente lineal, como se muestra en la Fig.10.2.

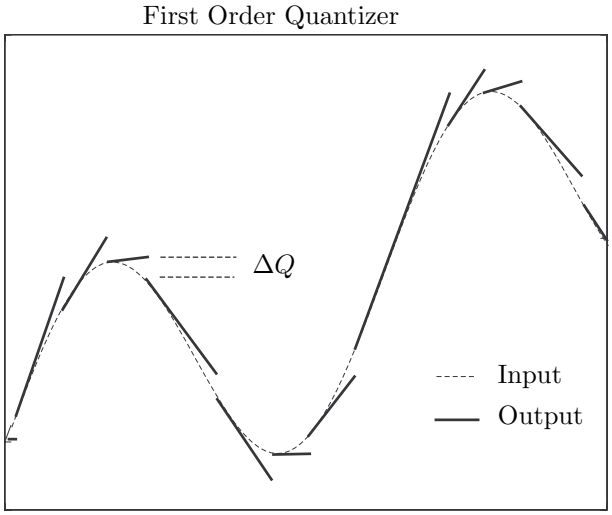

Figura 10.2: Entrada y salida en un cuantificador de primer orden.

Formalmente, diremos que las trayectorias $x(t)$ y $q(t)$ se relacionan con una función de cuantificación de primer orden si satisfacen:

$$q(t) = \begin{cases} x(t) & \text{si} \quad t = t_0 \vee |q(t^-) - x(t^-)| = \Delta Q \\ q(t_j) + m_j \cdot (t - t_j) & \text{en otro caso} \end{cases} \tag{10.29}$$

con la secuencia $t_0, \ldots, t_j, \ldots$ definida como:

$$t_{j+1} = \text{mín}(t), \quad \forall t > t_j \wedge |x(t_j) + m_j \cdot (t - t_j) - x(t)| = \Delta Q$$

y las pendientes:

$$m_0 = 0; \quad m_j = \dot{x}(t_j^-), \quad j = 1, \ldots, k, \ldots$$

Puede notarse que la función de cuantificación de primer orden tiene histéresis (debido a la presencia de la función valor absoluto).

El método de QSS2 entonces simula un sistema con el de la Ec.(10.2), donde las componentes de $\mathbf{q}(t)$ y de $\mathbf{x}(t)$ se relacionan componente a componente mediante funciones de cuantificación de primer orden. En consecuencia, las variables cuantificadas $q_i(t)$ son seccionalmente lineales.

En QSS, tanto las variables cuantificadas como las derivadas del estado eran seccionalmente constantes. En QSS2, las variables cuantificadas serán seccionalmente lineales, pero no puede garantizarse que las derivadas del estado también lo sean. El problema es que al aplicar una función no lineal a una trayectoria seccionalmente lineal el resultado no es seccionalmente lineal.

En el caso lineal, sin embargo, si las trayectorias de entrada también son seccionalmente lineales, las derivadas de las variables de estado serán también seccionalmente lineales y las trayectorias de estado serán seccionalmente parabólicas. En los casos no lineales, lo que haremos será aproximar las derivadas de los estados por funciones seccionalmente lineales, y obtendremos también trayectorias de estado seccionalmente parabólicas.

Para construir los modelos DEVS utilizaremos la misma idea de QSS, pero ahora, como las trayectorias serán seccionalmente lineales, deberemos tener en cuenta no sólo los valores sino también las pendientes de las mismas. Por lo tanto, los eventos deberán llevar dos números, uno indicando el valor instantáneo y el otro la pendiente de cada segmento lineal de la trayectoria.

Al igual que en QSS, tendremos modelos de integradores cuantificados (ahora de segundo orden) y funciones estáticas. Para obtener el modelo DEVS de un integrador cuantificado de segundo orden, supondremos que la derivada del estado se describe, en un intervalo $[t_k, t_{k+1}]$, mediante:

$$\dot{x}(t) = d_x(t_k) + m_{d_x}(t_k) \cdot (t - t_k) \tag{10.30}$$

donde $d_x(t_k)$ es la derivada del estado en t_k, y $m_{d_x}(t_k)$ es la pendiente correspondiente. La pendiente de la derivada del estado, m_{d_x}, será, en general, diferente

a la pendiente de la variable cuentificada, m_q. Luego, la trayectoria del estado se puede escribir como:

$$x(t) = x(t_k) + d_x(t_k) \cdot (t - t_k) + \frac{m_{d_x}(t_k)}{2} \cdot (t - t_k)^2 \tag{10.31}$$

Si t_k es un instante en el cual ocurre un cambio en la variable cuantificada, es decir, es el instante de una transición interna, entonces $q(t_k)$ tomará el mismo valor y pendiente que $x(t_k)$:

$$q(t) = x(t_k) + d_x(t_k) \cdot (t - t_k) = q(t_k) + m_q(t_k) \cdot (t - t_k) \tag{10.32}$$

y luego, el instante de tiempo en que $x(t)$ y $q(t)$ diferirán entre sí en ΔQ puede calcularse como:

$$t = t_k + \sqrt{\frac{2 \cdot \Delta Q}{|m_{d_x}(t_k)|}} \tag{10.33}$$

Si t_k es otro instante de tiempo, o sea, el tiempo de una transición externa, el tiempo en el cual $|q(t) - x(t)| = \Delta Q$ deberá recalcularse. Ahora, la trayectoria cuantificada puede escribirse como:

$$q(t) = q(t_k) + m_q(t_k) \cdot (t - t_k) \tag{10.34}$$

y deberemos calcular el valor de t en el cual $|q(t) - x(t)| = \Delta Q$ hallando las raíces de un polinomio cuadrático.

Siguiendo estas ideas, un modelo DEVS que representa el comportamiento de un integrador cuantificado de segundo orden puede ser el siguiente.

