Simulación Eficiente de Sistemas Rígidos y Conmutados

Ing. Franco Di Pietro

Tesis presentada para el cumplimiento parcial de los requisitos para obtener el título de

Doctor en Ingeniería

Director: Dr. Ernesto Kofman

Codirectora: Dra. Mónica Romero

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

II

Certifico que el trabajo incluido en esta tesis es el resultado de tareas de investigación originales y que no ha sido presentado para optar a un título de posgrado en ninguna otra Universidad o Institución.

Ing. Franco Di Pietro

Resumen

En esta tesis se desarrollan nuevos métodos de integración para ecuaciones diferenciales ordinarias. Estos surgen como producto de combinar las ideas de los métodos clásicos con lo métodos basados en cuantificación. Como resultado se proponen diferentes métodos de integración.

Por un lado, se plantea una mejora a los ya existentes métodos de estados cuantificados linealmente implícitos. Esta mejora supone una ampliación en el campo de aplicación de los mismos, permitiendo integrar estructuras más generales que en sus versiones originales. Para ello, ante determinadas situaciones problemáticas en lo que respecta a tiempos de simulación, se dan pasos utilizando métodos clásicos que en la búsqueda de evitar dichas situaciones.

Además, se presenta un método mixto que combina los enfoques clásico y el basado en cuantificación, de modo que se divide un dado sistema de dos subsistemas. Cada uno de ellos será simulado con un enfoque distinto, a la vez que podrán interactuar para comportarse como el sistema original. De este modo, pueden aprovecharse las ventajas que cada enfoque pueda tener para distintas partes de un sistema. Este enfoque resulta particularmente interesante para la simulación de sistemas que involucran más de un dominio físico.

La eficiencia de estos métodos es verificada haciendo comparaciones entre ejemplos simulados con los mismos y con distintas métodos, tanto clásicos como basado en cuantificación. En particular, se destacan las ventajas que poseen los algoritmos desarrollados con respecto a los métodos clásicos de integración.

Abstract

In this thesis, new integration methods are developed for ordinary differential equations. These arise as a product of combining the ideas of classic methods with methods based on quantification. As a result, different integration methods are proposed.

On the one hand, an improvement is proposed to the existing methods of linearly implicit quantized states. This improvement implies an extension in their field of application, allowing them to integrate more general structures than in their original versions. For this, in certain problematic situations with regard to simulation perfomances, steps are taken using classic methods that seek to avoid such situations.

Moreover, we present a mixed method that combines classic and quantification-based approaches, so that a given system is partitioned into two subsystems. Each one of them will be simulated with a different approach, at the same time that they will be able to interact in order to behave like the original system. In this way, you can use the advantages that each approach may have for different parts of a system. This approach is particularly interesting for the simulation of systems that involve more than one physical domain.

The efficiency of these methods is verified by comparisons between simulated examples using them and different methods, both classic and based on quantification. In particular, the advantages of using these methods when simulating switched power electronic circuits is noticed.

Índice general

Capítulo 1

Introducción

La simulación digital de sistemas dinámicos es un área de desarrollo permanente en los ámbitos teórico y de aplicación. Por un lado, la enorme evolución de las tecnologías informáticas (hardware y software) en las ultimas décadas motivó la profusión de una gran cantidad de métodos numéricos de resolución de Ecuaciones Diferenciales Ordinarias (EDOs), [1, 2, 3]. Por otro lado, la creciente complejidad de los modelos que describen los sistemas de ingeniería modernos y la necesidad de su análisis por simulación plantea cada día nuevos desafíos a la teoría de los métodos numéricos. Entre los numerosos campos con problemas abiertos, los que se tratarán en esta Tesis son el de la simulación eficiente de sistemas rígidos (normalmente denominados por su nombre en inglés *stiff*) y EDOs discontinuas.

Muchos sistemas dinámicos de la práctica, tanto en las ciencias como en la ingeniería, son stiff. Esto es, tienen matrices Jacobianas con autovalores muy separados entre sí sobre el eje real negativo del plano complejo. La integración de estos sistemas mediante métodos numéricos tradicionales de discretización temporal requiere de la utilización de algoritmos implícitos, ya que todos los métodos explícitos deben necesariamente restringir excesivamente el paso de integración para garantizar estabilidad numérica. Esto se debe a que los métodos explícitos exhiben dominios de estabilidad que se cierran en el semiplano izquierdo del plano complejo [4], y la única manera de evitar que la estabilidad numérica limite el paso de integración es con dominios de estabilidad cerrados en el semiplano derecho, característica que sólo se observa en algunos métodos implícitos. Como contrapartida, los métodos implícitos tienen una mayor carga computacional que los explícitos ya que requieren de algoritmos iterativos en cada paso para despejar el siguiente valor. Esto es además inaceptable en aplicaciones de tiempo real

dado que no puede garantizarse cuanto tiempo demandará la convergencia de las iteraciones (típicamente se utilizan iteraciones de Newton).

Un enfoque esencialmente distinto al de los métodos de integración clásica surge al reemplazar la discretización del tiempo por la cuantificación de las variables de estado. Esta idea dio lugar a los *métodos de integración por cuantificación*, que aproximan las ecuaciones diferenciales ordinarias por *sistemas de eventos discretos*.

El primero de estos métodos es el de QSS1 (Quantized State Systems) [5], que realiza una aproximación de primer orden. En base a principios similares, se desarrollaron también métodos de segundo (QSS2) [6] y de tercer orden (QSS3) [7], que permiten obtener una mejor precisión sin incrementar mucho el número de cálculos.

Los métodos de QSS tienen propiedades teóricas muy fuertes (estabilidad y existencia de cota de error global calculable para sistemas lineales) y presentan grandes ventajas al simular sistemas discontinuos [8]. Sin embargo, no son apropiados para la simulación de sistemas stiff, debido a la aparición de oscilaciones de alta frecuencia [4]. Lo que ocurre es que, al igual que en cualquier método clásico explícito, el paso de integración se reduce excesivamente para mantener la estabilidad numérica.

Posteriormente, fueron desarrollados siguiendo la idea de los métodos QSS, una familia de métodos numéricos para ecuaciones diferenciales ordinarias (EDOs) que permitan simular de manera eficiente sistemas stiff, donde la rigidez debe estar dada por ciertas condiciones estructurares, en particular debido únicamente a la presencia de términos grandes en la diagonal principal de la matriz Jacobiana del sistema. Estos métodos, denominados métodos QSS linealmente implícitos (LIQSS), aunque fueron formulados para EDOs pueden ser también aplicados a sistemas híbridos y a *Ecuaciones Diferenciales Algebraicas*.

En esta tesis, se desarrollan métodos que logran integrar eficientemente sistemas stiff con estructuras más generales. Inicialmente resolviendo subsistemas de 2×2 variables, dentro de un sistema de $n \times n$, y luego generalizando para el caso de $m \times m$ variables, donde $m \leq n$. Además de definir los métodos y estudiar sus propiedades teóricas (que resultan similares a las de los métodos QSS), se verán detalles de su implementación.

Otro punto muy importante que se trata en esta tesis es el de las aplicaciones prácticas de la simulación de sistemas discontinuos stiff. En particular, se mostrará cómo estos métodos obtienen ventajas sobre los métodos tradicionales en la simulación de modelos de circuitos de electrónica conmutada cuando en los mismos se modela con más realismo los elementos de conmutación, así como en presencia de elementos parásitos. Estos tipos de modelos

resultan ser stiff y presentan en general muchas discontinuidades. Además se mostrarán los resultados de simular sistemas multidominio, donde los métodos propuestos resultan notablemente eficientes.

1.1. Organización de la Tesis

Este primer capitulo introductorio brinda una descripción de la tesis completa, no sólo enumerando los resultados sino también intentado relacionarlos con el estado del arte actual.

El capítulo 2 presenta en las secciones 2.1 a 2.6 un breve resumen de los métodos clásicos de tiempo discreto y una descripción de sus principios de funcionamiento. A través del mismo se pretende poner en evidencia cuales son los inconvenientes que presenta este enfoque para la simulación de sistemas stiff y cuales son las soluciones que brinda para poder así comparar con los métodos que se desarrollarán en la Tesis. Además, muchos de los conceptos aquí introducidos, repensados para los métodos QSS, fueron los que inspiraron los métodos que se desarrollan en esta tesis. Tras esto, en la sección 2.7 se dan varias definiciones de sistemas stiff y a continuación se da un pequeño resumen de los métodos presentados a lo largo del capítulo indicando si son adecuados para simular sistemas stiff, la causa e inconvenientes que presentan.

En el capitulo 3, en la secciones 3.1 a 3.2 presentan los métodos por cuantificación de estados QSS (por sus siglas en inglés, *Quantized State Systems*), siendo estos los que dieron origen a los métodos que se desarrollaron en este trabajo. En la sección 3.3 se muestran las ventajas de los métodos QSS en el manejo de discontinuidades respecto a los métodos de tiempo discreto presentados en el segundo capítulo y en la sección 3.4 se presentan las propiedades teóricas de los mismos. Para poder mostrar las propiedades teóricas se explica en primer lugar la relación entre los métodos QSS y la teoría de sistemas perturbados. Luego, en la sección 3.5 se presentan las métodos QSS linealmente implícitos, LIQSS. Posteriormente, se ve en la sección 3.6 el comportamiento tanto de los métodos QSS como de los LIQSS a la hora de simlar sistemas stiff, mostrando los inconvenientes que pueden surgir en cada caso. Finalmente, en la sección 3.7 se describe brevemente la implementación de toda esta familia de métodos.

El capítulo 4 comienza mostrando en la sección 4.1, a través de un ejemplo, los inconvenientes que tienen los métodos LIQSS en la simulación de sistemas stiff, cuando la rigidez se debe a la presencia de términos grandes fuera de la diagonal principal de la matriz Jacobiana del sistema. A partir de

este punto, el resto de la Tesis contiene los resultados originales del trabajo. En primer lugar presenta la idea básica del método a través de un ejemplo para facilitar la comprensión de la definición formal del mismo dada en la sección 4.2. El desarrollo del método mLIQSS1 termina en la sección 4.3 donde se presenta el algoritmo detallando su funcionamiento. Por último, en la sección 4.4 se presentan dos ejemplos que permiten comparar los resultados obtenidos con el algoritmo modificado mLIQSS1 con los que se obtienen utilizando la versión original del mismo así como con algunos de los métodos clásicos introducidos en el capítulo 2.

A pesar de los buenos desempeños mostrados por el método mLIQSS1, y como era de esperarse al tratarse de un algoritmo de primer orden, no puede alcanzarse una buena precisión sin aumentar significativamente el número de pasos. Por esto se torna necesario formular aproximaciones de orden superior.

En el capítulo 5, en la sección 5.1 se presenta una versión mejorada del algoritmo original LIQSS2, justo a su algoritmo. Esta versión mejorada no resuelve la aparición de oscilaciones espurias dada por la interacción entre pares de variables, pero muestra un mejor desempeño que la versión propuesta originalmente en los casos en que estas oscilaciones no existan. Luego, en la sección 5.2 se presenta formalmente el método modificado, así como el algoritmo que implementa la idea del mismo. Posteriormente, en la sección 5.3 se verifica que las versiones modificadas propuestas de los algoritmos conservan las fuertes propiedades de los métodos originales que fueron convenientemente explicadas en la sección 3.4. Luego, en la sección 5.4 se describe brevemente la implementación de los algoritmos modificados. Finalmente, en la sección 5.5 se presentan resultados. En primera instancia, se muestra un ejemplo donde se ven las mejoras obtenidas debido a la mejora propuesta al método LIQSS2 en la sección 5.1. Luego se evalúa el desempeño del método mLIQSS2 en diversos campos de aplicación, en los cuales se verifican las condiciones de rigidez establecidas para el correcto funcionamiento de estos métodos, y en los cuales lo métodos originales fallan.

Luego, en el capítulo 6, se ve la aplicación de los métodos propuestos en los capítulos anteriores en el campo de los convertidores de electrónica conmutada en presencia de elementos parásitos. Hasta este punto, las simulaciones realizadas no contemplaron la presencia de dichos elementos, y los mismos pueden variar la estructura del mismo, condicionando así la eficiencia de los métodos mLIQSS. En la sección 6.1 se da una introducción a la problemática que presenta la presencia de elementos parásitos en el ámbito de la simulación. Luego, en la sección 6.2 se describe la rigidez desde un punto de vista circuital, así como la consecuencia de la misma al aplicar

los métodos QSS. Además se presenta un modo sistémico de analizar la posibilidad de simular eficientemente un dado circuito a partir de un simple análisis del circuito utilizando esta familia de métodos. Teniendo esto presente, en las secciones 6.3 y 6.4 se presenta el análisis de dos convertidores en presencia de elementos parásitos, así como el como la comparación de el desempeño de métodos mLIQSS, sus versiones originales y algunos métodos clásicos.

A partir de las limitaciones presentadas en los capítulos precedentes, surge la idea descripta en el capítulo 7, en el que se tratan los métodos mixtos. Para comenzar, en la sección 7.1 se da una introducción respecto a la motivación de desarrollar un métodos mixtos que combinen métodos clásicos con QSS. Luego, en la sección 7.2 se establece la definición formal de estos métodos, y se demuestran importantes propiedades teóricas como lo son la convergencia y estabilidad de estos métodos propuestos. En la sección 7.3 se describe la implementación de estos métodos propuestos, puntualmente en lo que refiere a un método de primer orden (QMM1) y uon de segundo orden (QMM2). Finalmente, en la sección 7.4 vemos ejemplos donde se comparan los resultados de simular algunos sistemas con métodos clásicos por un lado, con métodos de la familia QSS por otro, y con métodos mixtos que combinan los dos anteriores.

La tesis concluye con el capítulo 8, donde se presentan algunos problemas abiertos en los que se está trabajando actualmente y las conclusiones generales del trabajo realizado.

1.2. Contribuciones originales

A partir del capítulo 4, y hasta el final de la tesis, la mayor parte de los resultados son originales.

La principal contribución es el desarrollo de cuatro nuevos métodos de integración numérica para ecuaciones diferenciales ordinarias stiff y todo el trabajo hecho alrededor de los mismos: elaboración de algoritmos, estudio de las propiedades teóricas, implementación en un entorno de simulación, construcción de ejemplos y comparación de resultados con métodos clásicos de integración.

Siguiendo la idea de los métodos de integración cuantificados, es decir, reemplazando la discretización temporal por la cuantificación de los estados, estos métodos realizan aproximaciones de primer(mLIQSS1 y QMM1) y segundo orden(mLIQSS2 y QMM2) permitiendo simular eficientemente diversos tipos de sistemas stiff. La característica sobresaliente de los méto-

dos desarrollados es que permiten simular eficientemente sistemas stiff con estructuras más generales que las que permiten los métodos actuales.

1.3. Trabajos Relacionados y Relevancia de los Resultados

Los trabajos que tienen cierta relación con esta Tesis se pueden clasificar en dos categorías.

Por un lado, hay trabajos que intentan solucionar el problema de integrar numéricamente sistemas stiff de manera eficiente haciendo uso de los principios de los métodos clásicos de tiempo discreto. Por otro lado, hay trabajos que si bien no están enfocados en la simulación de sistemas stiff, desarrollan algoritmos similares a los que se proponen en esta tesis para la simulación de sistemas continuos cuantificando las variables de estado en lugar de la variable tiempo.

Con respecto al primer grupo, un gran número de libros y artículos de revistas se han ocupado del estudio de métodos numéricos para la integración de ecuaciones diferenciales stiff en las últimas décadas. Se han propuesto multitud de métodos buscando buenas propiedades de estabilidad lineal y no lineal. Todos estos métodos son implícitos. Se puede encontrar gran parte de este material recopilado en los libros [9] y [10].

Los más usados en la práctica son aquellos basados en métodos lineales multipaso, especialmente los métodos BDF [11, 1], por ser muy eficientes para un gran número de problemas de este tipo y los métodos Runge-Kutta implícitos [12], debido a sus buenas propiedades de estabilidad (A-estabilidad, L-estabilidad y B-estabilidad entre otras). Sin embargo, en todos los casos es necesario resolver un sistema no lineal de ecuaciones algebraicas en cada paso de la integración. Este problema es más grave en los métodos de Runge–Kutta implícitos ya que el sistema de ecuaciones a resolver resulta de mayor dimensión, lo que hace estas fórmulas poco eficientes cuando la dimensión del problema es grande.

Con el fin de reducir el coste computacional que se requiere en cada paso de la integración numérica, también se han desarrollado métodos linealmente implícitos, eliminando de este modo la necesidad de resolver sistemas no lineales de ecuaciones algebraicas.

De entre los muchos métodos de este tipo cabe destacar los conocidos como métodos Rosenbrock [13] así como los métodos Rosenbrock-Wanner o métodos Rosenbrock modificados, [14, 15, 16]. Estos métodos tienen el inconveniente de que es necesario evaluar la matriz Jacobiana en cada paso,

lo que les hace poco competitivos cuando esta evaluación es costosa desde un punto de vista computacional.

Por otro lado, algunos casos particulares de sistemas stiff pueden tratarse a través de métodos especiales de integración numérica. Entre estos encontramos los métodos multirate, que permiten utilizar distintos tamaños de paso de integración en cada variable de estado, reduciendo así la cantidad de cálculos realizados [17, 18]. También existen métodos denominados mixtos, que utilizan fórmulas explícitas para ciertas componentes del sistema y fórmulas implícitas para las restantes [4].

En lo que respecta al segundo grupo, las primeras ideas y definiciones se deben a Bernard Zeigler. A fines de los noventa presentó un enfoque esencialmente distinto al de los métodos de integración clásica que surge al reemplazar la discretización del tiempo por la cuantificación de las variables de estado [19]. Esta idea dio lugar a los métodos de integración por cuantificación, que aproximan las ecuaciones diferenciales ordinarias por sistemas de eventos discretos en términos del formalismo DEVS [20].

El primero de estos métodos es el de QSS1 (Quantized State Systems) [5], que realiza una aproximación de primer orden. En base a principios similares, se desarrollaron también métodos de segundo orden (QSS2) [6] y de tercer orden (QSS3) [7], que permiten obtener una mejor precisión sin incrementar mucho el número de cálculos. Además, los métodos QSS cuentan con una forma de control de error relativo basado en el uso de cuantificación logarítmica [21].

Los métodos de QSS tienen propiedades teóricas muy fuertes (estabilidad y existencia de cota de error global calculable para sistemas lineales) y presentan grandes ventajas al simular sistemas discontinuos [8]. Sin embargo, no son apropiados para la simulación de sistemas stiff, debido a la aparición de oscilaciones de alta frecuencia [10]. Lo que ocurre es que, al igual que cualquier método clásico explícito, el paso de integración se reduce excesivamente para mantener la estabilidad numérica.

Esta tesis puede ser vista como una continuación de los trabajos de Ernesto Kofman y Gustabo Migoni en el tema. Aquí, se señala el problema que presentan los métodos para simular sistemas stiff en los cuales la rigidez se debe a la presencia de términos grandes fuera de la diagonal principal de la matriz Jacobiana del sistema.

En primera medida, se implementó el método mLIQSS1, que combina el método LIQSS1 con el método clásico Backwar Euler. Este último se utiliza para resolver subsistemas de 2×2 en los casos en los que se prevea la presencia de oscilaciones espurias debido a la interacción de dos variables de estado. La idea detrás de este método sirvió para el posterior desarrollo

del método de segundo orden mLIQSS2. Para ello, primero se realizó una mejora sobre el método LIQSS2 original, que mejora aún más los resultados obtenidos con el mismo a la hora de simular sistemas stiff en los que la rigidez se debe a la presencia de términos grandes en al diagonal principal de la matriz Jacobiana. Estos dos métodos comparten todas sus propiedades teóricas con los métodos originales, dando lugar así a métodos que resultan muy eficientes a la hora de simular sistemas discontinuos con una rigidez más general que la alcanzada por lo métodos de base.

Luego, inspirados en las ideas de los métodos mixtos clásicos, que combinana métodos implícitos con métodos implícitos, y los métodos mLIQSS, surgen los métodos mixtos basados en cuantificación QMM. Estos combinan los métodos clásicos con los métodos LIQSS para simular distintas partes de un sistema, aprovechando así las bondades de cada metodología en las partes en las que cada una resulte conveniente. Además se demostrar propiedades teóricas sumamente importantes en lo que respecta a los métodos numéricos, como lo son convergencia y estabilidad.

1.4. Publicaciones de Apoyo

La mayor parte de los resultados incluidos en esta tesis ya fueron publicados en revistas y en memorias de conferencias.

El primer resultado fue la definición del método mLIQSS1, la implementación del mismo en un entorno de simulación, la construcción de modelos a modo de ejemplo y la obtención de resultados de simulación, con su correspondiente comparación con el método LIQSS1 original así como con algunos métodos clásicos. Estos resultados fueron publicados en la principal conferencia de mundo de la disciplina de Simulación [22], la *Winter Simulation Conference*.

Posteriormente, se ampliaron los resultados obtenidos con la introducción del método mLIQSS2, así como con la mejora del método LIQSS2. Estos resultados fueron publicados en la revista *Simulation. Transactions of the SCS* [23].

Además, los resultados de aplicar los métodos mLIQSS a la simulación de circuitos de electrónica conmutada en presencia de elementos parásitos fue publicada en una conferencia local [24].

Hay también un artículo en preparación para una revista internacional sobre los métodos mixtos basados en cuantificación de hasta segundo orden.

Capítulo 2

Métodos Clásicos de Integración y Sistemas Stiff

En este capítulo, basado principalmente en [10], se presentará una muy breve descripción de algunos de los métodos de integración de tiempo discreto que se mencionan en esta Tesis. En cada caso se resaltará sus ventajas y desventajas para la simulación de sistemas rígidos (usualmente denominados por su nombre en inglés, *stiff*).

Además, se introducirán varias definiciones de sistemas stiff a lo largo de este capítulo. Algunas de ellas, formuladas a partir del comportamiento de las simulaciones al utilizar determinados métodos de tiempo discreto.

Si bien este capítulo no trata sobre los métodos de integración por cuantificación (sobre los que se basa esta Tesis), los métodos aquí tratados se utilizan muchas veces para ser comparados con los métodos de integración desarrollados en esta Tesis. Además, conocer los principios de los métodos de tiempo discreto así como también sus ventajas y desventajas permite una mejor comprensión del motivo por el cual en determinados casos son realmente ventajosos los métodos desarrollados en esta Tesis así como también el modo en que se llegó al desarrollo de los mismos.

2.1. Conceptos básicos de integración numérica

En esta ección se introducirán caracteristicas generales de todos los métodos de tiempo discreto, así como también herramientas y conceptos que se utilizarán a lo largo del capítulo.

2.1.1. Principios de integración numérica

Los métodos de integración surgen como una herramienta que permite la simulación de modelos de sistemas continuos ya que por lo general resulta muy difícil encontrar soluciones analíticas de los mismos.

Para poder comprender el principio en que se basan todos los métodos de integración, analicemos en forma general la aproximación que realizan los métodos de integración del 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).

Denominando t^* al instante de tiempo en torno al cual se aproxima la trayectoria mediante una serie de Taylor, y sea t^*+h el instante de tiempo en el cual se quiere 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 Taylor que consideran para la aproximación.

2.1.2. Precisión de la aproximación y orden de un método

La precisión con la que se aproximan las derivadas de orden superior debe estar acorde al número de términos de la serie de Taylor 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.

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 buscar una solución de compromiso entre ambos factores.

2.1.3. Integración 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* (FE).

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

La simulación utilizando el método de FE 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*.

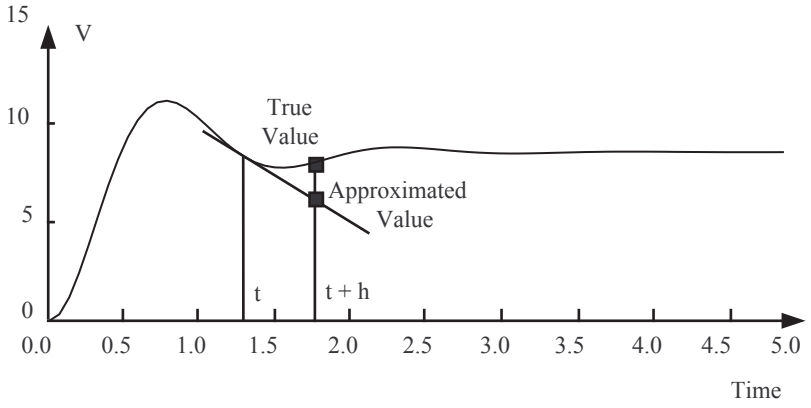


Figura 2.1: Integración numérica utilizando Forward Euler.

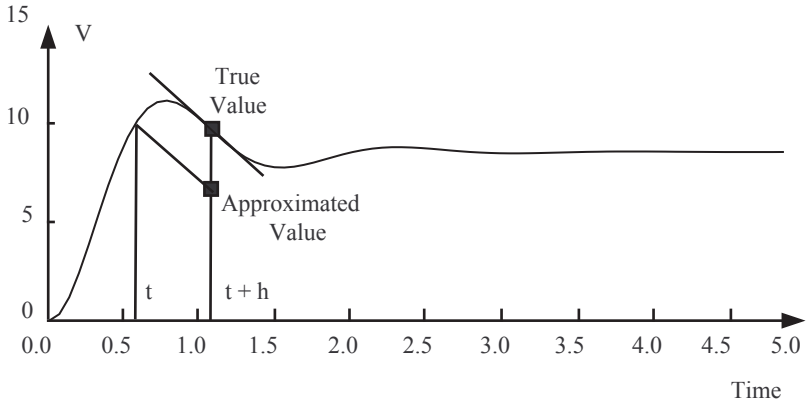


Figura 2.2: Integración numérica utilizando Backward Euler.

La Figura 2.2 muestra un esquema con una pequeña modificación.

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 del valor actual y del 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.

2.1.4. El Dominio de Estabilidad Numérica

Para poder definir los denominados dominios de estabilidad numérica, considerar nuevamente 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:

$$\text{Real}\{\text{Eig}(\mathbf{A})\} = \text{Real}\{\lambda\} < 0{,}0 \tag{2.9}$$

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

Aplicando 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), se obtiene:

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

que puede reescribirse en forma más compacta como:

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

donde $\mathbf{I^{(n)}}$ 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

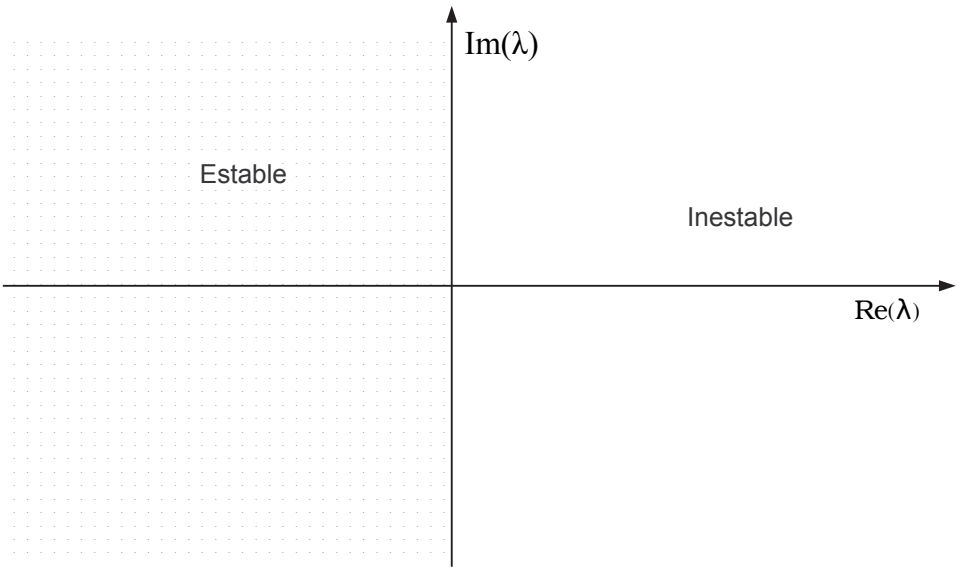


Figura 2.3: Dominio de estabilidad analítica.

que se hace es indexar el tiempo, es decir, k se refiere al k–ésimo paso de integración.

Lo que se hizo 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) se puede concluir que todos los autovalores de $\mathbf{A}$ multiplicados por el paso de integración h deben caer en un círculo de radio 1,0 alrededor del punto $-1{,}0$.

Se dice que un sistema lineal y estacionario de tiempo continuo integrado con un método dado 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.

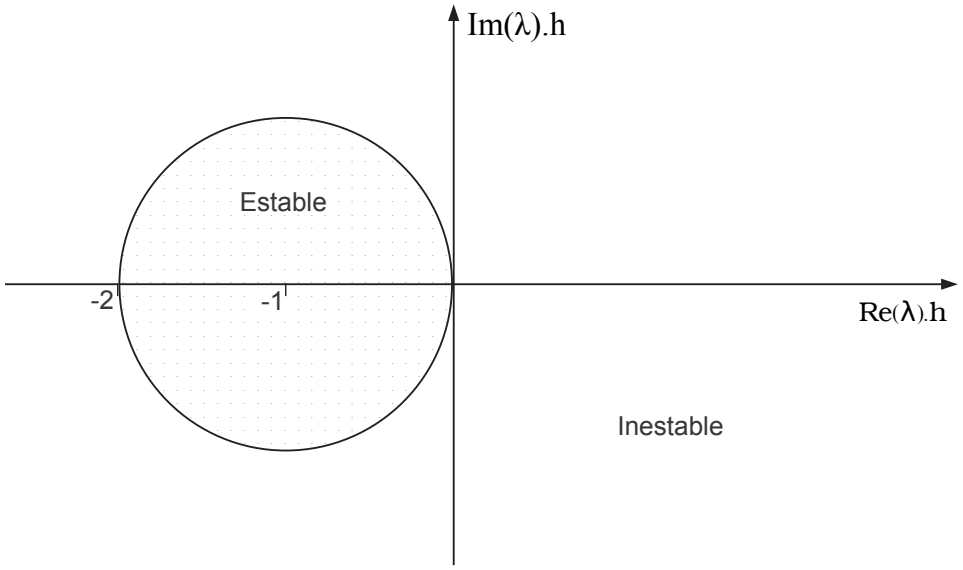


Figura 2.4: Dominio de estabilidad numérica de Forward Euler.

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

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

En el caso 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:

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

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

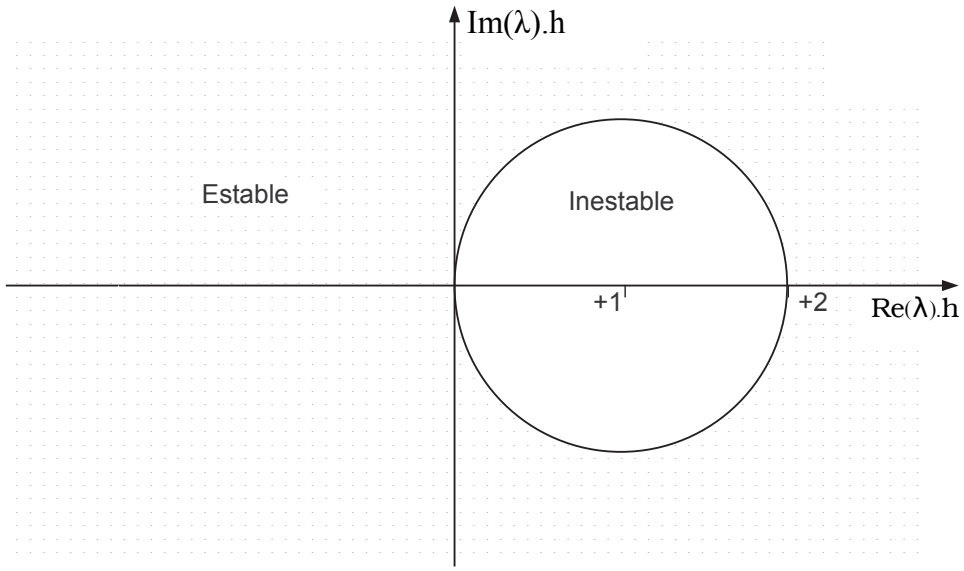


Figura 2.5: Dominio de estabilidad de Backward Euler.

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.

2.1.5. 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 se 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, se necesita algún procedimiento iterativo.

Un 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 2.6.

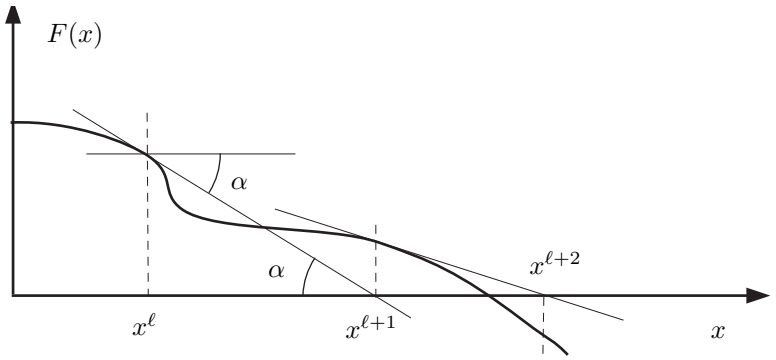


Figura 2.6: Iteración de Newton.

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

$$\tan\ \alpha = \frac{\partial \mathcal{F}^\ell}{\partial x} = \frac{\mathcal{F}^\ell}{x^\ell - x^{\ell+1}} \tag{2.18}$$

Entonces:

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

A modo de ejemplo se muestra a continuación como aplicar esta técnica para iterar en el método de BE aplicado a un sistema no lineal escalar, donde, en un punto x_k se tiene

$$\dot{x}_{k+1} = f(x_{k+1}, t_{k+1}) \tag{2.20}$$

por lo que al aplicar el algoritmo de BE

$$x_{k+1} = x_k + h \cdot \dot{x}_{k+1} \tag{2.21}$$

queda:

$$x_{k+1} = x_k + h \cdot f(x_{k+1}, t_{k+1}) \tag{2.22}$$

o

$$x_k + h \cdot f(x_{k+1}, t_{k+1}) - x_{k+1} = 0 \tag{2.23}$$

La Ecuación (2.23) está en la forma adecuada para aplicar la iteración de Newton. Aquí, la variable desconocida es x_{k+1}. Luego,

$$x_{k+1}^{\ell+1} = x_{k+1}^{\ell} - \frac{x_k + h \cdot f(x_{k+1}^{\ell}, t_{k+1}) - x_{k+1}^{\ell}}{h \cdot \partial f(x_{k+1}^{\ell}, t_{k+1})/\partial x - 1} \tag{2.24}$$

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:

$$\mathbf{x}^{\boldsymbol{\ell+1}} = \mathbf{x}^{\ell} - \left(\mathcal{H}^{\ell}\right)^{-1} \cdot \mathcal{F}^{\ell} \tag{2.25}$$

donde:

$$\mathcal{H} = \frac{\partial \mathcal{F}}{\partial \mathbf{x}} = \begin{bmatrix} \partial \mathcal{F}_1/\partial x_1 & \partial \mathcal{F}_1/\partial x_2 & \dots & \partial \mathcal{F}_1/\partial x_n \\ \partial \mathcal{F}_2/\partial x_1 & \partial \mathcal{F}_2/\partial x_2 & \dots & \partial \mathcal{F}_2/\partial x_n \\ \vdots & \vdots & \ddots & \vdots \\ \partial \mathcal{F}_n/\partial x_1 & \partial \mathcal{F}_n/\partial x_2 & \dots & \partial \mathcal{F}_n/\partial x_n \end{bmatrix} \tag{2.26}$$

es la *matriz Hessiana* del problema.