$M_{QI2} = (X, S, Y, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta)$, donde:
$X = \Re^2 \times \mathbb{N}$
$S = \Re^5 \times \Re_0^+$
$Y = \Re^2 \times \mathbb{N}$
$\delta_{\text{int}}(d_x, m_{d_x}, x, q, m_q, \sigma) = (d_x + m_{d_x} \cdot \sigma, m_{d_x}, \tilde{q}, \tilde{q}, d_x + m_{d_x} \cdot \sigma, \sigma_1)$
$\delta_{\text{ext}}(d_x, m_{d_x}, x, q, m_q, \sigma, e, x_v, m_{x_v}, p) = (x_v, m_{x_v}, \tilde{x}, q + m_q \cdot e, m_q, \sigma_2)$
$\lambda(d_x, m_{d_x}, x, q, m_q, \sigma) = (\tilde{q}, d_x + m_{d_x} \cdot \sigma, 0)$
$ta(d_x, m_{d_x}, x, q, m_q, \sigma) = \sigma$

con:

$$\tilde{q} = x + d_x \cdot \sigma + \frac{m_{d_x}}{2} \cdot \sigma^2; \;\; \tilde{x} = x + d_x \cdot e + \frac{m_{d_x}}{2} \cdot e^2$$

$$\sigma_1 = \begin{cases} \sqrt{\frac{2 \cdot \Delta Q}{|m_{d_x}|}} & \text{si} \quad m_{d_x} \neq 0 \\ \infty & \text{en otro caso} \end{cases} \tag{10.35}$$

y σ_2 debe calcularse como la menor solución positiva de:

$$|\tilde{x} + x_v \cdot \sigma_2 + \frac{m_{x_v}}{2} \cdot \sigma_2^2 - (q + m_q \cdot e + m_q \cdot \sigma_2)| = \Delta Q \tag{10.36}$$

Este modelo M_{QI2} representa exactamente un integrador cuantificado de segundo orden con entrada seccionalmente lineal.

El modelo atomico PowerDEVS correspondiente puede ponerse como:

ATOMIC MODEL QSS2INT
State Variables and Parameters:

double $dx, mdx, X, q, mq, sigma$; //states
double y[2]; //output
double inf, dq; //parameters

Init Function:

va_list $parameters$;
va_start($parameters$, t);
dq = va_arg($parameters$, double);
X = va_arg($parameters$, double);
$inf = 1e10$;
$q = X$;
$dx = 0$;
$mdx = 0$;
$mq = 0$;
$sigma = 0$;

Time Advance Function:

return $sigma$;

Internal Transition Function:

$X = X + dx * sigma + mdx/2 * sigma * sigma$;
$q = X$;
$dx = dx + mdx * sigma$;
$mq = dx$;
if ($mdx == 0$) {
  $sigma = inf$;
}
else
  $sigma$ = sqrt($2 * dq$/fabs(mdx));
};

External Transition Function:

double $*xv$;
double a, b, c, s;
xv = (double$*$)($x.value$);
$X = X + dx * e + mdx/2 * e * e$;
$dx = xv$[0]; //input value
$mdx = xv$[1]; //input slope
if ($sigma$! = 0) {
  $q = q + mq * e$;

```
a = mdx/2;
b = dx − mq;
c = X − q + dq;
sigma = inf;
if (a == 0) {
    if (b ! = 0) {
        s = −c/b;
        if (s > 0) {sigma = s;};
        c = X − q − dq;
        s = −c/b;
        if ((s > 0) && (s < sigma)) {sigma = s;};
    };
}
else {
    s = (−b + sqrt(b ∗ b − 4 ∗ a ∗ c))/2/a;
    if (s > 0) {sigma = s;};
    s = (−b − sqrt(b ∗ b − 4 ∗ a ∗ c))/2/a;
    if ((s > 0) && (s < sigma)) {sigma = s;};
    c = X − q − dq;
    s = (−b + sqrt(b ∗ b − 4 ∗ a ∗ c))/2/a;
    if ((s > 0) && (s < sigma)) {sigma = s;};
    s = (−b − sqrt(b ∗ b − 4 ∗ a ∗ c))/2/a;
    if ((s > 0) && (s < sigma)) {sigma = s;};
};
};
```

Output Function:

```
y[0] = X + dx ∗ sigma + mdx/2 ∗ sigma ∗ sigma;
y[1] = u + mdx ∗ sigma;
return Event(&y[0], 0);
```

En QSS, además de integradores cuantificados, utilizamos también modelos DEVS de funciones estáticas. Para QSS2 también necesitaremos estas funciones, pero las mismas deberán también tener en cuenta las pendientes.

Cada componente f_j de una función estática $\mathbf{f}(\mathbf{q}, \mathbf{u})$ recibe las trayectorias seccionalmente lineales de las variables cuantificadas y de entrada.

Definiendo $\mathbf{v} \triangleq [\mathbf{q}; \mathbf{u}]$, cada componente de $\mathbf{v}$ es seccionalmente lineal:

$$v_j(t) = v_j(t_k) + m_{v_j}(t_k) \cdot (t - t_k)$$

Luego, la salida de la función estática se puede escribir:

$$\dot{x}_i(t) = f_i(\mathbf{v}(t)) = f_i(v_1(t_k) + m_{v_1}(t_k) \cdot (t - t_k), \ldots, v_l(t_k) + m_{v_l}(t_k) \cdot (t - t_k))$$

donde l es el número de componentes de $\mathbf{v}(t)$.

Definiendo $\mathbf{m_v} \triangleq [m_{v_1}, \ldots, m_{v_l}]^T$, la última ecuación puede reescribirse como:

$$\dot{x}_i(t) = f_i(\mathbf{v}(t)) = f_i(\mathbf{v}(t_k) + \mathbf{m_v}(t_k) \cdot (t - t_k))$$

que puede desarrollarse en serie de Taylor:

$$\dot{x}_i(t) = f_i(\mathbf{v}(t)) = f_i(\mathbf{v}(t_k)) + \left(\frac{\partial f_j}{\partial \mathbf{v}}(\mathbf{v}(t_k))\right)^T \cdot \mathbf{m_v}(t_k) \cdot (t - t_k) + \dots \quad (10.37)$$

Luego, una aproximación seccionalmente lineal de la salida puede obtenerse truncando la serie de Taylor tras los dos primeros términos de la Ec.(10.37).

En el caso lineal, tenemos $\mathbf{f}(\mathbf{v}(t)) = \mathbf{A} \cdot \mathbf{v}(t)$, y luego: $f_i(\mathbf{v}(t)) = \mathbf{a_i}^T \cdot \mathbf{v}(t)$ donde $\mathbf{a_i} \in \Re^l$. Luego:

$$\dot{x}_i(t) = \mathbf{a_i}^T \cdot \mathbf{v}(t_k) + \mathbf{a_i}^T \cdot \mathbf{m_v}(t_k) \cdot (t - t_k)$$

lo que implica que el valor de salida y su pendiente se obtienen como combinaciones lineales de sus valores de entrada y sus pendientes respectivas. La construcción del modelo DEVS correspondiente se deja como ejercicio (Prob. [P10.3]).

El caso no lineal requiere conocer las derivadas parciales de f_i. En caso que no conozcamos la expresión de las mismas, deberíamos calcularlas de manera numérica.

Un modelo DEVS que sigue esta idea para una fución no lineal genérica $f_i(\mathbf{v}) = f(v_1, \dots, v_l)$ es el que sigue:

$$\begin{aligned}
&M_{F2} = (X, S, Y, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta), \text{ donde:}\\
&X = \Re^2 \times \mathbb{N}\\
&S = \Re^{3l} \times \Re_0^+\\
&Y = \Re^2 \times \mathbb{N}\\
&\delta_{\text{int}}(\mathbf{v}, \mathbf{m_v}, \mathbf{c}, \sigma) = (\mathbf{v}, \mathbf{m_v}, \mathbf{c}, \infty)\\
&\delta_{\text{ext}}(\mathbf{v}, \mathbf{m_v}, \mathbf{c}, \sigma, e, x_v, m_{x_v}, p) = (\tilde{\mathbf{v}}, \tilde{\mathbf{m}}_\mathbf{v}, \tilde{\mathbf{c}}, 0)\\
&\lambda(\mathbf{v}, \mathbf{m_v}, \mathbf{c}, \sigma) = (f_i(\mathbf{v}), m_f, 0)\\
&ta(\mathbf{v}, \mathbf{m_v}, \mathbf{c}, \sigma) = \sigma
\end{aligned}$$

Los coeficientes $\mathbf{c} = (c_1, \dots, c_l)^T$ y $\tilde{\mathbf{c}} = (\tilde{c}_1, \dots, \tilde{c}_l)^T$ son estimas de las derivadas parciales $\frac{\partial f_i}{\partial v_j}$ utilizadas para calcular la pendiente de salida de acuerdo a:

$$m_f = \sum_{j=1}^{n} c_j \cdot m_{v_j}$$

Tras una transición externa, las entradas, sus pendientes y las derivadas se actualizan según:

$$\tilde{v}_j = \begin{cases} x_v & \text{si} \quad p + 1 = j \\ v_j + m_{v_j} \cdot e & \text{en otro caso} \end{cases}$$

$$\tilde{m}_{v_j} = \begin{cases} m_{x_v} & \text{si} \quad p+1=j \\ m_{v_j} & \text{en otro caso} \end{cases}$$

$$\tilde{c}_j = \begin{cases} \dfrac{f_i(\mathbf{v}+\mathbf{m_v}\cdot e)-f_i(\tilde{\mathbf{v}})}{v_j+m_{v_j}\cdot e-\tilde{v}_j} & \text{si} \quad p+1=j \quad \wedge \quad v_j+m_{v_j}\cdot e-\tilde{v}_j \neq 0 \\ c_j & \text{en otro caso} \end{cases} \tag{10.38}$$

donde p denota el número del puerto correspondiente a la entrada que cambia. Siempre que sea posible, convendrá reemplazar la Ec.(10.38) por la expresión analítica de la derivada parcial correspondiente.

El modelo PowerDEVS de esta función estática no lineal general es el que sigue:

ATOMIC MODEL STFUNCTION2
State Variables and Parameters:

double $sigma, v[10], mv[10], c[10]$; //states
double $y[2]$; //output
double inf;
int l;

Init Function:

va_list $parameters$;
va_start($parameters$, t);
l = va_arg($parameters$, double);
$inf = 1e10$;
$sigma = inf$;
for (int $i = 0; i < l; i++$) {
  $v[i] = 0$;
  $mv[i] = 0$;
};

Time Advance Function:

return $sigma$;

Internal Transition Function:

$sigma = inf$;

External Transition Function:

double $*xv$;
double $fv, vaux$;
xv = (double$*$)($x.value$);
for (int $i = 0; i < l; i++$) {
  $v[i] = v[i] + mv[i] * e$;
};
$fv = f_j(v)$;//put your function here
$vaux = v[x.port]$;
$v[x.port] = xv[0]$;
$mv[x.port] = xv[1]$;

$y[0] = f_j(v)$; //put your function here
if $(vaux\ != v[x.port])$ {
  $c[x.port] = (fv - y[0])/(vaux - v[x.port])$;
};
$y[1] = 0$;
for (int $i = 0; i < l; i++$) {
  $y[1] = y[1] + mv[i] * c[i]$;
};
$sigma = 0$;

Output Function:

return Event($\&y[0], 0$);

El problema de este modelo de PowerDEVS es que debe definirse un modelo atómico nuevo para cada función distinta. Sin embargo, en PowerDEVS hay un bloque que permite ingresar la expresión de la función f_j como un parámetro, y que luego hace uso de un parser de expresiones algebraicas.

Además, en PowerDEVS hay modelos de distintas funciones no lineales ya definidas (mutiplicadores, senos, cosenos, funciones cuadráticas, cúbicas, etc.) que difieren del modelo mostrado en que utilizan la expresión analítica de las derivadas parciales.

En cuanto a las señales de entrada, el principio es exactamente el mismo que en QSS, sólo que ahora podemos también representar trayectorias seccionalmente lineales en forma exacta. PowerDEVS cuenta con varios modelos de generadores de señales para QSS2.

Utilizando los modelos M_{QI2}, M_{F2} y eventualmente modelos para generar trayectorias de entrada, y acoplando de la misma forma que en QSS, podemos simular cualquier sistema mediante el método de QSS2.

Respecto a las propiedades de estabilidad y precisión, recordemos que en QSS deducimos las mismas a partir de considerar que el QSS era una versión perturbada del sistema original, y que las perturbaciones eran acotadas (ya que x_i y q_i nunca diferían entre sí más que ΔQ_i). Es muy fácil de ver que en QSS2 pasa exactamente lo mismo, ya que la definición de la función de cuantificación de primer orden también implica que cada término de perturbación estará acotado por el quantum correspondiente. En definitiva, la Ec.(10.27) tendrá validez también en QSS2.