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

$$\mathbf{x}_{\mathbf{k+1}}^{\boldsymbol{\ell+1}} = \mathbf{x}_{\mathbf{k+1}}^{\ell} - [h \cdot \mathcal{J}_{\mathbf{k+1}}^{\ell} - \mathbf{I}^{(\mathbf{n})}]^{-1} \cdot [\mathbf{x_k} + h \cdot \mathbf{f}(\mathbf{x}_{\mathbf{k+1}}^{\ell}, t_{k+1}) - \mathbf{x}_{\mathbf{k+1}}^{\ell}] \tag{2.27}$$

donde:

$$\mathcal{J} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{bmatrix} \partial f_1/\partial x_1 & \partial f_1/\partial x_2 & \dots & \partial f_1/\partial x_n \\ \partial f_2/\partial x_1 & \partial f_2/\partial x_2 & \dots & \partial f_2/\partial x_n \\ \vdots & \vdots & \ddots & \vdots \\ \partial f_n/\partial x_1 & \partial f_n/\partial x_2 & \dots & \partial f_n/\partial x_n \end{bmatrix} \tag{2.28}$$

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 qué tan a menudo recalculan el Jacobiano (esto se denomina iteración de Newton modificada). Cuanto más no lineal sea el problema, más a menudo hay que recalcular el Jacobiano. Así mismo, al cambiar el paso de integración, hay que refactorizar el Hessiano.

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.

2.2. Métodos de integración Monopaso

Se denomina métodos de integración monopaso a aquellos que calculan x_{k+1} utilizando únicamente información sobre x_k.

2.2.1. Métodos Runge-Kutta

Un método de Runge–Kutta es un algoritmo que avanza la solución desde $x_k(t_k)$ hasta $x_{k+1}(t_k + h)$, usando una formula del tipo

$$\mathbf{x_{k+1}} \;=\; \mathbf{x_k} \;+ h \cdot (c_1 \cdot \mathbf{k_1} + \ldots + c_n \cdot \mathbf{k_n})$$

donde las llamadas etapas $k_1 \ldots k_n$ se calculan sucesivamente a apartir de las ecuaciones:

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

El número n de evaluaciones de función en el algoritmo se llama 'numero de etapas' y frecuentemente es considerado como una medida del costo computacional de la formula considerada.

Una forma muy común de representar a los algoritmos de RK es a través de la tabla de Butcher. Esta tabla toma en el caso general la siguiente 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

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 2.1 brinda un pantallazo histórico sobre el desarrollo de los métodos de RK.

2.2.2. Dominio de estabilidad de los algoritmos RK

Dado que los RK desarrollados hasta aquí son explícitos, sus dominios de estabilidad son similares al del algoritmo de FE, o sea, que el contorno de estabilidad se cierra sobre el semiplano izquierdo del plano $(\lambda \cdot h)$.

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

Tabla 2.1: Historia de los Algoritmos de Runge–Kutta.

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 y en principio los dominios de estabilidad serán algo distintos entre sí en este caso.

Los dominios de estabilidad de los métodos RK1 (Euler) a RK4 se muestran en la Fig.2.7.

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.

2.2.3. Sistemas Stiff

Un sistema lineal y estacionario se dice que es *stiff* cuando es estable y hay autovalores 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.

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

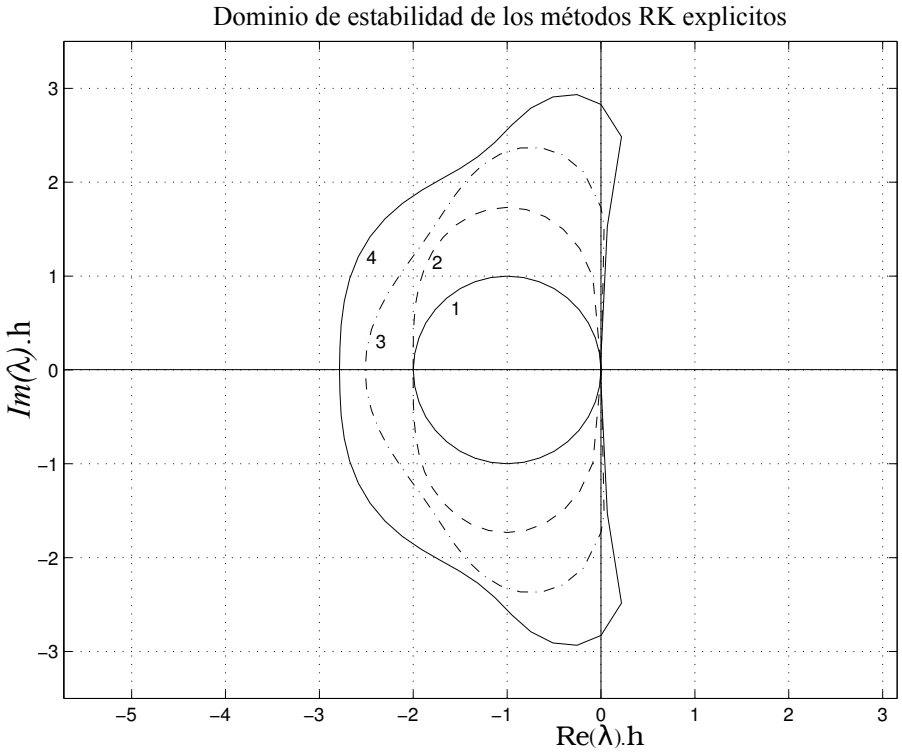


Figura 2.7: Dominios de estabilidad de los métodos explícitos RK.

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*.

2.2.4. Métodos de Interpolación hacia Atrás

Para poder tratar los problemas stiff se necesitan algoritmos A–estables. A continuación se mostrará una clase especial de algoritmos IRK que se

denominan *métodos de backinterpolation*, o métodos de interpolación hacia atrás (métodos BI). Los métodos BI pueden hacerse F–estables, L–estables o algo en el medio de ambas características de acuerdo a la necesidad del usuario. Solo se mostrarán a continuación algunos de los métodos F estables ya que son los que sirven para la simulación de sistemas stiff (sobre los que trata esta Tesis) y además su explicación es bastante mas fácil e intuitiva.

La idea de los Métodos BI consiste en dar un paso hacia atrás e iterar hasta que la condición 'final' del paso hacia atrás coincida con de x_k. Para poder comprender la idea observemos que el método de BE:

$$\mathbf{x_{k+1}} = \mathbf{x_k} + h \cdot \dot{\mathbf{x}}_{\mathbf{k+1}} \tag{2.29}$$

La Ec.(2.29) puede reescribirse como:

$$\mathbf{x_k} = \mathbf{x_{k+1}} - h \cdot \dot{\mathbf{x}}_{\mathbf{k+1}} \tag{2.30}$$

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 $\mathbf{x_k}$.

Aplicando esta idea se pueden obtener algoritmos BRK de distintos órdenes a partir de los algoritmos RK. En la figura Fig.(2.8) pueden verse los dominios de estabilidad de los mismos

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 $-h$.

2.2.5. Control de Paso

Como se vio anteriormente, 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

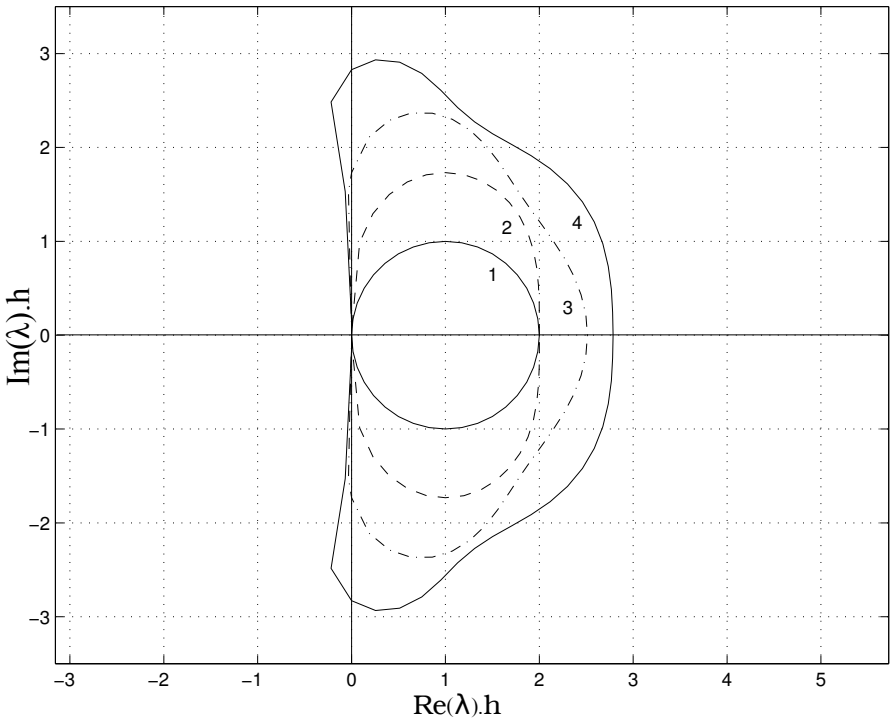


Figura 2.8: Dominios de estabilidad de los métodos básicos de interpolación hacia atrás.

constante. Esta observación nos lleva a la necesidad de buscar *métodos de paso variable* que a su vez requerirán de *algoritmos de control de paso*.

Básicamente la idea del control de paso consiste en tomar dos algoritmos distintos RK, y repetir dos veces el mismo paso, uno con cada algoritmo. Los resultados de ambos diferirán en ε. La diferencia entre ambas soluciones, ε, se utiliza como estima del error de integración local.

Luego se define el error relativo según

$$\varepsilon_{\text{rel}} = \frac{|x_1 - x_2|}{\text{máx}(|x_1|, |x_2|, \delta)} \tag{2.31}$$

donde δ es un factor pequeño, por ejemplo,$\delta = 10^{-10}$, introducido para evitar problemas cuando los estados están próximos a cero.

El control de paso consiste en tratar de mantener constante el valor de ε_{rel}. Para esto existen varias heurísticas, una de ellas consiste en rechazar un paso y volverlo a realizar cuando el error obtenido es mayor que la tolerancia de error prefijada. Otra posibilidad consiste en aceptar un paso aunque el

error sea mayor que la tolerancia de error y disminuir el paso de integración en el próximo paso.

Como se mencionó al comienzo de esta sección, para estimar el paso se precisan realizar cada paso con dos algoritmos RK distintos. Aparentemente el problema de proceder así es que es el muy alto el costo de evaluar cada algoritmo RK al menos una vez en cada paso (cada paso se calcula 2 veces al menos). Una idea que logra mejorar notablemente el costo computacional así introducido consiste en utilizar algoritmos RK empotrados (dos algoritmos RK que coinciden en sus primeras etapas). El más utilizado de estos métodos es el RK4/5 que posee la siguiente 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 este método puede verse que x_1 es un RK4 de cinco etapas y x_2 es un RK5 de 6 etapas. Sin embargo, ambos algoritmos comparten las primeras 5 etapas. Luego, el método completo con control de paso resulta ser un RK5 de 6 etapas y el único costo adicional del control de paso es el cálculo del corrector de RK4. En definitiva el control de paso es casi gratuito.

2.3. Métodos de Integración Multipaso

En la sección 2.2, se mostraron 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 explícita, 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 se comienza un nuevo paso, se deja de lado todas las evaluaciones anteriores de la función $\mathbf{f}$ en el paso anterior.

En esta sección se verá 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. En tal sentido, estos métodos utilizan como base polinomios de interpolación o de extrapolación de orden alto.

2.3.1. Fórmulas Explícitas de Adams–Bashforth

Un exemplo dentro de la familia de los métodos Adams–Bashforth es el famoso método de Adams–Bashforth de tercer orden (AB3):

$$\mathbf{x}(t_{k+1}) = \mathbf{x}(t_k) + \frac{h}{12}\left(23\mathbf{f}_k - 16\mathbf{f}_{k-1} + 5\mathbf{f}_{k-2}\right) \tag{2.32}$$

Todos los algoritmos Adams–Bashforth son explícitos y pueden representarse mediante un vector α y una matriz β:

$$\alpha = \begin{bmatrix} 1 & 2 & 12 & 24 & 720 & 1440 \end{bmatrix}^T \tag{2.33a}$$

$$\beta = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & -1 & 0 & 0 & 0 & 0 \\ 23 & -16 & 5 & 0 & 0 & 0 \\ 55 & -59 & 37 & -9 & 0 & 0 \\ 1901 & -2774 & 2616 & -1274 & 251 & 0 \\ 4277 & -7923 & 9982 & -7298 & 2877 & -475 \end{bmatrix} \tag{2.33b}$$

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.

En la figura 2.9 se muestran los dominios de estabilidad de los métodos ABi

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

Desafortunadamente, los dominios se achican al aumentar el orden. De manera que en este caso no se puede aumentar el orden para poder usar pasos más grandes como era el caso de los métodos RK.

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

2.3.2. Fórmulas Implícitas de Adams–Moulton

En el casos de esta familia de métodos se puede mostrar a modo de ejemplo el caso del algoritmo implícito de Adams–Moulton de tercer orden:

$$\mathbf{x}(t_{k+1}) = \mathbf{x}(t_k) + \frac{h}{12}\left(5\mathbf{f}_{k+1} + 8\mathbf{f}_k - \mathbf{f}_{k-1}\right) \tag{2.34}$$

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

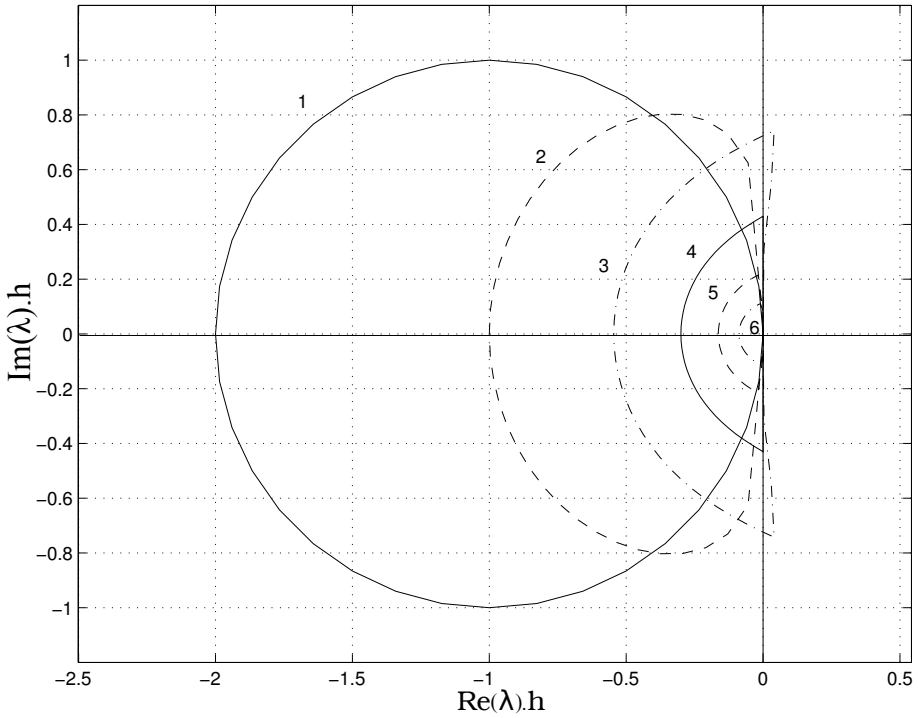


Figura 2.9: Dominios de estabilidad de los algoritmos explícitos AB.

$$\alpha = \begin{bmatrix} 1 & 2 & 12 & 24 & 720 & 1440 \end{bmatrix} \tag{2.35a}$$

$$\beta = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 5 & 8 & -1 & 0 & 0 & 0 \\ 9 & 19 & -5 & 1 & 0 & 0 \\ 251 & 646 & -264 & 106 & -19 & 0 \\ 475 & 1427 & -798 & 482 & -173 & 27 \end{bmatrix} \tag{2.35b}$$

Resulta claro que AM1 es el mismo método que BE.

En la figura 2.10 pueden verse los dominios de estabilidad de los algoritmos implícitos de Adams–Moulton. AM1 y AM2 son algoritmos útiles ... pero ya se los conocía con los nombres BE y regla trapezoidal respectivamente. 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.

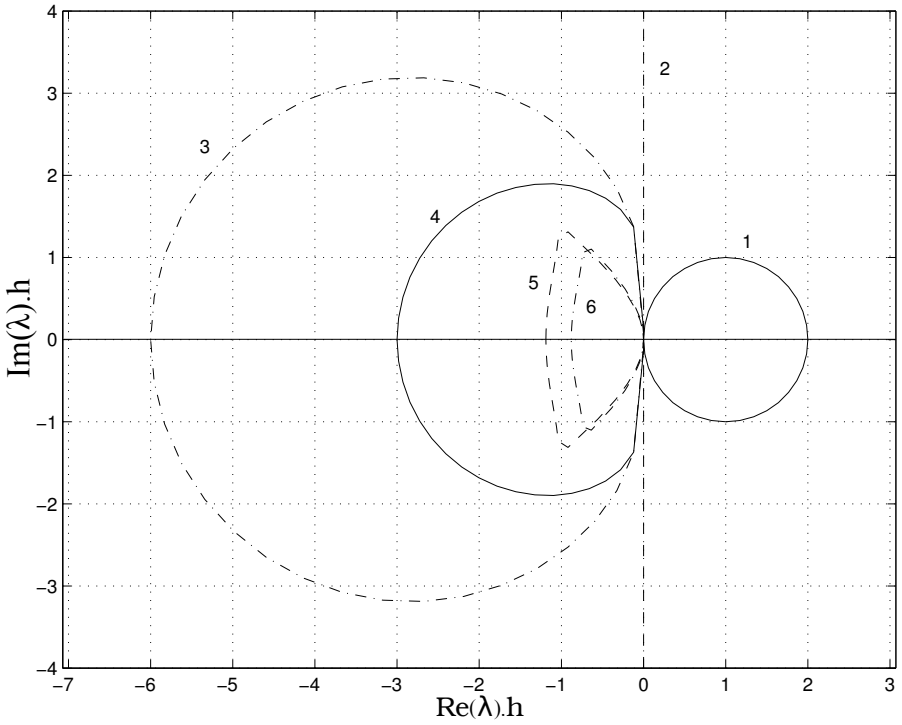


Figura 2.10: Dominios de estabilidad de los algoritmos implícitos de AM.