Una pregunta que puede surgir aquí es la siguiente: ¿para qué usar QSS2 si tiene el mismo error que QSS?. La respuesta es simple: en QSS el número de pasos aumenta linealmente al disminuir el quantum, en cambio en QSS2 el número de pasos aumenta sólo con la raíz cuadrada, como puede verse en la Ec.(10.35).

Además, mirando la Fig.10.2 queda claro que QSS2 utilizará en general menos pasos que QSS cuando ambos tengan el mismo quantum.

Como ejemplo de todo esto, veamos un caso de simulación con QSS2: El circuito de la Fig.10.3 representa una linea de transmisión RLC similar a la que vimos en el capítulo 8 del curso.

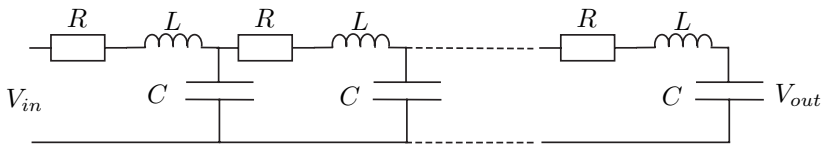

Figura 10.3: Línea de transmisión RLC.

Considerando cinco secciones, se obtiene un modelo lineal de orden 10 cuya matriz de evolución es:

$$\mathbf{A} = \begin{pmatrix} -R/L & -1/L & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/C & 0 & -1/C & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/L & -R/L & -1/L & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/C & 0 & -1/C & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/C & 0 \end{pmatrix}$$

Considerando que esta línea representa una pista de un circuito digital, una forma típica de trayectoria de entrada es una onda trapezoidal, que puede representarse exactamente en QSS2 mediante un modelo DEVS (la trayectoria es seccionalmente lineal).

Dado que la representación es exacta, tiene validez la ecuación de la cota de error Ec.(10.27).

Para simular este ejemplo, usamos los parámetros $R = 80\ \Omega$, $C = 0.2\ pF$, y $L = 20\ nH$.

La entrada trapezoidal tiene tiempos de subida y bajada de $T_1 = T_3 = 10\ psec$, mientras que la duración de los estados alto y bajo es $T_0 = T_2 = 1\ nsec$. Los niveles bajo y alto son $0\ V$ y $2.5\ V$, respectivamente.

Tomamos un quantum $\Delta v = 4\ mV$ para las variables que representan voltajes (x_j con j par) y $\Delta i = 10\ \mu A$ para las corrientes (x_j con j impar).

De acuerdo a la Ec.(10.27), esta cuantificación garantiza que el error en la variable $V_{out} = x_{10}$ es menor que $250\ mV$.

Las trayectorias de entrada y salida se muestran en la Fig.10.4.

La simulación tomó un total de 2536 pasos (entre 198 y 319 transiciones internas en cada integrador) para obtener los primeros 3.2 $nsec$ de las trayectorias.

Repetimos el experimento usando un quantum 100 veces menor en todas las variables, lo que asegura que el error en V_{out} es menor que $2.5\ mV$. Esta nueva simulación consumió un total de 26.883 pasos (aproximadamente 10 veces más, corroborando lo que dijimos antes de que el número de pasos crece con la raíz cuadrada).

Aunque el número de pasos es grande, no debemos olvidar que cada paso involucra sólamente cálculos escalares en tres integradores (el que realiza la

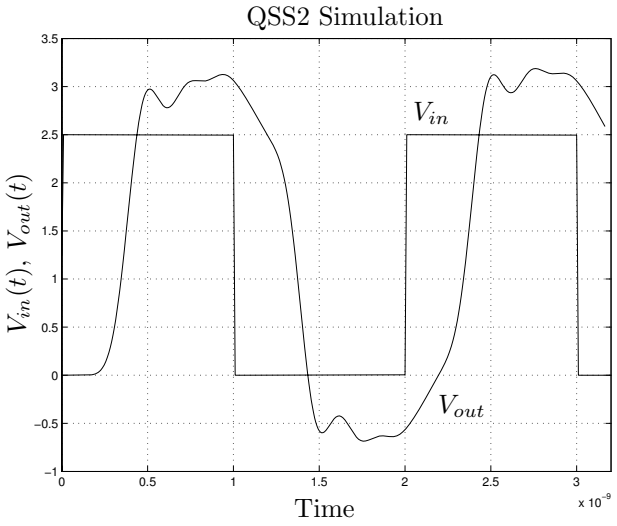

Figura 10.4: Simulación con QSS2 de una línea de transmisión RLC.

transición y los dos que están directamente conectados al mismo). Esto se debe a la raleza de la matriz **A**.

10.7. Simulación de DAEs con los métodos de QSS

La Figura 10.5 muestra la línea de transmisión de la Fig.10.3, modificada con el agregado de una carga.

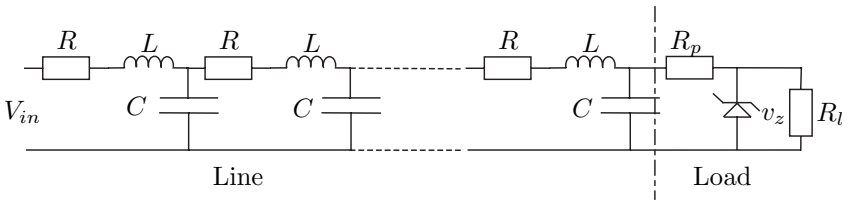

Figura 10.5: Línea de transmisión con protección para sobretensión.

La carga tiene una resistencia R_l, y un circuito de protección formado por un diodo Zener y un resistor R_p. El diodo Zener satisface la siguiente relación:

$$i_z = \frac{I_0}{1 - (v_z/v_{br})^m} \tag{10.39}$$

donde m, v_{br} e I_0 son parámetros. Considerando cinco secciones de la línea de transmisión, se obtienen las siguientes ecuaciones:

$$\begin{aligned}
\frac{\mathrm{d}x_1}{\mathrm{d}t} &= \frac{1}{L}\cdot v_{in} - \frac{R}{L}\cdot x_1 - \frac{1}{L}\cdot x_2 \\
\frac{\mathrm{d}x_2}{\mathrm{d}t} &= \frac{1}{C}\cdot x_1 - \frac{1}{C}\cdot x_3 \\
\frac{\mathrm{d}x_3}{\mathrm{d}t} &= \frac{1}{L}\cdot x_2 - \frac{R}{L}\cdot x_3 - \frac{1}{L}\cdot x_4 \\
\frac{\mathrm{d}x_4}{\mathrm{d}t} &= \frac{1}{C}\cdot x_3 - \frac{1}{C}\cdot x_5 \\
&\vdots \\
\frac{\mathrm{d}x_9}{\mathrm{d}t} &= \frac{1}{L}\cdot x_8 - \frac{R}{L}\cdot x_9 - \frac{1}{L}\cdot x_{10} \\
\frac{\mathrm{d}x_{10}}{\mathrm{d}t} &= \frac{1}{C}\cdot x_9 - \frac{1}{R_pC}\cdot (x_{10} - v_z)
\end{aligned} \tag{10.40}$$

Aquí, la variable de salida, v_z, es una variable algebraica que satisface:

$$\frac{1}{R_p}\cdot x_{10} - \left(\frac{1}{R_p} + \frac{1}{R_l}\right)\cdot v_z - \frac{I_0}{1-(v_z/v_{br})^m} = 0 \tag{10.41}$$

El sistema completo es evidentemente una DAE y cualquier método de los que vimos en los capítulos anteriores debería iterar sobre la Ec.(10.41) durante cada paso para encontrar el valor de v_z.

Si en cambio intentamos aplicar QSS o QSS3, deberíamos reemplazar x_i por q_i en las ecuaciones Ec.(10.40) y Ec.(10.41):

$$\begin{aligned}
\frac{\mathrm{d}x_1}{\mathrm{d}t} &= \frac{1}{L}\cdot v_{in} - \frac{R}{L}\cdot q_1 - \frac{1}{L}\cdot q_2 \\
\frac{\mathrm{d}x_2}{\mathrm{d}t} &= \frac{1}{C}\cdot q_1 - \frac{1}{C}\cdot q_3 \\
\frac{\mathrm{d}x_3}{\mathrm{d}t} &= \frac{1}{L}\cdot q_2 - \frac{R}{L}\cdot q_3 - \frac{1}{L}\cdot q_4 \\
\frac{\mathrm{d}x_4}{\mathrm{d}t} &= \frac{1}{C}\cdot q_3 - \frac{1}{C}\cdot q_5 \\
&\vdots \\
\frac{\mathrm{d}x_9}{\mathrm{d}t} &= \frac{1}{L}\cdot q_8 - \frac{R}{L}\cdot q_9 - \frac{1}{L}\cdot q_{10} \\
\frac{\mathrm{d}x_{10}}{\mathrm{d}t} &= \frac{1}{C}\cdot q_9 - \frac{1}{R_pC}\cdot (q_{10} - v_z)
\end{aligned} \tag{10.42}$$

$$\frac{1}{R_p}\cdot q_{10} - \left(\frac{1}{R_p} + \frac{1}{R_l}\right)\cdot v_z - \frac{I_0}{1-(v_z/v_{br})^m} = 0 \tag{10.43}$$

Notar que la Ec.(10.42) parece un QSS, excepto por la presencia de v_z. Esta variable v_z está acoplada a q_{10}, y por lo tanto, cada vez que haya una transición en q_{10}, deberíamos iterar en la Ec.(10.43) para encontrar el nuevo valor de v_z, y luego utilizar este nuevo valor en la Ec.(10.42).

Podemos construir un diagrama de bloques de las Ecs.(10.42)–(10.43) como sigue:

- Representamos la Ec.(10.42) mediante integradores cuantificados y funciones estáticas, tratando a v_z como si fuera una entrada externa.

- Agregamos un nuevo modelo atómico que compute v_z como función de q_{10}. En consecuencia, este bloque tiene a q_{10} como entrada y a v_z como salida.

Este último bloque estará en cargo de la iteración para determinar el nuevo valor de v_z, cada vez que q_{10} cambie. Ignorando errores de redondeo y asumiendo que el bloque de iteración pueda calcular v_z de forma correcta, el modelo DEVS resultante simulará exactamente la DAE definida por las Ecs.(10.42)–(10.43).

Notar que el bloque de iteración sólo se activa por los cambios en q_{10}. En todos los otros pasos (cuando cambian las otras nueve variables cuantificadas), no se realiza ninguna iteración. De esta manera, los métodos de QSS explotan intrínsecamente la raleza de las DAEs.

El sistema original de las Ecs.(10.40)–(10.41) es un caso particular del modelo implícito que vimos en el Capítulo 6 del curso:

$$\tilde{\mathbf{f}}(\dot{\mathbf{x}}_\mathbf{a}, \mathbf{x}_\mathbf{a}, \mathbf{u}) = 0 \tag{10.44}$$

En el caso de los métodos de QSS, conviene reescribir este sistema en la forma semi–explícita:

$$\begin{aligned} \dot{\mathbf{x}}_\mathbf{a} &= \mathbf{f}(\mathbf{x}_\mathbf{a}, \mathbf{u}, \mathbf{z}_\mathbf{a}) && \text{(10.45a)} \\ 0 &= \mathbf{g}(\mathbf{x}_{\mathbf{a}_\mathbf{r}}, \mathbf{u}_\mathbf{r}, \mathbf{z}_\mathbf{a}) && \text{(10.45b)} \end{aligned}$$

donde $\mathbf{z_a}$ es un vector de variables algebraicas de dimensión menor o igual que n. Los vectores $\mathbf{x}_{\mathbf{a}_\mathbf{r}}$ y $\mathbf{u_r}$ son versiones reducidas de $\mathbf{x_a}$ y $\mathbf{u}$, respectivamente, que expresan el hecho que no todas las variables de estado y de entrada influyen directamente en las ecuaciones implícitas del modelo.