2.3.3. Fórmulas de Adams–Bashforth–Moulton

Los métodos de ABi y de AMi por sí solos no tienen ventajas debido a que sus regiones de estabilidad son muy pobres. Se puede entonces llegar a los algoritmos de Adams–Bashforth–Moulton construyendo 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.

En la figura 2.11 se muestran los dominios de estabilidad de los métodos ABM. En la misma puede verse que los dominios de estabilidad de ABMi son considerablemente mayores que los de ABi pero lamentablemente también se reduce al aumentar el orden.

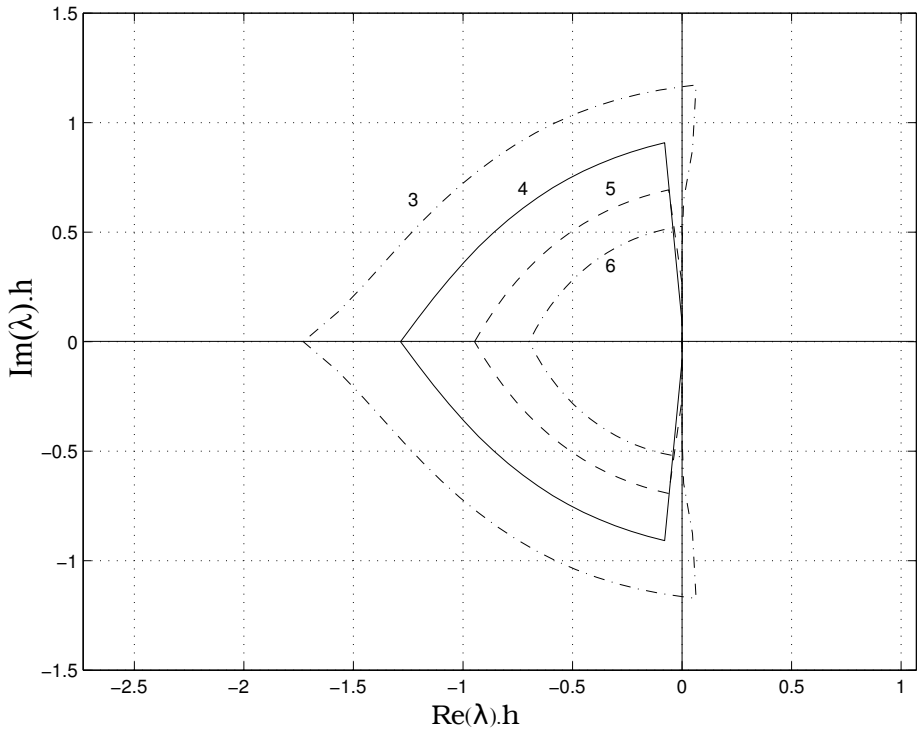


Figura 2.11: Dominios de estabilidad de los algoritmos predictor–corrector ABM.

2.3.4. Fórmulas de Diferencias Hacia Atrás (BDF)

Hasta ahora, todos los métodos multipaso presentados no sirven para simular sistemas stiff ya que los dominios de estabilidad de los mismos se cierran sobre el semiplano izquierdo. A continuación se presentará un método multipaso cuyo dominio de estabilidad se cierra a la derecha del semiplano $\lambda \cdot h$.

El algoritmo:

$$\mathbf{x}_{k+1} = \frac{18}{11}\mathbf{x}_k - \frac{9}{11}\mathbf{x}_{k-1} + \frac{2}{11}\mathbf{x}_{k-2} + \frac{6}{11} \cdot h \cdot \mathbf{f}_{k+1} \tag{2.36}$$

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

Se pueden obtener distintos algoritmos de BDFi. La familia de métodos BDF también puede ser representada mediante un vector α y una matriz β:

$$\alpha = \begin{bmatrix} 1 & 2/3 & 6/11 & 12/25 & 60/137 \end{bmatrix}^T \tag{2.37a}$$

$$\beta = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 4/3 & -1/3 & 0 & 0 & 0 \\ 18/11 & -9/11 & 2/11 & 0 & 0 \\ 48/25 & -36/25 & 16/25 & -3/25 & 0 \\ 300/137 & -300/137 & 200/137 & -75/137 & 12/137 \end{bmatrix} \tag{2.37b}$$

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.2.12.

Finalmente, los métodos BDF son 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 solamente una banda muy angosta de área estable a la izquierda del origen y solamente 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.

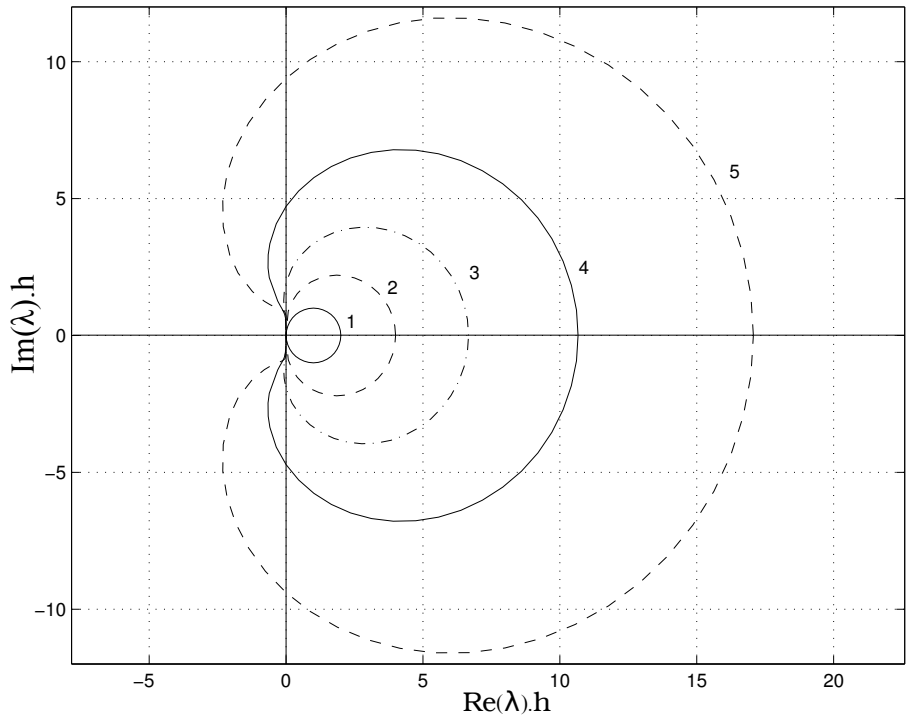


Figura 2.12: Dominios de estabilidad de los métodos implícitos BDF.

Al igual que todos los métodos implícitos, cuando el sistema es no lineal en los métodos BDF se debe utilizar el método de iteración de Newton ya que en general no es posible despejar x_{k+1} al operar como se describió a partir de la ec.(2.37).

2.3.5. Control de Paso

El control de paso en estos métodos no es ni sencillo ni económico, ya que las fórmulas multipaso se basan en considerar que los puntos en el tiempo están equiespaciados. Luego, si se cambia el paso de integración durante la simulación, los puntos en el tiempo no están más equiespaciados y se deben utilizar polinomios de interpolación o extrapolación para poder calcular valores equiespaciados temporalmente y poder así continuar usando los métodos multipaso. El cálculo mediante estos polinomios se debe rea-

lizar cada vez que se cambia el paso. Debido al costo que esto implica, en estos métodos se trata de variar lo menos posible el paso.

2.3.6. 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).

Para los métodos de tipo ABM y AB la solución que se usa generalmente es comenzar dando los primeros pasos con algoritmos de Runge–Kutta del mismo orden de método multipaso a utilizar para no comenzar con problemas de precisión.

El caso de BDF es un poco más problemático ya que al tratar con sistemas stiff, RK deberá utilizar pasos excesivamente pequeños. En este caso lo que se hace es utilizar BRK en el arranque para no tener limitaciones de estabilidad en el arranque.

2.4. Métodos Linealmente Implícitos

Los métodos linealmente implícitos o semi–implícitos explotan el hecho de que los métodos implícitos aplicados a sistemas lineales pueden implementarse directamente mediante inversión matricial.

Uno de los métodos de este tipo más utilizados es la fórmula de Euler semi–implícita:

$$\mathbf{x_{k+1}} = \mathbf{x_k} + h \cdot [\mathbf{f}(\mathbf{x_k}, t_k) + \mathcal{J}_{\mathbf{x_k},t_k} \cdot (\mathbf{x_{k+1}} - \mathbf{x_k})] \tag{2.38}$$

donde

$$\mathcal{J}_{\mathbf{x_k},t_k} = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\mathbf{x_k},t_k} \tag{2.39}$$

es la matriz Jacobiana evaluada en $(\mathbf{x_k}, t_k)$.

Notar que

$$\mathcal{J}_{\mathbf{x_k},t_k} \cdot (\mathbf{x_{k+1}} - \mathbf{x_k}) \approx \mathbf{f}(\mathbf{x_{k+1}}, t_{k+1}) - \mathbf{f}(\mathbf{x_k}, t_k) \tag{2.40}$$

y entonces:

$$\mathbf{f}(\mathbf{x_k}, t_k) + \mathcal{J}_{\mathbf{x_k},t_k} \cdot (\mathbf{x_{k+1}} - \mathbf{x_k}) \approx \mathbf{f}(\mathbf{x_{k+1}}, t_{k+1}) \tag{2.41}$$

Es decir, el método de Euler linealmente implícito se aproxima al método de Backward Euler. Más aún, en el caso lineal:

$$\dot{\mathbf{x}} = \mathbf{A} \cdot \mathbf{x} \tag{2.42}$$

tenemos:

$$\mathbf{x_{k+1}} = \mathbf{x_k} + h \cdot [\mathbf{A} \cdot \mathbf{x_k} + \mathcal{J}_{\mathbf{x_k},t_k} \cdot (\mathbf{x_{k+1}} - \mathbf{x_k})] = \mathbf{x_k} + h \cdot \mathbf{A} \cdot \mathbf{x_{k+1}} \tag{2.43}$$

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 (2.38) puede reescribirse como:

$$(I - h \cdot \mathcal{J}_{\mathbf{x_k},t_k}) \cdot \mathbf{x_{k+1}} = (I - h \cdot \mathcal{J}_{\mathbf{x_k},t_k}) \cdot \mathbf{x_k} + h \cdot \mathbf{f}(\mathbf{x_k}, t_k) \tag{2.44}$$

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:

$$\mathbf{x_{k+1}} = \mathbf{x_k} + h \cdot (I - h \cdot \mathcal{J}_{\mathbf{x_k},t_k})^{-1} \cdot \mathbf{f}(\mathbf{x_k}, t_k) \tag{2.45}$$

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.

Teniendo en cuenta que el dominio de estabilidad coincide con el de BE, este método resulta apropiado para simular sistemas stiff. Los métodos linealmente implícitos de bajo orden son a menudo la mejor opción para la simulación en tiempo real. Sin embargo, cuando la dimensión del problema es grande, el costo de la inversión matricial puede resultar demasiado alto.

2.5. Integración Multirate

La rigidez en sistemas grandes está frecuentemente relacionada con la presencia de algunos subsistemas lentos y otros rápidos que se pueden identificar. 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. 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 llevan a los conceptos de integración multipaso e integración de modo mixto.

Si, por ejemplo se tiene un sistema en el cual coexisten dos dinámicas (una rápida y otra lenta), y se pueden identificar claramente los submodelos rápidos y lentos. Entonces, se puede dividir al sistema en dos apartes y utilizar dos pasos de integración distintos en cada parte.

De esta forma se integra el subsistema rápido con un paso pequeño, y se integra el subsistema lento con un paso mayor.

Esta idea puede ser expresada para sistemas de la forma:

$$\dot{\mathbf{x}}_\mathbf{f}(t) = \mathbf{f}_\mathbf{f}(\mathbf{x}_\mathbf{f}, \mathbf{x}_\mathbf{s}, t) \tag{2.46a}$$

$$\dot{\mathbf{x}}_\mathbf{s}(t) = \mathbf{f}_\mathbf{s}(\mathbf{x}_\mathbf{f}, \mathbf{x}_\mathbf{s}, t) \tag{2.46b}$$

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 lleva a unas ecuaciones en diferencias de la forma:

$$\begin{aligned} \mathbf{x}_\mathbf{f}(t_i + (j+1)\cdot h) &= \mathbf{x}_\mathbf{f}(t_i + j\cdot h) + h \cdot \mathbf{f}_\mathbf{f}(\mathbf{x}_\mathbf{f}(t_i + j\cdot h), \\ &\quad \mathbf{x}_\mathbf{s}(t_i + j\cdot h), t_i + j\cdot h) \qquad (2.47a) \\ \mathbf{x}_\mathbf{s}(t_i + k\cdot h) &= \mathbf{x}_\mathbf{s}(t_i) + h \cdot \mathbf{f}_\mathbf{s}(\mathbf{x}_\mathbf{f}(t_i), \mathbf{x}_\mathbf{s}(t_i), t_i) \qquad (2.47b) \end{aligned}$$

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 (2.47a–b) no especifican cómo calcular $\mathbf{x}_\mathbf{s}(t_i + j\cdot h)$, ya que las variables del subsistema lento no se evalúan en los instantes intermedios.

Una opción es poner $\mathbf{x}_\mathbf{s}(t_i + j\cdot h) = \mathbf{x}_\mathbf{s}(t_i)$, es decir, utilizar el último valor calculado.

2.6. Simulación de Sistemas Discontinuos

Como se vio en las secciones anteriores de este capítulo, todos los métodos de integración de tiempo discreto 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.

El siguiente ejemplo consiste en una pelota rebotando contra el piso.

$$\dot{x}(t) = v(t) \tag{2.48a}$$

$$\dot{v}(t) = -g - s_w(t) \cdot \frac{1}{m}(k \cdot x(t) + b \cdot v(t)) \tag{2.48b}$$

donde

$$s_w = \begin{cases} 0 & \text{si } x(t) > 0 \\ 1 & \text{en otro caso} \end{cases} \tag{2.49}$$

En este modelo, se considera 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.

Los parámetros usados en el mismo son: $m = 1$, $b = 30$, $k = 1 \times 10^6$ y $g = 9{,}81$.

Simulando este modelo varias veces utilizando el método RK4, durante 5 segundos a partir de la condicion inicial $x(0) = 1$, $v(0) = 0$. Se comenzaron a tener resultados *decentes* con un paso de integración $h = 0{,}002$. En la Fig 2.13 se muestran los resultados de la simulación con pasos $h = 0{,}002$, $h = 0{,}001$ y $h = 0{,}0005$.

Mirando el comienzo de la simulación, no hay error apreciable hasta el primer pique y a partir del mismo las soluciones difieren notablemente entre sí. Además, en la simulación con paso $h = 0{,}002$ luego del séptimo pique la pelota gana más altura de la que tenia anteriormente lo cual es evidentemente erroneo. El problema tiene que ver con la discontinuidad haciendo que no se pueda confiar en los resultados de simulación usando paso fijo.

En la figura 2.14 se muestran los resultados al simular el sistema utilizando un método de paso variable, en este caso un método RK23 (utiliza un RK de segundo orden y para controlar el paso utiliza el mismo método implementado con dos semipasos de $h/2$). El método es de segundo orden, y el término del error es de tercer orden. Para la simulación se utilizaron las tolerancias relativas 10^{-3}, 10^{-4} y 10^{-5}.

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 realizadas es que en algunos pasos los métodos integraban a través de una discontinuidad. Es decir, calculaban

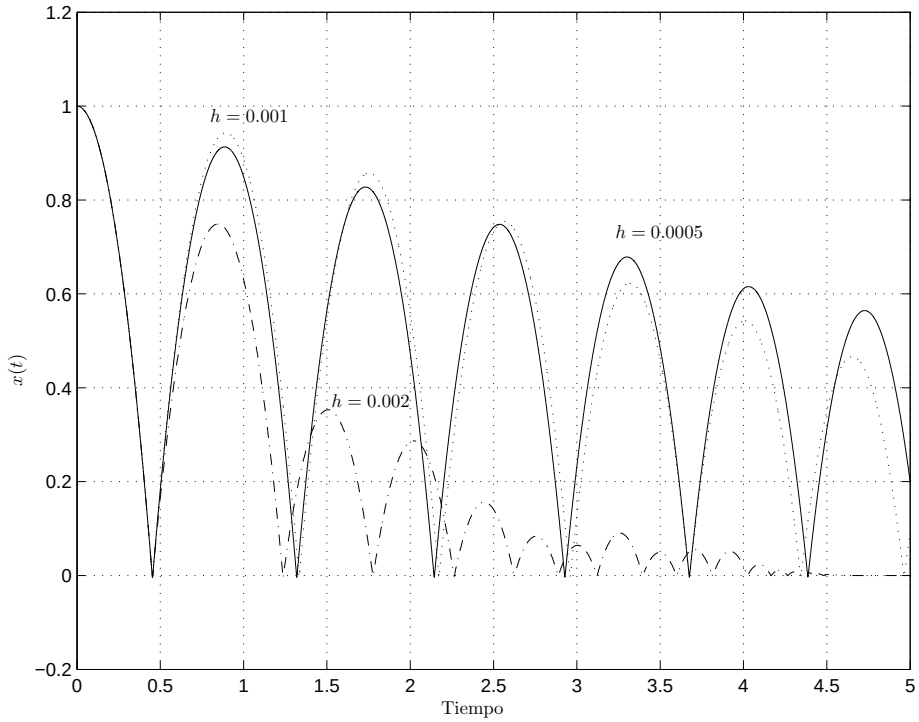


Figura 2.13: Simulación con RK4 de la pelotita rebotando.

como si tuvieran 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 se necesita es un método de paso variable que dé un paso exactamente en el instante t^* en el que ocurre la discontinuidad. De esa forma, siempre se estará integrando una función 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*.