Una forma directa de transformar el sistema de la Ec.(10.44) en el sistema de la Ec.(10.45) es definiendo $\mathbf{z_a} \triangleq \dot{\mathbf{x}}_\mathbf{a}$, de donde $\mathbf{x}_{\mathbf{a}_\mathbf{r}} = \mathbf{x_a}$, $\mathbf{u_r} = \mathbf{u}$, y $\mathbf{g} = \mathbf{f} - \mathbf{z_a}$.

Sin embargo, en muchos casos (como en el de la linea de transmisión), la dimensión de $\mathbf{x}_{\mathbf{a}_\mathbf{r}}$ y de $\mathbf{u_r}$ puede ser efectivamente mucho menor que n.

El uso de los métodos de QSS entonces transforma la Ec.(10.45) en:

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{f}(\mathbf{q}, \mathbf{u}, \mathbf{z}) && \text{(10.46a)} \\ 0 &= \mathbf{g}(\mathbf{q_r}, \mathbf{u_r}, \mathbf{z}) && \text{(10.46b)} \end{aligned}$$

y entonces las iteraciones sólo deben ser realizadas en los pasos en los que $\mathbf{q_r}$ o $\mathbf{u_r}$ cambien.

Todo lo que debemos agregar para simular una DAE con QSS o QSS2 será un modelo DEVS que reciba como entradas $\mathbf{q_r}$ o $\mathbf{u_r}$ y que tenga como salidas a $\mathbf{z}$. En los hechos, será un modelo muy similar al de una función estática sólo que deberá calcular las componentes z_i utilizando algún método iterativo.

La Figura 10.6 muestra el nuevo esquema de acoplamiento con el agregado de un nuevo modelo DEVS que calcula $\mathbf{z}$.

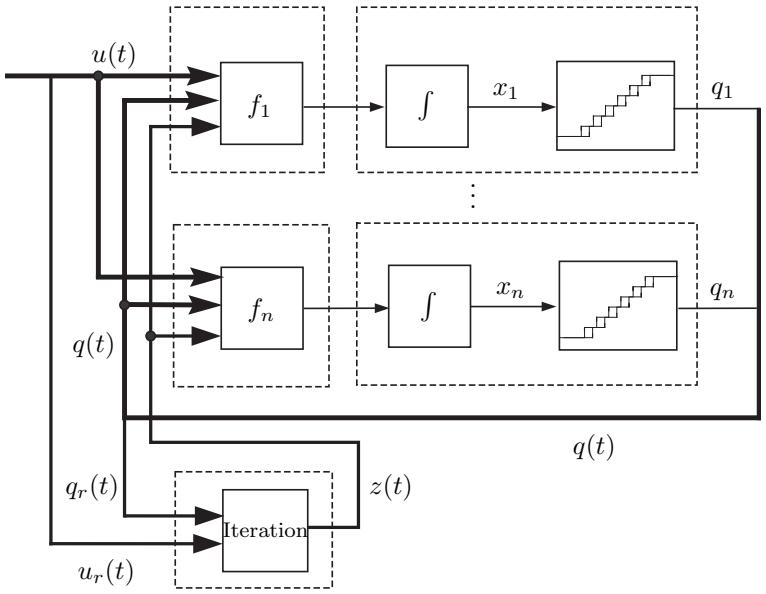

Figura 10.6: Esquema de acoplamiento para la simulación con QSS de la Ec.(10.45).

Aquí no desarrollaremos el modelo genérico de una función estática implícita. De todas formas, aclaramos que PowerDEVS tiene un modelo genérico que resuelve una restricción algebraica de la forma $g(\mathbf{v}, z) = 0$, tanto para el método de QSS como para QSS2 (y luego veremos que también lo hace para QSS3).

10.8. Manejo de Discontinuidades

En los métodos de tiempo discreto debíamos detectar exactamente los instantes en los cuales ocurrían las discontinuidades, ya que no podíamos integrar pasando a través de una discontinuidad. En el caso de los eventos temporales símplemente debíamos ajustar el paso para que este coincida con el tiempo de ocurrencia del evento. Frente a eventos de estado, en cambio, debíamos iterar para encontrar el instante preciso en el que ocurría dicho evento.

En el caso de los métodos de QSS, todos estos problemas desaparecerán. Para ver esto, comencemos analizando un ejemplo sencillo.

El circuito de la Fig. 10.7 es el *elevador* de tensión, que ya vimos en el Capítulo 7 del curso.

Recordemos que en este circuito, la llave se cierra y se abre periódicamente, y la tensión sobre la carga R_l depende de la relación entre el tiempo que está cerrada y el tiempo que permanece abierta. Habíamos visto también el siguiente modelo simplificado de la dinámica del circuito:

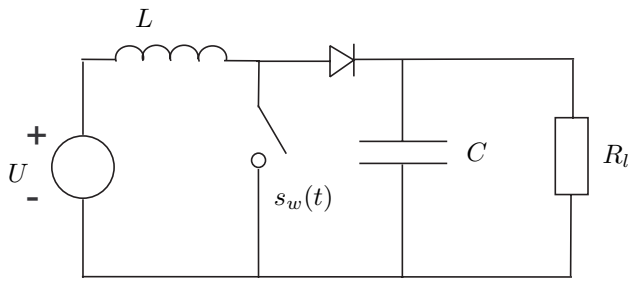

Figura 10.7: Circuito Elevador.

$$\dot{x}_1(t) = \frac{1}{L}(U - s_w(t) \cdot x_2(t)) \tag{10.47a}$$

$$\dot{x}_2(t) = \frac{1}{C}(s_w(t) \cdot x_1(t) - \frac{x_2(t)}{R_l}) \tag{10.47b}$$

donde x_2 es la corriente por la bobina, x_1 es la tensión en el capacitor, U es la tensión continua de alimentación y $s_w(t)$ vale 1 (llave abierta) o 0 (llave cerrada).

La señal $s_w(t)$ estaba representada por:

$$s_w(t) = \begin{cases} 0 & \text{si } t \cdot f - \text{int}(t \cdot f) < \delta \\ 1 & \text{en otro caso} \end{cases} \tag{10.48}$$

donde $0 \leq \delta \leq 1$ era el *ciclo activo* de la llave, es decir, la fracción de período que permanece cerrada y f es la frecuencia de conmutación. Notar que la llave permanece cerrada entre 0 y δ/f, abierta entre δ/f y $1/f$, nuevamente cerrada entre $1/f$ y $(1+\delta)/f$, etc.

Para aplicar el método de QSS o de QSS2, símplemente haremos lo siguiente:

$$\dot{x}_1(t) = \frac{1}{L}(U - s_w(t) \cdot q_2(t)) \tag{10.49a}$$

$$\dot{x}_2(t) = \frac{1}{C}(s_w(t) \cdot q_1(t) - \frac{q_2(t)}{R_l}) \tag{10.49b}$$

Mirando esta última ecuación, si nadie nos dice que $s_w(t)$ viene de un subsistema discreto, podríamos pensar que se trata de una simple entrada seccionalmente constante. De hecho, los métodos de QSS lo tratarán así. El hecho de que una señal sea seccionalmente constante era un problema en los métodos de tiempo discreto. QSS en cambio trata todas las señales como seccionalmente constantes.

Tanto los integradores como las funciones estáticas pueden recibir eventos en cualquier momento (de hecho, funcionan así aún con los sistemas sin discontinuidades). Por eso, una señal de entrada seccionalmente constante como $s_w(t)$ no trae ningún problema y no requiere ningún tratamiento especial.

Para corroborar esto, símplemente construimos el modelo PowerDEVS de la Ec.(10.49). El mismo puede verse en la Fig.10.8.

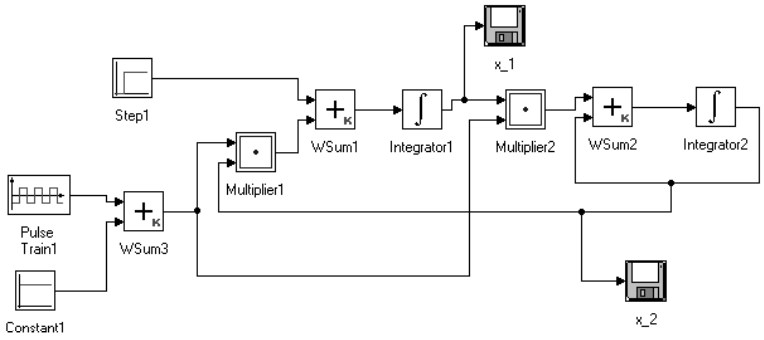

Figura 10.8: Modelo PowerDEVS del circuito elevador.

Para dicho modelo utilizamos los parámetros $C = 220\times10^{-6}$; $L = 150\times10^{-6}$; $R_l = 30$; $U = 5$; $f = 25000$ y $\delta = 0.63$. La simulación desde condiciones iniciales nulas con tiempo final $t_f = 0.1$ con el método de QSS2 usando un quantum $\Delta Q_i = 0.01$ en ambas variables da como resultado la gráfica que se muestra en la Fig.10.9.

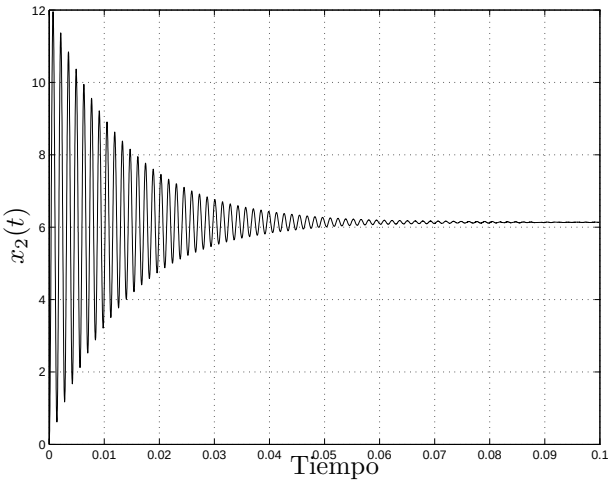

Figura 10.9: Tensión de salida en el circuito elevador.

El número de pasos fue de 5518 en el integrador que calcula x_1 y 4134 en el de x_2. Además, hubo 5000 eventos en la variable $s_w(t)$. Además de haber muy pocos pasos en comparación con la frecuencia a la que ocurren las discontiunidades, la principal ventaja de esta simulación es que no tuvimos que hacer ningún tipo de tratamiento de discontinuidades. Los métodos de QSS se las arreglan por sí solos.

Veamos ahora que ocurre en presencia de eventos de estado. Para eso, retomaremos el ejemplo de la pelotita que rebota contra el piso del Capítulo 7 del curso:

$$\dot{x}_1(t) = x_2(t) \tag{10.50a}$$

$$\dot{x}_2(t) = -g - s_w(t) \cdot \frac{1}{m}(k \cdot x_1(t) + b \cdot x_2(t)) \tag{10.50b}$$

donde

$$s_w = \begin{cases} 0 & \text{si } x_1(t) > 0 \\ 1 & \text{en otro caso} \end{cases} \tag{10.51}$$

Si aplicamos aquí QSS o QSS2, transformaremos la Ecs.(10.50) en:

$$\dot{x}_1(t) = q_2(t) \tag{10.52a}$$

$$\dot{x}_2(t) = -g - s_w(t) \cdot \frac{1}{m}(k \cdot q_1(t) + b \cdot q_2(t)) \tag{10.52b}$$

En cuanto a la Ec.(10.51) podríamos dejarla como está (y en ese caso calcularíamos $s_w(t)$ a partir de la variable de estado x_1), o podríamos cambiarla por:

$$s_w = \begin{cases} 0 & \text{si } q_1(t) > 0 \\ 1 & \text{en otro caso} \end{cases} \tag{10.53}$$

lo que dice que calculamos la variable $s_w(t)$ a partir de la variable cuantificada q_1. Cualquiera de las dos opciones es válida.

El único cambio respecto a lo que veníamos haciendo es que ahora, $s_w(t)$ es una función discontinua del estado. Todo lo que necesitamos es un bloque que tenga como entrada q_1 (o x_1) y como salida $s_w(t)$. El resto son simples integradores cuantificados y funciones estáticas.

El bloque que calcula $s_w(t)$ es de hecho una función estática, pero cuando la entrada sea seccionalmente lineal o parabólica, la salida será seccionalmente constante y los instantes de las discontinuidades en la salida serán distintos a los de las discontinuidades de la entrada. Por lo tanto, dicho bloque deberá tener la capacidad de detectar sus propias discontinuidades (o sea, los cruces por 0 de q_1 o de x_1).