2.6.1. Eventos Temporales

Se denomina *Eventos temporales* a las discontinuidades de las cuales se sabe con cierta anticipación el tiempo de ocurrencia de las mismas. La forma de tratar eventos temporales es muy sencilla. Dado que se conoce cuando ocurrirán, símplemente se le debe avisar 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

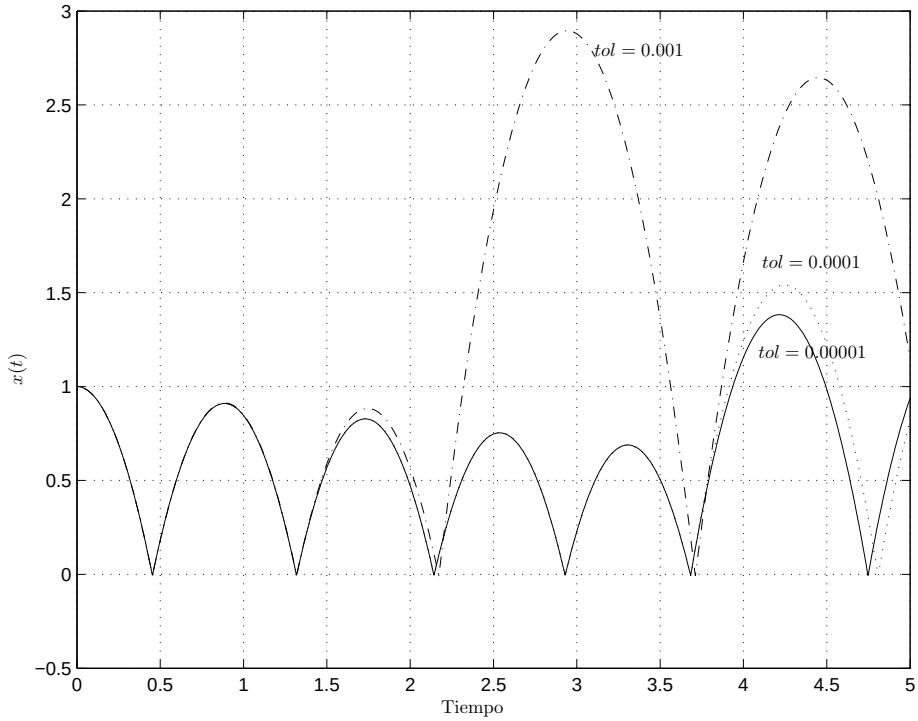


Figura 2.14: Simulación con RK23 de la pelotita rebotando.

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.

2.6.2. 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 se sabe es que los piques se dan cuando la altura es cero. Es decir, se conoce 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 $\mathcal{F}_0(x, v) = x$.

Se dice 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.

2.7. Métodos de tiempo discreto y sistemas stiff

Hasta este punto se ha hecho un repaso con cierto grado de rigurosidad de la mayoría de los métodos de integración de tiempo discreto sin dejar de lado aquellos que no son apropiados para la simulación de sistemas stiff. El objetivo de esta sección es recalcar las definiciones de sistema stiff y remarcar cuales de los métodos mencionados en el transcurso de este capítulo son adecuados para sistemas stiff y cuales no, dando además, una breve descripción del motivo.

Existen muchas definiciones de sistema stiff, algunas de ellas son:

- Un sistema LTI se dice que es stiff si todos sus autovalores tienen parte real negativa y la relación entre las mismas es muy grande.

- Se esta en presencia de un sistema stiff cuando algunas componentes de la solución varían mucho más rápido que las otras.

- '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'.

Las dos primeras definiciones si bien son las más difundidas no definen claramente lo que es un sistema stiff. La primera definición solo se aplica a sistemas LTI dejando de lados los sistemas no lineales y además, no deja claro cuan grande debe ser la relación entre las partes reales de los autovalores. En el caso de la segunda definición, si bien es de carácter más general ya que incluye también a los sistemas no lineales, no es una definición que permita diferenciar claramente un sistema stiff de uno no stiff ya que nuevamente no aclara cuanto más rápido debe ser una componente respecto a la otra.

Por lo tanto la tercera definición es la que más precisamente define el significado de sistema stiff.

Como se vio en el transcurso de este capítulo, para integrar entonces sistemas stiff en forma eficiente, se deben utilizar métodos que incluyan en su región estable el semiplano izquierdo completo del plano $(\lambda \cdot h)$, o al menos una gran porción del mismo. Teniendo en cuenta esto y las regiones de estabilidad de los métodos desarrollados se puede concluir que:

- Dentro de los denominados método de integración monopaso

  - dado que los métodos RK tienen dominios de estabilidad que se cierran sobre el lado izquierdo del plano $(\lambda \cdot h)$, los mismos son muy ineficientes para la simulación de este tipo de sistemas.
  - Los método de interpolación hacia atrás BRK tienen dominios de estabilidad que se cierran sobre el lado derecho del plano $(\lambda \cdot h)$ lo cual hace que sean adecuados para la simulación de este tipo de sistemas. Sin embargo, en estos métodos es necesario resolver un sistema no lineal de ecuaciones (algebraicas) en cada paso de la integración, lo que hace estas fórmulas poco competitivas especialmente cuando la dimensión del problema es grande. De hecho, al integrar numéricamente un sistema de ecuaciones diferenciales

de dimensión N con un método Runge-Kutta implícito de m etapas, es necesario en general resolver en cada paso un sistema no lineal de ecuaciones de dimensión $m \cdot N$. Además, como todos los métodos de tiempo discreto, cuando el sistema es además discontinuo, estos métodos tienen un costo adicional debido a los algoritmos implementados para la detección de discontinuidades.

- En el caso de los métodos de integración multipaso, los únicos que pueden utilizarse para simular sistemas stiff son las denominadas formulas de diferencia hacia atrás (BDF). Estos métodos poseen dominios de estabilidad que se cierran sobre el lado derecho del plano $(\lambda \cdot h)$. En forma similar a los métodos BRK, hay que resolver en cada paso un sistema de ecuaciones no lineal pero en este caso sólo del orden del sistema. El mayor inconveniente que tienen estos métodos respecto a los BRK es que precisan de otros métodos para su inicialización. Luego, cuando el sistema es discontinuo, además del costo del algoritmo de detección de discontinuidades, los mismos deben usar otros métodos para reinicializarse después de cada discontinuidad.

- Los métodos Linealmente implícitos tienen la ventaja de poseer regiones de estabilidad que se cierran en el lado derecho del plano $(\lambda \cdot h)$ haciéndolos aptos para la simulación de sistemas stiff. Para implementar estas fórmulas, basta resolver un sistema lineal de ecuaciones algebraicas en cada paso, con las ventajas que esto supone sobre los métodos considerados con anterioridad. Esto los convierte en métodos adecuados para la simulación de sistemas stiff en tiempo real donde resulta muy importante que el tiempo de cada paso sea predecible y bajo. Al igual que los métodos antes mencionados, presentan las mismas dificultades que los métodos BRK cuando los sistemas son además discontinuos.

- Finalmente, los métodos multirate son una muy buena opción cuando el sistema es de gran dimensión y las componentes del mismo que generan la dinámica rápida son fácilmente diferenciables del resto. La contrapartida de estos métodos es que además del trabajo de modelado, requieren que la persona que desea simular el sistema identifique las partes del mismo y de alguna manera informe sobre las mismas al programa que implementa la simulación.

Capítulo 3

Métodos de integración por cuantificación (QSS)

Como se vio en el capítulo anterior, para poder simular sistemas continuos los métodos de integración tradicionales, realizan una discretización del tiempo de manera de transformar el modelo de tiempo continuo en un modelo de ecuaciones en diferencias "equivalente". De este modo, la solución del modelo de tiempo discreto en los instantes de muestreo se aproxima a la solución del modelo original en dichos instantes.

Si bien los métodos de integración tradicionales son los mas difundidos y han dado respuesta a muchos problemas de simulación, existen también muchos modelos para los cuales la solución brindada por los mismos no ha sido del todo satisfactoria desde el punto de vista de eficiencia. Además, existen modelos de sistemas que, según cómo se formulen, pueden llegar a ser imposibles de simular a "ciegas", entendiéndose por esto, cargando simplemente el modelo matemático en un simulador sin tener mucha noción del comportamiento del modelo.

A fines de los 90's se comenzó a desarrollar una metodología alternativa a la clásica discretización temporal para poder aproximar los sistemas continuos. En la misma, en lugar de discretizar el tiempo, se cuantifican las variables de estado manteniendo continua la variable tiempo. De este modo, se verá que en lugar de obtenerse un modelo de tiempo discreto equivalente, se llega a un modelo de eventos discreto equivalente. Y que esta discretización puede representarse fácilmente mediante el formalismo DEVS.

En este capítulo se presentarán los principios de esta metodología y los métodos de integración basados en la cuantificación de los estados sobre los cuales se basan los trabajos desarrollados en esta tesis.

Glossario

Los distintos algoritmos descriptos a lo largo de esta tesis usan la siguiente notación:

---

t = tiempo actual de simulación
t_f = tiempo final de simulación
x = vector de estados(x_i se refiere a la i−ésima componente)
$\dot{x}$ = vector de derivadas primeras respecto del tiempo
$\ddot{x}$ = vector de derivadas segundas respecto del tiempo
q = vector de estados cuantificados
$\dot{q}$ = vector de pendientes de estados cuantificados
t^x = arreglo que contiene el tiempo del último cambio para cada estado
t^η = arreglo que contiene el tiempo de la próxima actualización para cada estado cuantificado
ΔQ = arreglo que contiene el cuantum de cada estado
e = tiempo transcurrido (variable auxiliar)
A = matriz Jacobiana
u = matriz de coeficientes afines
$\dot{u}$ = matriz de pendientes de los coeficientes afines
$\dot{x}_i^+$ = estimación del valor futuro de $\dot{x}_i$
q_i^- = valor anterior de el estado cuantificado q_i
$\dot{x}_i^-$ = valor anterior de la derivada del estado $\dot{x}_i$
h = paso de integración (variable auxiliar)
t^q = rreglo que contiene el tiempo del último cambio para cada estado cuantificado
$\ddot{x}_i^+$ = estimación del valor futuro de $\ddot{x}_i$

3.1. Ejemplo Introductorio de discretización espacial

Un oscilador armónico puede ser representado mediante el modelo de un sistema continuo de segundo orden:

$$\begin{aligned} \dot{x}_{a_1}(t) &= x_{a_2}(t) \\ \dot{x}_{a_2}(t) &= -x_{a_1}(t). \end{aligned} \tag{3.1}$$

Si se saben las condiciones iniciales $x_{a_1}(t_0)$ y $x_{a_2}(t_0)$, dado que el sistema es lineal, es muy fácil encontrar la solución analítica del modelo, siendo ésta $x_{a_i}(t) = c_i \sin(t) + d_i \cos(t)$ con c_i y d_i constantes.

Veamos ahora que ocurre si se modifica el sistema (3.1) de la siguiente manera:

$$\begin{aligned} \dot{x}_1(t) &= \text{floor}[x_2(t)] \triangleq q_2(t) \\ \dot{x}_2(t) &= -\text{floor}[x_1(t)] \triangleq -q_1(t) \end{aligned} \tag{3.2}$$

donde floor(x_i) es una función que da como resultado el valor entero más cercano a x_i y que es a su vez menor que x_i.

A pesar de que este nuevo sistema es no lineal y discontinuo, se lo puede simular fácilmente. Tomemos como condiciones iniciales $x_1(0) = 4{,}5$ y $x_2(0) = 0{,}5$.

Con estos valores iniciales se tiene que $q_1(0) = 4$ y $q_2(0) = 0$ y se se mantienen en esos valores hasta que $x_1(t)$ o $x_2(t)$ cruce a través de alguno de los valores enteros más próximos a ella. En consecuencia, $\dot{x}_1(0) = 0$ y $\dot{x}_2(0) = -4$ por lo que x_1 se mantiene constante y x_2 decrece con pendiente -4.

Luego de $0{,}5/4 = 0{,}125$ segundos (instante $t_1 = 0{,}125$), x_2 cruza por 0 y q_2 toma el valor -1. Entonces, cambia la pendiente de x_1 tomando el valor $\dot{x}_1(t_1^+) = -1$

Luego, x_2 cruza por -1 en el instante $t_2 = t_1 + 1/4$, y q_2 toma el valor -2. En ese instante, $x_1(t_2) = 4{,}5 - 1/4 = 4{,}25$ y $\dot{x}_1(t_2^+) = -2$.

El próximo cambio ocurre cuando x_1 cruza por 4 en el instante de tiempo $t_3 = t_2 + 0{,}25/2$. Luego, $q_1(t_3^+) = 3$ y la pendiente de x_2 pasa a ser -3. Continuando con el análisis de la misma manera, se obtienen los resultados de la *simulación* mostrados en las figuras 3.1–3.2. Los resultados son muy similares a la solución analítica del sistema original3.1.

Aparentemente, cuando se reemplaza x_i por $q_i = \text{floor}(x_i)$ en un sistema como el de la Ec.(3.1), se obtiene una aproximación del problema original pero que puede ser resuelta en un número finito de pasos conservando un comportamiento similar al del sistema original.

Mas aún, se puede asociar la solución del sistema (3.2) con el comportamiento de un sistema de eventos discretos (ya que realizan un número finito de cambios), pero no al de un sistema de tiempo discreto. En este caso, los eventos corresponden a los cruces de las variables de estado por los niveles de discretización de las mismas, cuando ocurren pueden provocar cambios en las derivadas de los estados, que conducen a una reprogramación del tiempo en que ocurre el próximo cruce de nivel.

Para poder ver como se puede generalizar esta idea, se deberá introducir antes algunas herramientas que permiten representar y simular sistemas como el de la Ec.(3.2).

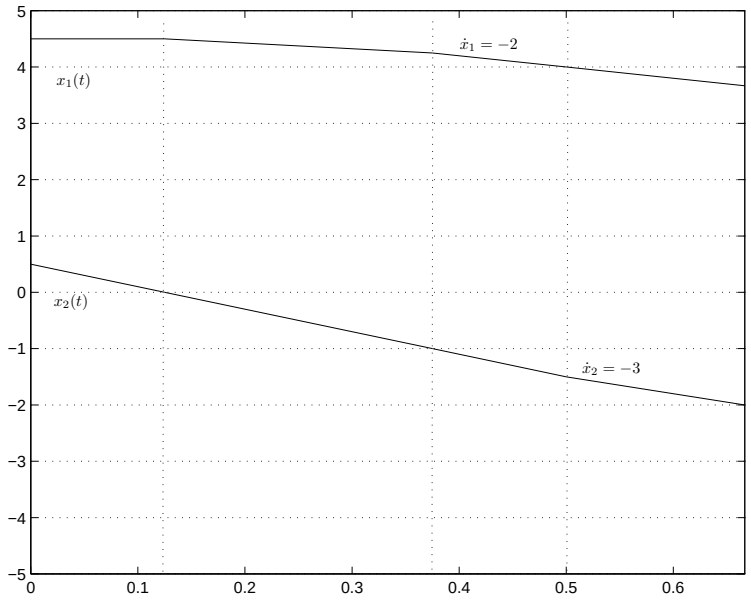


Figura 3.1: Simulación de las trayectorias del sistema de la Ec.(3.2) (Comienzo).

3.2. Método de los Sistemas de Estados Cuantificados (QSS)

En el sistema dado por (3.2), 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*. En el caso mostrado, el bloque con forma de escalera es un caso particular de cuantificador con cuantificación uniforme.

Esta idea constituye la primer aproximación a un método para simular sistemas continuos mediante eventos discretos. Desafortunadamente, esta idea no funciona.

Este método conduce, en la mayor parte de los casos, a 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. Por ejemplo, dada la siguiente ecuación diferencial de primer orden:

$$\dot{x}(t) = -x(t) - 0{,}5 \tag{3.3}$$

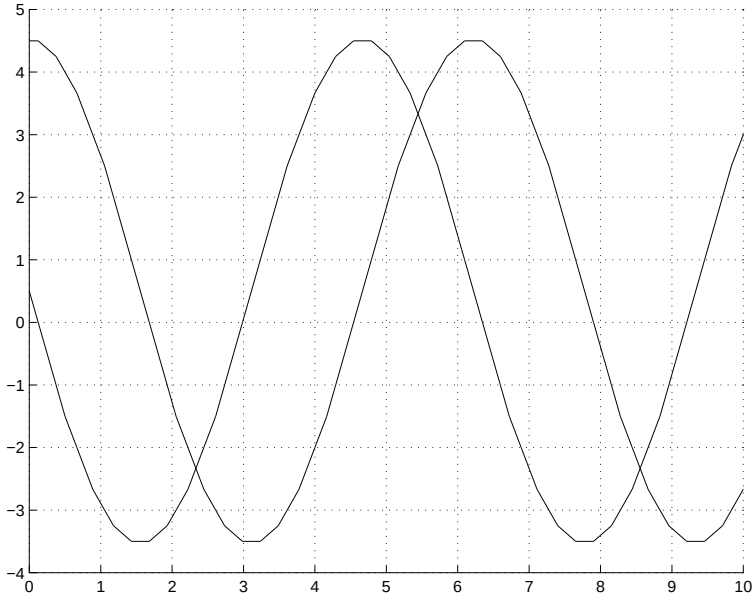


Figura 3.2: Simulación de las trayectorias del sistema de la Ec.(3.2) (Comienzo).

con condiciones iniciales $x(0) = 0$. Si se procede, como se describió anteriormente, reemplazando $x(t)$ por $q(t) = \text{floor}[x(t)]$ en el lado derecho de (3.3) se obtiene

$$\dot{x}(t) = -q(t) - 0{,}5 \tag{3.4}$$

Si se simula este sistema, se obtiene el siguiente resultado. En el instante $t = 0$ se tiene $x(0) = 0$ y por lo tanto $q(0) = 0$. Entonces, la derivada $\dot{x}(0) = -0{,}5$ y $x(0^+)$ es negativa. Luego, $q(0^+) = -1$ y $\dot{x}(0^+) = +0{,}5$.