La Fig.10.10 muestra un modelo PowerDEVS del sistema. En dicho modelo, aparece una función discontinua (Comparador), que es el modelo DEVS correspondiente a la Ec.(10.53). Este bloque compara dos entradas (pueden ser hasta seccionalmente parabólicas) y produce eventos cuando las mismas se cruzan. En nuestro caso, una es constante e igual a cero (altura del piso) y la otra es la altura de la pelotita (q_1). La salida en este caso es entonces la variable $s_w(t)$.

El bloque Comparador que mencionamos, símplemente calcula en forma exacta el tiempo para el próximo cruce resolviendo una ecuación cuadrática (en caso que alguna entrada sea seccionalmente parabólica, sino es más simple aún) cada vez que recibe un nuevo evento.

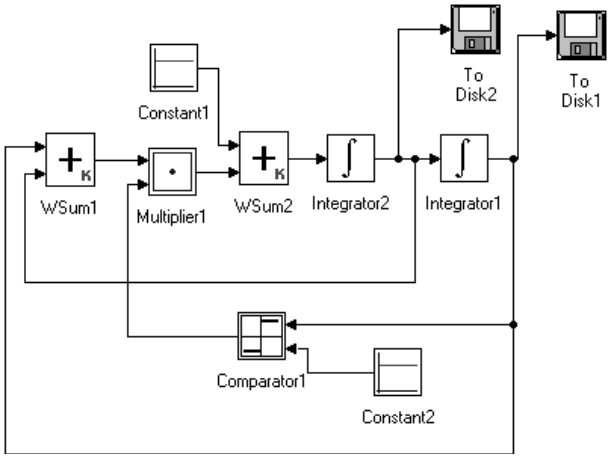

Figura 10.10: Modelo PowerDEVS de la pelotita rebotando.

Como podemos ver en este ejemplo, el manejo de las discontinuidades en los métodos de QSS se hace en la propia implementación de las funciones estáticas. El resto de los bloques (integradores cuantificados, generadores, y demás funciones estáticas) no se modifican para nada.

PowerDEVS cuenta con varios bloques *híbridos* que manejan discontinuidades y que pueden utilizarse con los métodos de QSS.

Para corroborar el funcionamiento correcto de este modelos simulamos el sistema con el método de QSS2 durante 5 segundos. En este caso, consideramos parámetros $m = 1$, $b = 30$, $k = 1 \times 10^6$ y $g = 9.81$. Las condiciones iniciales fueron $x(0) = 1$ y $v(0) = 0$. Utilizamos un quantum $\Delta Q_i = 1 \times 10^{-4}$ en ambas variables. Los resultados se muestran en la Fig.10.12.

Esta manera tan simple y eficiente de tratar las discontinuidades es de hecho la principal ventaja práctica de los métodos de QSS.

10.9. Problemas Propuestos

[P10.1] Cota de Error en Sistemas Lineales

Dado el sistema Lineal y Estacionario:

$$\begin{aligned} \dot{x}_{a_1} &= x_{a_2} \\ \dot{x}_{a_2} &= -2 \cdot x_{a_1} - 3 \cdot x_{a_2} \end{aligned}$$

con condiciones iniciales, $x_{a_1}(0) = x_{a_2}(0) = 1$.

1. Calcular la solución analítica, $x_{a_1}(t)$ y $x_{a_2}(t)$.

2. Obtener una solución aproximada, $x_1(t)$ y $x_2(t)$, con el método de QSS usando $\Delta Q = 0.05$ en ambas variables.

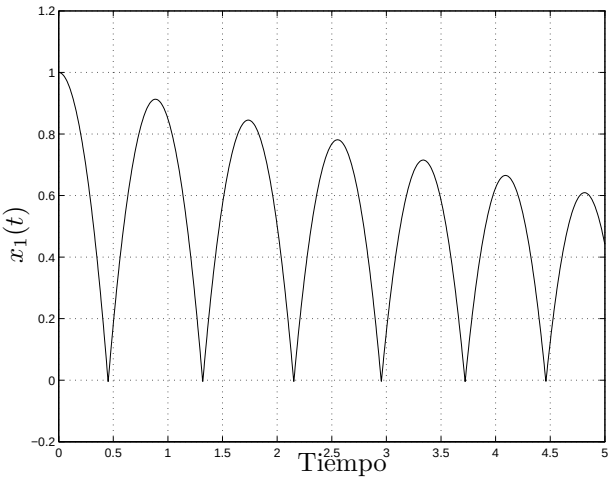

Figura 10.11: Simulación con QSS2 de la pelotita rebotando.

3. Dibujar las trayectorias de error, $e_i(t) = x_i(t) - x_{a_i}(t)$, y compararlas con la cota teórica de error dada por la Ec.(10.27).

[P10.2] Error de Cuantificación de Entrada
Demostrar la validez de la Ec.(10.28).

[P10.3] Función Lineal Estática en QSS2
Obtener el modelo DEVS (o PowerDEVS) de una función estática lineal:

$$f_i(\mathbf{z}) = \sum_{j=1}^{l} a_{i,j} \cdot z_j \tag{P10.3a}$$

que tenga en cuenta los valores y sus pendientes.

[P10.4] Circuito Elevador con Diodos
Repetir el Problema [P7.5] utilizando el método de QSS2.

[P10.5] Neurona Pulsante
Repetir el Problema [P7.6] utilizando el método de QSS2.

Bibliografía

[1] François Cellier. *Continuous System Modeling*. Springer, New York, 1991.

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

[3] Donald Erdman. Study of Kinetics: The Estimation and Simulation of Systems of First-Order Differential Equations. En *SUGI Proceedings*, 1995.

[4] Eugene Izhikevich. Simple Model of Spiking Neurons. *IEEE Transactions on Neural Networks*, 14(6):1569–1572, 2003.

[5] H. Khalil. *Nonlinear Systems*. Prentice-Hall, New Jersey, 3rd edición, 2002.

[6] Alexei Sharov. Quantitative Population Ecology. On-line Lecture Course. Department of Entomology Virginia Tech, Blacksburg, VA, 1996. http://www.gypsymoth.ento.vt.edu/∼sharov/PopEcol/popecol.html.