Como ahora la derivada es positiva, $x(t)$ vuelve inmediatamente a cero y $q(t)$ también. Ahora, estamos nuevamente en las condiciones en que se comenzó a simular el sistema y el tiempo de simulación no avanzó.

Este comportamiento da como resultado una oscilación permanente en la cual $q(t)$ cambia entre 0 y -1. El problema reside en que estas oscilaciones tienen periodo 0 (frecuencia infinita).

A pesar de que la metodología presentada no funciona, sirvió de base para el desarrollo de la familia de métodos de integración que se muestra en las siguientes secciones de este capítulo. Estos últimos, fueron la referencia tomada para el desarrollo de los métodos de integración presentados en esta

tesis.

Si se analizan las oscilaciones infinitamente rápidas en el sistema de la ecuación (3.3) se puede 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 instantáneamente sino que tiene una frecuencia máxima acotada.

En base al cambio de la función de cuantificación sin histéresis por una con histéresis surgió toda una familia de métodos de integración por cuantificación denominados QSS (Quantaized State System). Dentro de esta familia existen métodos de primer, segundo y tercer orden denominados QSS1, QSS2 y QSS3 respectivamente.

3.2.1. Método de integración de estados cuantificados QSS1

Para poder mostrar formalmente el métodos QSS es conveniente definir primero el concepto de *función de cuantificación con histéresis*.

Sea que $x(t) : \Re \to \Re$ es una trayectoria continua en el tiempo, y $q(t) : \Re \to \Re$ una trayectoria seccionalmente constante. Luego, $x(t)$ y $q(t)$ se relacionan mediante una función de cuantificación con histéresis uniforme si se cumple que:

$$q(t) = \begin{cases} \text{floor}[x(t_0)/\Delta Q] \cdot \Delta Q & \text{si } t = t_0 \\ x(t) & \text{si } |q(t^-) - x(t)| \geq \Delta Q \\ q(t^-) & \text{en otro caso} \end{cases} \tag{3.5}$$

El parámetro de la cuantificación ΔQ se denomina *quántum*.

Las Figuras 3.3–3.4 muestran una función de cuantificación con histéresis de orden cero y dos variables relacionadas mediante dicha función

Notar que un cuantificador con histéresis tiene memoria. Es decir, calcula $q(t)$ no solo en función del valor actual de $x(t)$ sino también de su valor pasado.

Una vez definida la función de cuantificación con histéresis se puede definir el método QSS1.

> Dado un sistema invariante en el tiempo en su forma de sistema de ecuaciones diferenciales ordinarias (ODEs):
>
> $$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t), t) \tag{3.6}$$

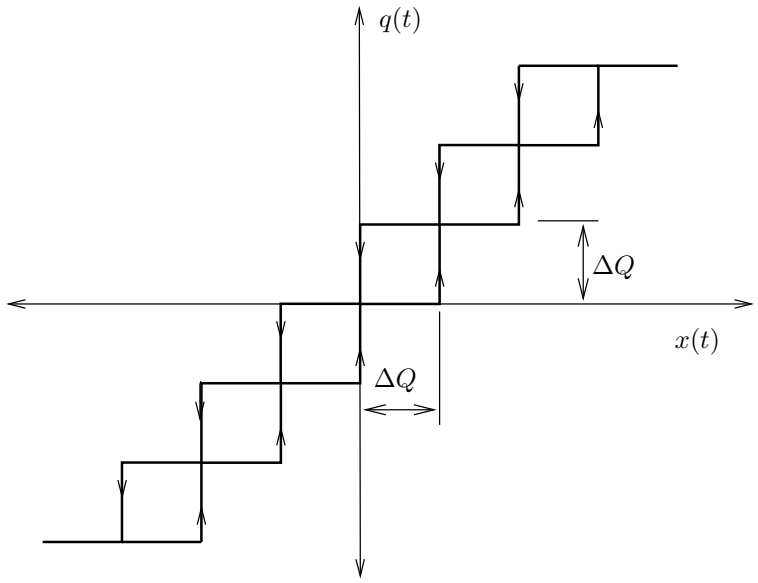


Figura 3.3: Función de Cuantificación con Histéresis de orden cero.

donde $\mathbf{x}(t) \in \mathbb{R}^n$ es el vector de estados, el método QSS de orden 1 (QSS1) resulve de manera analítica el siguiente sistema de ODEs aproximado, llamado *sistema de estados cuantificados*(en inglés, *Quantized State System*):

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{q}(t), t) \tag{3.7}$$

Aquí, $\mathbf{q}(t)$ es el vector de *estado cuantificados*, cuyos componentes están sujetos a seguir trayectorias seccionalmente contantes. Cada estoado cuantificado $q_i(t)$ está relacionado con el correspondiente estado $x_i(t)$ a través de una *función de cuantificación con histéresis de orden cero*:

$$q_i(t) = \begin{cases} x_i(t) & \text{if } |q_i(t^-) - x_i(t)| = \Delta Q_i \\ q_i(t^-) & \text{en otro caso} \end{cases}$$

Es decir, para utilizar el método de QSS1 simplemente se debe elegir el *quántum* a utilizar ΔQ_i para cada variable de estado.

De aquí podemos ver que $q_i(t)$ sólo cambia cuando difiere de $x_i(t)$ en una magnitud de ΔQ_i llamada *quantum*. Cuando la diferencia entre entre el estado y el estado cuantificado alcanza dicho quantum, el estado cuantificado

$q_i(t)$ toma el valor del estado $x_i(t)$. En la Figura 3.4 puede verse la trayectoria de un estado, así como la de su correspondiente estado cuantificado.

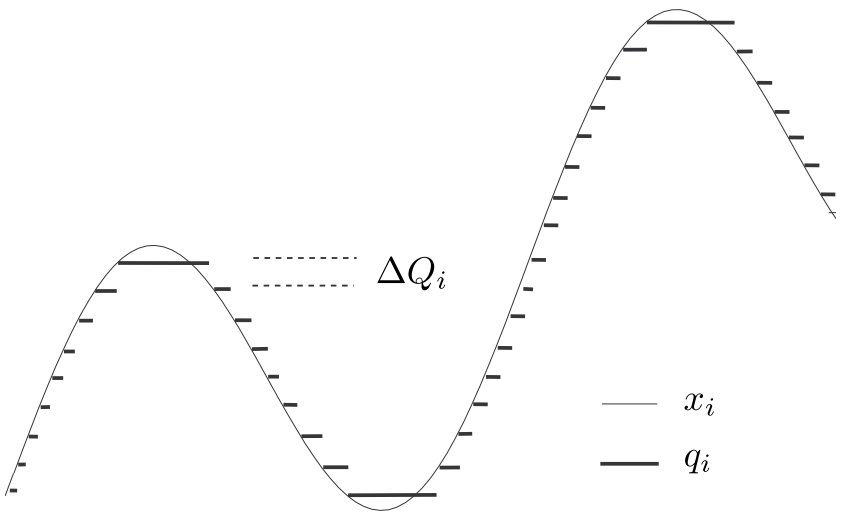


Figura 3.4: Variables Relacionadas con una Función de Cuantificación con Histéresis de orden cero.

Dado que las trayectorias de los estados cuantificados $q_i(t)$ son seccionalmente constantes, las derivadas temporales de los estados $\dot{x}_i(t)$ también siguen trayectorias seccionalmente constantes, y como consecuencia de esto, los estados $x_i(t)$ siguen trayectorias seccionalmente lineales.

Debido a esta particular forma de las trayectorias, la solución numérica de la Ec. (3.7) es muy sencilla y puede ser fácilmente traducida en un simple algoritmo de simulación.

Para $j = 1, \ldots, n$, sea t_j^η el próximo tiempo en el cual $|q_j(t) - x_j(t)| = \Delta Q_j$. Entonces, el algoritmo de simulación QSS funciona del siguiente modo[1]:

[1]El Algoritmo 3.1 describe el comportamiento de QSS1 durante la actualización de estados cuantificados. Los pasos que tienen lugar como consecuencia de las discontinuidades, o debido a cambios significativos en las trayectorias de entrada, no son descriptos aquí dado que son idénticos para los distintos algoritmos de la familia QSS.

Algoritmo 3.1: QSS1.

```
1  Mientras(t < t_f) // simular hasta tiempo final tf
2    t = min(t_j^η) // avanzar tiempo de simulación
3    i = argmin(t_j^η) // el i−ésimo estado cuantificado cambia
         primero
4    e = t − t_i^x // tiempo transcurrido desde la ultima
         actualización de xi
5    x_i = x_i + ẋ_i · e // actualizar el i−ésimo estado
6    q_i = x_i // actualizar el i−ésimo estado cuantificado
7    t_i^η = min(τ > t) tal_que |q_i − x_i(τ)| = ΔQ_i // calcular el
         tiempo del próximo cambio en el i−ésimo estado
         cuantificado
8    Para j ∈ [1, n] tal_que ẋ_j depende_de q_i
9      e = t − t_j^x // tiempo transcurrido desde la última
           actualización de xj
10     x_j = x_j + ẋ_j · e // actualizar el j−ésimo estado
11     Si j ≠ i entonces t_j^x = t // última actualización de xj
12     ẋ_j = f_j(q, t) // recalcular la derivada del j−ésimo
           estado
13     t_j^η = min(τ > t) tal_que |q_j − x_j(τ)| = ΔQ_j // recalcular el
           tiempo del próximo cambio en el j−ésimo estado
           cuantificado
14   fin_para
15   t_i^x = t // última actualización de xi
16 fin_mientras
```

Además, esa forma de las trayectorias implica que los instantes de tiempo en los cuales los estados cuantificados cruzan un determinado umbral puede ser directamente calculado, permitiendo la sencilla detección de discontinuidades[2]. En más, cuando tiene lugar una discontinuidad, está producirá un cambio en la derivada de algún estado, del mismo modo que que lo haría una actualización de un estado cuantificado en un paso normal. Por lo tanto, la simulación no necesita ser reiniciada. En conclusión, la detección y manejo de discontinuidades no representa ningún costo computacional extra en comparación a la de cualquier otro paso. Así, el método QSS1 resulta muy eficiente a la hora de simular sistemas discontinuos [8].

Sin embargo, pese a esta ventaja y a las buenas propiedades de este método, que serán tratadas más adelante, QSS1 sólo realiza una aproximación de

[2]En las herramientas de simulación de QSS, las discontinuidades son detectadas usando los estados cuantificados q_i. También puede ser utilizados los estados x_i. en dicho caso, la detección de cruces por cero puede ser más precisa, pero en presencia de condiciones de cruce no lineales, esta detección puede resultar más compleja.

primer orden, y por lo tanto, no es posible obtener resultados precisos sin reducir significativamente el quantum ΔQ_i, lo cual se traduce en un incremento del número de pasos de simulación. Por ello, han sido desarrollados métodos QSS de orde superior.

3.2.2. Método de integración de estados cuantificados QSS2

El método QSS1 si bien es el método que pudo simular sistemas continuos aproximándolos por sistemas cuantificados en forma segura (sin generar modelos ilegítimos), está muy limitado por ser un método de primer orden. Esta limitación hace que para simular un sistema con una precisión medianamente alta el número de pasos sea inaceptable.

El método QSS2 se basa en los mismos principios que QSS1 solo que en lugar de utilizar un una función de cuantificación de orden cero utiliza una de primer orden.

Para poder definir entonces este método se debe ver en primer lugar el como es una función de cuantificación de primer orden.

Usando la idea de la función de cuantificación de orden cero, se puede definir una función de cuantificación de primer orden como una función que genera una trayectoria de salida seccionalmente lineal cuyo valor y pendiente cambia solamente cuando la diferencia entre esta salida y la entrada se vuelve mayor que un valor predeterminado. Este valor, es el quántum. Esta idea puede verse en la figura 3.5.

La trayectoria de salida seccionalmente lineal comienza con el valor de la entrada y con pendiente igual a la entrada y luego, cuando la trayectoria de entrada y de salida difieren en ΔQ, la trayectoria de salida se representa por un nuevo segmento que comienza en el valor de la entrada. Denominando $q(t)$ a la trayectoria de salida, $\dot{q}(t)$ a la pendiente de la misma y $x(t)$ a la entrada, el comportamiento descripto se puede escribir formalmente como:

$$
\begin{aligned}
q(t) &= \begin{cases} x(t) & \text{si } t = t_0 \vee |q(t^-) - x(t^-)| \geq \Delta Q \\ q(t_j) + \dot{q}_j \cdot (t - t_j) & \text{c.o.c} \end{cases} \\
\dot{q}(t_j) &= \begin{cases} \dot{x}(t_j^-) & \text{si } q(t_j) \neq q(t_j^-) \\ 0 & \text{si } t = t_0 \end{cases}
\end{aligned}
$$

Una propiedad fundamental de la función de cuantificación de primer orden descripta es que

$$|x(t) - q(t)| \leq \Delta q \quad \forall t \geq 0 \tag{3.8}$$

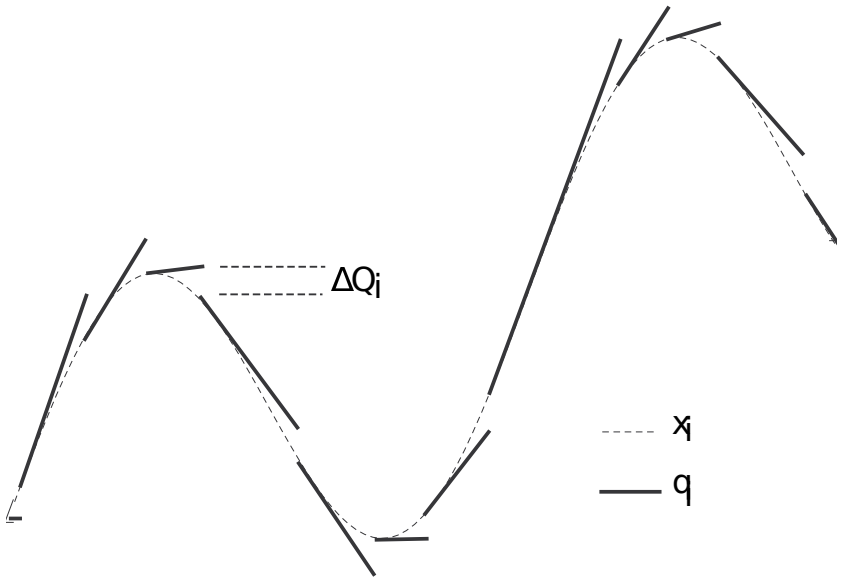


Figura 3.5: Función de cuantificación de primer orden.

Reemplazando las funciones de cuantificación de orden cero de QSS1 por funciones de cuantificación de primer orden, se obtiene un método QSS de segundo orden (QSS2). Utilizando este método, se obtienen sistemas de estados cuantificados de segundo orden.

La definición de QSS2 es practicamente la misma que la de QSS. Los dos métodos difieren únicamente en el modo en que se calcula $q_i(t)$ a partir de $x_i(t)$. Teniendo en cuenta que las trayectorias de las variables cuantificadas $q_i(t)$ en QSS2 son seccionalmente lineales, entonces las trayectorias $\dot{x}(t)$ de las derivadas de los estados también son seccionalmente lineales [3] y entonces las trayectorias $x_i(t)$ de las variables de estado resultan seccionalmente parabólicas.

Como en el caso del método QSS1, el método QSS también puede ser implementado por modelos DEVS de integradores cuantificados y funciones estáticas. Sin embargo, los nuevos modelos atómicos deben tener en cuenta además del valor de la trayectoria de entrada y de salida, la pendiente de las mismas.

[3]Para ser rigurosos, esto solo es cierto para sistemas LTI. En los demás casos, las derivadas de los estados se aproximan mediante trayectorias seccionalmente lineales.

El algoritmo de simulación para QSS2 es similar al de QSS1, excepto que también se debe calcular la pendiente del estado cuantificado $\dot{q}_i(t)$ y la derivada segunda del estado $\ddot{x}_i(t)$, como se muestra a continuación:

Algoritmo 3.2: QSS2.

1 **Mientras** $(t < t_f)$ // simular hasta tiempo final tf
2 $t = \mathbf{min}(t_j^\eta)$ // avanzar tiempo de simulación
3 $i = \mathbf{argmin}(t_j^\eta)$ // el i−ésimo estado cuantificado cambia primero
4 $e = t - t_i^x$ // tiempo transcurrido desde la ultima actualización de xi
5 // actualizar el i−ésimo estado y su derivada
6 $x_i = x_i + \dot{x}_i \cdot e + 0{,}5 \cdot \ddot{x}_i \cdot e^2$
7 $\dot{x}_i = \dot{x}_i + \ddot{x}_i \cdot e$
8 // actualizar el i−ésimo estado cuantificado
9 $q_i = x_i$
10 $\dot{q}_i = \dot{x}_i$
11 $t_i^\eta = \mathbf{min}(\tau > t)$ **tal_que** $|q_i(\tau) - x_i(\tau)| = \Delta Q_i$ // calcular el tiempo del próximo cambio en el i−ésimo estado cuantificado
12 **Para** $j \in [1, n]$ **tal_que** $\dot{x}_j$ **depende_de** q_i
13 $e = t - t_j^x$ // tiempo transcurrido desde la última actualización de xj
14 // actualizar el j−ésimo estado y sus derivadas
15 $x_j = x_j + \dot{x}_i \cdot e + 0{,}5 \cdot \ddot{x}_j \cdot e^2$
16 $\dot{x}_j = f_j(\mathbf{q}(t), t)$ // recalcular la derivada del estado
17 $\ddot{x}_j = \dot{f}_j(\mathbf{q}(t), t)$ // recalcular la derivada segunda del estado
18 $t_j^\eta = \mathbf{min}(\tau > t)$ **tal_que** $|q_j(\tau) - x_j(\tau)| = \Delta Q_j$ // recalcular el tiempo del próximo cambio en el j−ésimo estado cuantificado
19 **Si** $j \neq i$ **entonces** $t_j^x = t$ // última actualización de xj
20 **fin_para**
21 $t_i^x = t$ // última actualización de xi
22 **fin_mientras**

Los pasos del método QSS2 son más costos computacionalmente que los de QSS1. En particular, hay dos evaluaciones de funciones escalares, para computar $\dot{x}_j$ y $\ddot{x}_j$ (lineas 16–17) y el cálculo del tiempo de la próxima actualización del estado cuantificado en la linea 18 involucra la resolución de una ecuación cuadrática. Estos costos adicionales son compensados por el hecho de que QSS2 puede realizar pasos mucho mayores consiguiendo mejores resultados de precision.

3.2.3. Control de error relativo en los métodos QSS

Uno de los problemas aún no explicados es cómo se puede seleccionar el quántum. En esta sección se mostrará un método para tener control de error relativo [21] en los métodos QSS.

Usando cuantificación logarítmica, es decir, haciendo que el quántum sea proporcional a la magnitud de las variables cuantificadas, se logra que los métodos QSS tengan un control de error relativo.

La idea básica de la cuantificación logarítmica consiste en tomar el valor del quántum ΔQ_i proporcional al valor de x_i en el momento en que se alcanza una condición de evento.

El problema de escoger el quántum de esa manera se da cuando x_i es próximo a cero. En este caso, ΔQ_i resultará demasiado pequeño y se producirán una gran cantidad de eventos innecesarios. Entonces, la elección correcta del quántum es:

$$\Delta Q_i(t) = \max\left(E_{rel_i} \cdot |x_i(t_k)|, \Delta Q_{i_min}\right) \tag{3.9}$$

donde t_k es el último instante de tiempo en que ocurrió un evento en x_i

Para poder ver que de esta forma se logra un control del error relativo, se deberá hacer un análisis del error. Dado el siguiente sistema LTI:

$$\dot{\mathbf{x}}_\mathbf{a}(t) = A \cdot \mathbf{x}_\mathbf{a}(t) + B \cdot \mathbf{u}(t) \tag{3.10}$$

definiendo $\mathbf{\Delta x}(t) \triangleq \mathbf{q}(t) - \mathbf{x}(t)$, los métodos QSS lo aproximan por

$$\dot{\mathbf{x}}(t) = A \cdot (\mathbf{x}(t) + \mathbf{\Delta x}) + B \cdot \mathbf{u}(t) \tag{3.11}$$

Haciendo ahora la diferencia entre las Ecs.(3.10) y (3.11), se obtiene la ecuación para el error $\mathbf{e}(t) = \mathbf{x}(t) - \mathbf{x}_\mathbf{a}(t)$:

$$\dot{\mathbf{e}}(t) = A \cdot (\mathbf{e}(t) + \mathbf{\Delta x}(t)) \tag{3.12}$$

donde $|\mathbf{\Delta x}(t)| \leq \mathbf{\Delta Q}(t)$ para todo $t \geq t_0$. Luego, trabajando con las ecuaciones y aplicando el teorema 2.2 de [21] se demostró que:

$$|\mathbf{e}(t)| \leq (I - RE_{rel})^{-1} R \max\left(E_{rel} \cdot |x_{max}|, \Delta Q_{min}\right) \tag{3.13}$$

donde A es una matriz Hurwitz con forma canónica de Jordan $\Lambda = V^{-1}AV$ y $R \triangleq |V| \left|[Re(\Lambda)]^{-1} V^{-1} H\right|$

En esta ecuación se ve claramente que la cota de error es proporcional al máximo valor de $\mathbf{x}_a(t)$, Es decir, se logra un control del error relativo.

Además, en la mayoría de las aplicaciones se elije E_{rel} muy pequeño (un valor típico es $E_{rel} = 0{,}01$). En este caso se puede aproximar la ecuación (3.13) por:

$$|\mathbf{e}(t)| \leq R \max\left(E_{rel} \cdot |x_{max}|, \Delta Q_{min}\right) \tag{3.14}$$

3.3. Manejo de discontinuidades

En los métodos de tiempo discreto se deben detectar exactamente los instantes en los cuales ocurren las discontinuidades, ya que no se puede integrar pasando a través de una discontinuidad. En el caso de los eventos temporales simplemente se debe ajustar el paso para que éste coincida con el tiempo de ocurrencia del evento. Frente a eventos de estado, en cambio, se debe iterar para encontrar el instante preciso en el que ocurre dicho evento.

En el caso de los métodos de QSS, todos estos problemas desaparecen. Para poder ver esto, se analizará un ejemplo sencillo.

La figura 3.6 muestra dos posibles situaciones en que puede estar una pelota rebotando. Mientras la pelota está en el aire se tiene en cuenta la influencia de la fuerza de gravedad y del rozamiento del aire. Cuando la pelota está en contacto con el suelo, se considera a la misma como el modelo de un resorte comprimido con un amortiguador

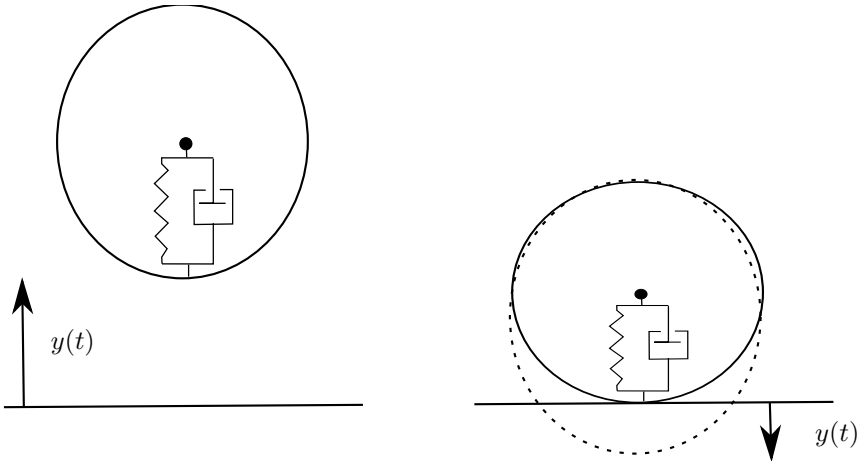


Figura 3.6: Pelota Rebotando

Este sistema puede ser modelado mediante las siguientes ecuaciones:

$$\begin{aligned} \dot{y}(t) &= v(t) \\ \dot{v}(t) &= \begin{cases} -g - \frac{b_a}{m} \cdot v(t) \cdot |v(t)| & \text{si } y(t) > 0 \\ -g - \frac{k}{m} \cdot y(t) - \frac{b}{m} \cdot v(t) & \text{c.o.c} \end{cases} \end{aligned} \tag{3.15}$$

donde g es la aceleración de la gravedad, m es la masa de la pelota, k es la constante del resorte, b_a es el coeficiente de rozamiento del aire y b es el coeficiente de amortiguamiento del amortiguador.

El problema con este modelo es que el lado derecho de la última ecuación es discontinuo. Si un método de integración numérico realizase un paso a través de la discontinuidad, se obtendría un resultado cuyo error sería inaceptable.

Como se mencionó antes, los métodos de integración convencionales resuelven este problema encontrando el instante exacto en que ocurre la discontinuidad. Luego, avanzan la simulación hasta ese instante y reinician la simulación a partir de ese instante con las condiciones iniciales obtenidas en el momento de la discontinuidad .

A pesar de que esta metodología generalmente funciona bien, agrega un costo computacional adicional importante, ya que encontrar el instante de ocurrencia de una discontinuidad implica realizar algunas iteraciones y además, reiniciar la simulación también puede ser costoso.

Como se vio en este capítulo, las trayectorias de los métodos QSS son seccionalmente constantes, lineales o parabólicas según sea el orden del método. En consecuencia, el instante de ocurrencia de un evento puede ser detectado analíticamente.

A pesar de que el evento asociado a una discontinuidad debe ocurrir en el instante exacto de ocurrencia de la misma, los métodos QSS no precisan ser reinicializados luego del mismo. El motivo es simplemente que las discontinuidades ocurren regularmente en los métodos QSS dado que las trayectorias $q_i(t)$ en los mismos son discontinuas. En consecuencia, en los métodos QSS las discontinuidades tienen el mismo efecto que un paso normal [8].

Esta es la razón por la cual los métodos QSS son en general más eficientes que los métodos convencionales en el manejo de discontinuidades. En aquellos sistemas en los que ocurren discontinuidades más rápidamente que la dinámica del sistema continuo, lo cual suele ser muy común en los modelos de circuitos de electrónica de potencia, los métodos QSS reducen significativamente el tiempo computacional requerido en la simulación en comparación con los métodos tradicionales de simulación de ODEs [8].

3.4. Propiedades teóricas de los métodos QSS

Usando notación vectorial, el sistema:

$$\begin{aligned} \dot{x}_{a_1} &= f_1(x_{a_1}, x_{a_2}, \cdots, x_{a_n}, u_1, \cdots, u_m) \\ \dot{x}_{a_2} &= f_2(x_{a_1}, x_{a_2}, \cdots, x_{a_n}, u_1, \cdots, u_m) \\ &\vdots \\ \dot{x}_{a_n} &= f_n(x_{a_1}, x_{a_2}, \cdots, x_{a_n}, u_1, \cdots, u_m) \end{aligned}$$

toma la forma:

$$\dot{\mathbf{x}}_\mathbf{a}(t) = \mathbf{f}(\mathbf{x}_\mathbf{a}(t), \mathbf{u}(t)) \tag{3.16}$$

mientras que la aproximación QSS

$$\begin{aligned} \dot{x}_1 &= f_1(q_1, q_2, \cdots, q_n, u_1, \cdots, u_m) \\ \dot{x}_2 &= f_2(q_1, q_2, \cdots, q_n, u_1, \cdots, u_m) \\ &\vdots \\ \dot{x}_n &= f_n(q_1, q_2, \cdots, q_n, u_1, \cdots, u_m) \end{aligned}$$

se puede escribir como

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{q}(t), \mathbf{u}(t)) \tag{3.17}$$

Definiendo $\mathbf{\Delta x}(t) = \mathbf{q}(t) - \mathbf{x}(t)$, esta última se puede reescribir como:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t) + \mathbf{\Delta x}(t), \mathbf{u}(t)) \tag{3.18}$$

Esta última ecuación solo difiere del sistema original (3.16) por la presencia del termino de perturbación $\mathbf{\Delta x}(t)$.

Una propiedad fundamental de las funciones de cuantificación usadas por los métodos QSS es que $x_i(t)$ y $q_i(t)$ nunca difieren entre sí en más que el quántum ΔQ_i. Entonces, cada componente $\Delta x_i(t)$ del vector $\mathbf{\Delta x}(t)$ está acotado por el quántum ΔQ_i.

Como consecuencia de este hecho, se puede analizar el efecto de usar las aproximaciones QSS como un problema de ODEs con perturbaciones acotadas.

Basándose en esta idea, la primera propiedad que se probó en el contexto de los métodos QSS fue la de *convergencia* [5]. El análisis muestra que la solución de la Ec.(3.17) aproxima a la solución de la Ec.(3.16) cuando el quántum ΔQ_i más grande se elige lo suficientemente chico. La importancia de esta propiedad reside en que se puede lograr un error de simulación arbitrariamente chico utilizando una cuantificación lo suficientemente chica.

Una condición suficiente que asegura que las trayectorias del sistema de ecuaciones (3.17) converge a las trayectorias de las Ecs.(3.16) es que la función $\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t))$ sea localmente Lipschitz.

A pesar de que la convergencia es una propiedad teórica importante, no ofrece ninguna información cuantitativa sobre la relación entre el quántum y el error, y además no establece ninguna condición sobre el dominio de estabilidad.

Las propiedades de estabilidad de los métodos QSS fueron estudiadas en [5] buscando una función de Lyapunov para el sistema perturbado. Este análisis muestra que cuando el sistema de la Ec.(3.16) tiene una punto de equilibrio asintóticamente estable, para cualquier región arbitrariamente pequeña entorno al punto de equilibrio se puede encontrar una cuantificación tal que la solución de la Ec.(3.17) termine dentro de esa región. Además, a partir de este análisis se puede obtener un algoritmo que permite calcular el quántum necesario.

Una condición suficiente que asegura la estabilidad numérica de los métodos QSS cerca de un punto de equilibrio analíticamente estable es que la función $\mathbf{f}$ sea continua y continuamente diferenciable. Por lo tanto, la condición de estabilidad es más fuerte que la de convergencia.

Entonces, los métodos de QSS poseen herramientas que pueden ser aplicadas a sistemas no lineales para elegir el quántum de manera de asegurar que el error de simulación en régimen permanente sea menor que una cota prefijada. A pesar de que este resultado representa una gran ventaja sobre los métodos de tiempo discreto clásicos, en los cuales la estabilidad se estudia usualmente en el contexto de sistemas lineales invariantes en el tiempo (LTI)) únicamente, el algoritmo es algo complicado y requiere usar una función de Lyapunov de la Ec.(3.16) que en general no se puede obtener fácilmente. Luego, este análisis de estabilidad es mas importante desde el punto de vista teórico que práctico.

Tal como en los métodos de tiempo discreto, la propiedad más interesante de los métodos QSS se obtiene del análisis de aplicar los mismos a sistemas LTI.

El resultado principal de este análisis se encuentra en [6] y establece que el error en la simulación QSS de un sistema LTI asintóticamente estable está siempre acotado. Esta cota de error puede ser calculada a partir del quántum y algunas propiedades geométricas del sistema y no depende ni de las condiciones iniciales, ni de las trayectorias de entrada. Además, permanece constante durante la simulación.

Un sistema LTI se puede escribir como:

$$\dot{\mathbf{x}}_\mathbf{a}(t) = \mathbf{A} \cdot \mathbf{x}_\mathbf{a}(t) + \mathbf{B} \cdot \mathbf{u}(t) \tag{3.19}$$

con $\mathbf{A}$ y $\mathbf{B}$ matrices de valores reales de $n \times n$ y $m \times n$ respectivamente, donde n es el orden del sistema y m es el número de entradas del mismo.

Una aproximación QSS de este sistema está dada por:

$$\dot{\mathbf{x}}(t) = \mathbf{A} \cdot \mathbf{q}(t) + \mathbf{B} \cdot \mathbf{u}(t) \tag{3.20}$$

Sea $\mathbf{x}_\mathbf{a}(t)$ la solución de la Ec.(3.19) y $\mathbf{x}(t)$ la solución de la aproximación QSS de la Ec.(3.20) comenzando desde las mismas condiciones iniciales ($\mathbf{x}_\mathbf{a}(t_0) = \mathbf{x}(t_0)$).

Para un sistema LTI analíticamente estable, el error global de simulación está acotado por la siguiente desigualdad:

$$|\mathbf{x}(t) - \mathbf{x}_\mathbf{a}(t)| \preceq |\mathbf{V}| \cdot |\text{Real}\{\mathbf{\Lambda}\}^{-1} \cdot \mathbf{\Lambda}| \cdot |\mathbf{V}^{-1}| \cdot \mathbf{\Delta Q} \tag{3.21}$$

para todo $t \geq t_0$.

donde $\Lambda = V^{-1} A V$ es la descomposición modal de A, el vector $\mathbf{\Delta Q} \triangleq [\Delta Q_1, \cdots, \Delta Q_n]^T$ está compuesto por el conjunto de los quántum usados para cada estado, el símbolo $|\cdot|$ representa el valor absoluto componente a componente de un vector o matriz y el símbolo '$\preceq$' compara componente a componente vectores de igual largo.

Entonces, la Ec.(3.21) brinda una cota de error global de simulación para cada componente del vector de estado del sistema. Esta cota de error depende linealmente del quántum.

Se puede ver entonces que:

> Los métodos QSS tienen un control de error intrínseco, sin necesidad de recurrir a normas de adaptación.

Cabe notar que esta cota de error global implica que las soluciones numéricas no pueden divergir. Finalmente, una observación muy importante es:

> Los métodos QSS permanecen numéricamente estables sin necesidad de utilizar fórmulas implícitas.

Además del análisis basado en perturbaciones que se mostró en esta sección, en [25] se desarrolló un modo alternativo para demostrar la estabilidad de los métodos QSS haciendo uso de un equivalente de las aproximaciones QSS usando multirate de tiempo discreto.

3.5. Métodos QSS Linealmente Implícitos

Pese a las ventajas expuestas, los métodos QSS1 y QSS2 resultan ineficeintes a la hora de simular sistemas stiff. En presencia de dinámicas lentas y rápidas, estos métodos introducen oscilaciones espurias de alta frecuencia que provocan una gran cantidad de pasos, con su consecuente costo computacional [4].

Para superar este inconveniente, la familia de métodos QSS fue extendida con un conjunto de algoritmos llamados QSS Linealmente Implícitos (LIQSS), los cuales resultan apropiados para la simulación de sistemas stiff [26]. Los métodos LIQSS combinan los principios de los métodos QSS con los de los métodos linealmente implícitos clásicos. Hay algoritmos que realizan aproximaciones de primer y segundo , LIQSS1 y LIQSS2 respectivamente.

La idea principal detrás de estos métodos está inspirada en los métodos implícitos clásicos que evalúanlas derivadas de los estados en instantes de tiempo futuros. En los métodos clásicos, estas evaluaciones requieren de iteraciones y/o inversiones matriciales para resolver ecuaciones implícitas. Sin embargo, teniendo en cuenta que en los métodos QSS el valor futuro de un estado cuantificado se conoce a priori (vale $q_i(t) \pm \Delta Q_i$), la implementación de los algoritmos LIQSS resulta explícita y no requiere de iteraciones ni de inversiones matriciales.

Los métodos LIQSS comparten con los QSS la definición de la Ec. (3.7), sin embargo, los estados cuantificados se calculan de un modo más complejo, teniendo en cuenta el signo de la derivada de los estados.

En LIQSS1 la idea es que $q_i(t)$ tome el valor $x_i(t) + \Delta Q_i(t)$ cuando el valor futuro de la derivada del estado $\dot{x}_i(t^+)$ sea positiva. Por el contrario, cuando el valor futuro de la derivada del estado es negativa, $q_i(t)$ toma el valor $x_i(t) - \Delta Q_i(t)$. Luego, cuando x_i alcanza a q_i, se da un nuevo paso. De este modo, el valor de estado cuantificado resulta ser el valor futuro de del estado, y las derivadas en la Ec. (3.7) se calculan usando ese valor futuro del estado, del mismo que como sucede en los métodos implícitos clásicos.

A los efectos de predecir el signo de la futura derivada del estado, se utiliza la siguiente aproximación lineal de la dinámica del i-ésimo estado:

$$\dot{x}_i(t) = A_{i,i} \cdot q_i(t) + u_{i,i}(t) \tag{3.22}$$

donde$A_{i,i} = \frac{\partial f_i}{\partial x_i}$ es el i-ésimo término de la diagonal principal de la matriz Jacobiana y $u_{i,i}(t) = f_i(\mathbf{q}(t), t) - A_{i,i} \cdot q_i(t)$ es un coeficeinte afín.

Puede suceder que $A_{i,i} \cdot (x_i(t) + \Delta Q_i) + u_{i,i}(t) < 0$, esto es, que cuando al proponer $q_i(t) = x_i(t) + \Delta Q_i$ la derivada $\dot{x}_i(t^+)$ resulte negativa. También

puede suceder que $A_{i,i} \cdot (x_i(t) - \Delta Q_i) + u_{i,i}(t) > 0$. Así, $q_i(t)$ no puede ser elegido de modo tal que resulte un valor futuro para $x_i(t)$. Sin embargo, en ese caso, q_i puede elegirse de modo tal que $\dot{x}_i(t) = 0$. Ese punto de equilibrio para q_i puede ser calculado de la Ec. (3.22) como

$$q_i = -\frac{u_{i,i}}{A_{i,i}} \tag{3.23}$$

Entonces, el algoritmo de simulación de LIQSS1 es como se muestra a continuación:

Algoritmo 3.3: LIQSS1.

Mientras $(t < t_f)$ // simular hasta tiempo final tf
  $t = \mathbf{min}(t_j^\eta)$ // avanzar tiempo de simulación
  $i = \mathbf{argmin}(t_j^\eta)$ // el i−ésimo estado cuantificado cambia primero
  $e = t - t_i^x$ // tiempo transcurrido desde la ultima actualización de xi
  $x_i = x_i + \dot{x}_i \cdot e$ // actualizar el i−ésimo estado
  $q_i^- = q_i$ // almacenar el valor previo de qi
  $\dot{x}_i^- = \dot{x}_i$ // almacenar el valor previo de dxi/dt
  $\dot{x}_i^+ = A_{i,i} \cdot (x_i + sign(\dot{x}_i) \cdot \Delta Q_i) + u_{i,i}$ // estima de la futura derivada usando una aproximación lineal almacenada en los arreglos A y u
  Si $(\dot{x}_i \cdot \dot{x}_i^+ > 0)$ // la derivada del estado conserva su signo previo
    $q_i = x_i + sign(\dot{x}_i) \cdot \Delta Q_i$
  sino //the state changes its direction
    $q_i = -u_{i,i}/A_{i,i}$ // choose qi such that dxi/dt = 0
  fin_si
  $t_i^\eta = \mathbf{min}(\tau > t)$ **tal_que** $x_i(\tau) = q_i$ // compute the time of then next change in the i−th quantized state
  for each $j \in [1, n]$ such that $\dot{x}_j$ depends on q_i
    $e = t - t_j^x$ // elapsed time since last xj update
    $x_j = x_j + \dot{x}_j \cdot e$ // update j−th state value
    if $j \neq i$ **then** $t_j^x = t$// last xj update
    $\dot{x}_j = f_j(\mathbf{q}, t)$ // recompute j−th state derivative
    $t_j^\eta = \mathbf{min}(\tau > t)$ subject to $x_j(\tau) = q_j$ or $|q_j - x_j(\tau)| = 2\Delta Q_j$ // recompute the time of the next change in the j−th quantized state
  end for
  // update linear approximation coefficients
  $A_{i,i} = (\dot{x}_i - \dot{x}_i^-)/(q_i - q_i^-)$ // Jacobian diagonal entry
  $u_{i,i} = \dot{x}_i - A_{i,i} \cdot q_i$ // affine coefficient

```
25    t_i^x = t // last xi update
26  end while
```

Puede observarse que LIQSS1 sólo requiere de algunos cálculos adicionales con respecto a QSS1. En particular, LIQSS1 estima el valor futuro de la derivada del estado utilizando un modelo lineal(linea 8), estima el término $A_{i,i}$ de la diagonal principal de la matriz Jacobiana y el coeficiente afín (lineas 23–24)[4].

Nótese también que en la linea 20 el algoritmo chequea la condición adicional $|q_j - x_j(\tau)| = 2\Delta Q_j$,ya que un cambio en la variable q_i puede cambiar el signo de la derivada del estado $\dot{x}_j(t)$ de modo tal que x_j deje de aproximarse a q_j. En ese caso, se verá luego que el hecho de que x_j no siempre se aproxime a q_j puede resultar en una solución ineficiente para algunos sistemas stiff.

LIQSS1 comparte las principales ventajas con QSS1 y logra integrareficientemente sistemas stiff, siempre y caundo la rigidez se deba a la presencia de términos grandes en la diagonal principal de la matriz Jacobiana. Al igual que con QSS1, este método no puede logra buena exactitud, por lo que se han desarrollado métodos LIQSS de orden superior.

El método de segundo orden LIQSS2 combina la ideas de QSS2 con LIQSS1. En este caso, al igual que en QSS2, el estado cuantificado sigue trayectorias seccionalmente lineales. Este algoritmo será explicado más adelante.

3.6. QSS en sistemas stiff

La Figura 3.7 muestra el resultado de simular el sistema

$$\dot{x} = -0{,}1 \cdot (x - 10{,}5); \qquad x_0 = 0 \tag{3.24}$$

utilizando QSS1 con quantum $\Delta Q = 1$. Puede verse que, pese a que la solución analítica debería converger a $x(\infty) = 10{,}5$, la simulación termina en oscilaciones espurias en torno al punto de equilibrio. Si este sistema fuera la parte rápida de un sistema más grande con partes mucho más lentas (es decir, un sistema stiff), estas oscilaciones espurias implicarían realizar muchos cálculos innecesarios y costosos computacionalmente.

[4]Nótese que $A_{i,i}$ es calculado usando dos valores de $\dot{x}_i = f_i(\mathbf{q}, t)$. El primero es previo al cambio en q_i, y el segundo es obtenido luego del cambio. El hecho de que el primer valor pueda ser obtenido para un instante de tiempo t anterior, no debería afectar los cáclulos dado que los cambios en f_i debido al avance del tiempo son tratados como pasos de entrada.

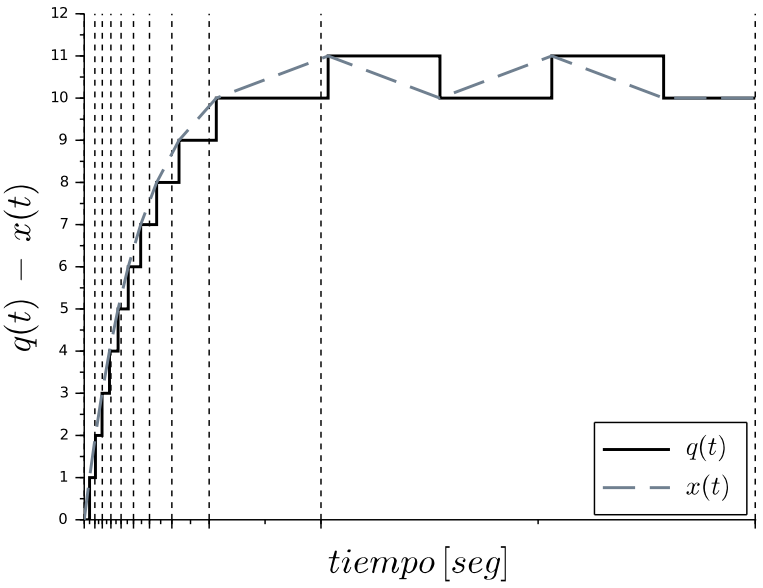


Figura 3.7: Simulación del sistema 3.24 con QSS1.

Los métodos de LIQSS [26] solucionan este problema ya que, con una idea inspirada en los métodos implícitos clásicos, evalúan las derivadas usando el valor futuro de los mismos. Esta evaluación futura que produce que los métodos clásicos deban resolver un sistema de ecuaciones en cada paso, en los métodos de LIQSS no tiene el mismo efecto ya que el futuro estado cuantificado se conoce de antemano (es, $q_i(t) \pm \Delta Q_i$). Por esto, la implementación de los métodos LIQSS resulta en algoritmos que no requieren de iteraciones.

La Figura 3.8 muestra el resultado de simular nuevamente el sistema de la Ec. 3.24 pero ahora utilizando el método LIQSS1, donde las oscilaciones en torno al punto de equilibrio no están presentes.

Dado que los LIQSS sólo miran el valor futuro de cada variable individualmente, sólo pueden evitar las oscilaciones espurias que se deben al cambio de dicha variable. Sin embargo, hay muchas situaciones en las que las oscilaciones espurias se dan de a pares de variables, donde el cambio de una variable provoca un cambio en la derivada de la segunda variable y viceversa. Este tema será abordado en los capítulos siguientes.

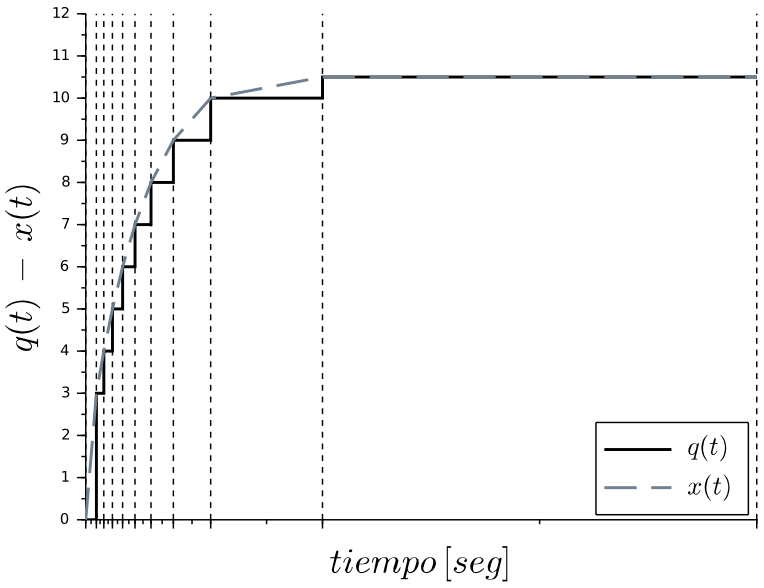


Figura 3.8: Simulación del sistema 3.24 con LIQSS1.

3.7. Implementación de los métodos QSS

La manera más sencilla de implementar los métodos QSS es construyendo un modelo DEVS equivalente, donde los eventos representan cambios en las variables cuantificados. Basado en esta idea, toda la familia de métodos QSS fue implementada en PowerDEVS [27], una plataforma de simulación basada en DEVS especialmente diseñada y adaptada para la simulación de sistemas híbridos basada en los métodos QSS. Además, los métodos QSS también fueron cumplimentados en DEVS en una biblioteca de Modelica[28], también puede encontrarse una implementación de QSS1 en CD++ [29] y en VLE [30].

Pese a resultar simple, la implementación de los algorimos QSS en DEVS resulta ineficiente debido que los cálculos involucrados en cada paso de simulación son realizados por diferentes modelos DEVS atómicos por miedo del intercambio de eventos. El problema es que el manejo y la coordinación de los correspondientes mensajes resulta más computacionalmente costoso que los pasos de simulación en sí. Por este motivo, la familia completa de métodos QSS fue implmenteada en el *stand–alone QSS solver* [31], el cual

mejora por más de un orden de magnitud los resultados obtenidos con el enfoque basado en DEVS. Desde el punto de vista del formalismo DEVS, puede pensarse que el *stand–alone QSS solver* reemplaza el modelo DEVS acomplado por un único modelo atómico.

Capítulo 4

Método LIQSS modificado de primer orden

En esta sección, en primera instancia se analizan las limitaciones de los algoritmos LIQSS respecto a la aparición de oscilaciones rápidas en sistemas en los que la rigidez no se debe exclusivamente a la presencia de términos grandes en la diagonal principal de la matriz Jacobiana. Luego, se propone una idea para solucionar dicho inconveniente, y usando este enfoque, se propone un método LIQSS modificado de primer orden (mLIQSS1).

4.1. Limitaciones de LIQSS

La simulación de un sistema estable de primer orden con el algoritmo QSS1 produce usualmente como resultado una trayectoria de estado que presenta oscilaciones al rededor del punto de equilibrio[4]. Estas oscilaciones son la razón por la cual QSS no resulta eficiente a la hora de simular sistemas *stiff*.

Este problema es resuelto por LIQSS, que previene la aparición de esas oscilaciones tomando como estado cuantificado un valor futuro del estado, esto es $q_i = x_i + \text{sign}(\dot{x}_i) \cdot \Delta Q_i$. Cuando esto no es posible, es decir, cuando $q_i = x_i + \Delta Q_i$ produce que $\dot{x}_i < 0$, o bien que proponer $q_i = x_i - \Delta Q_i$ produce que $\dot{x}_i > 0$, entonces LIQSS1 detecta la cercanía a un punto de equilibrio que puede ser computado utilizando una aproximación lineal como la mostrada en la Ec (3.23).

Sin embargo, LIQSS1 no puede asegurar que q_i sea siempre un valor futuro de x_i ya que, luego de computar q_i, $\dot{x}_i$ puede cambiar de signo debido al cambio en otro estado cuantificado q_j. En tal caso, también puede suceder

que en el próximo cambio de q_i produzca un cambio en el signo de $\dot{x}_j$. Esta situación puede ser causante de la aparición de oscilaciones involucrando a los estados x_i y x_j.

Se ilustra esta situación con el siguiente ejemplo. Consideremos el siguiente sistema

$$\begin{aligned} \dot{x}_1 &= -x_1 - x_2 + 0{,}2 \\ \dot{x}_2 &= x_1 - x_2 + 1{,}2 \end{aligned} \tag{4.1}$$

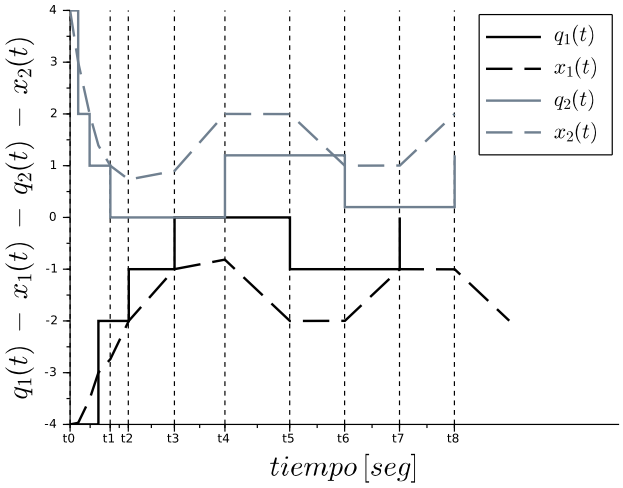
con condiciones iniciales $\mathbf{x_0} = [-4 \quad 4]^T$ y un quantum $\Delta Q_{1,2} = 1$.

Los sucesivos pasos de la simulación de este sistema utilizando el algoritmo LIQSS1 puede verse en la Tabla 4.1.Puede verse allí que en el tiempo $t_6 = 7{,}01$, los estados y sus correspondientes versiones cuantificadas son idénticos a los que tienen lugar para el tiempo $t_2 = 2{,}95$. Así, en lugar de arribar a un punto de equilibrio, la simulación presenta una oscilación con un período de $t_6 - t_2 = 4{,}06$, como es mostrado en la Figura 4.1.

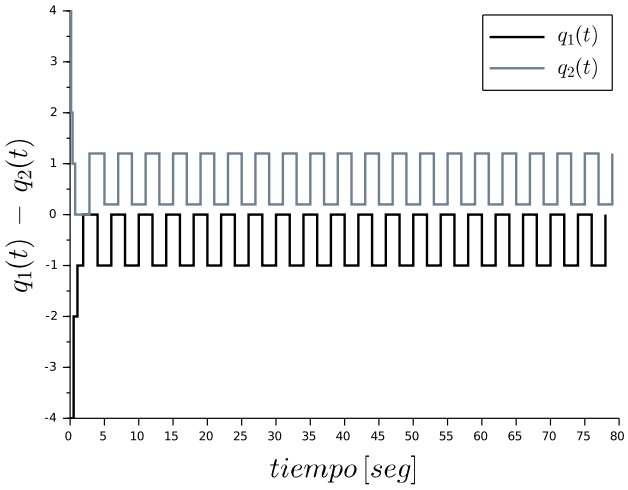
	tiempo	q_1	q_2	$\dot{x}_1$	$\dot{x}_2$	próximo tiempo
t_0	0	-4	4	0.2	-6.8	0.29
⋮	⋮	⋮	⋮	⋮	⋮	⋮
t_1	1.65	0	0	0	1.2	2.95
t_2	2.95	0	1.2	-1	0	4.21
t_3	4.21	-1	1.2	0	-1	5.01
t_4	5.01	-1	0.2	1	0	6.01
t_5	6.01	0	0.2	0	1	7.01
t_6	7.01	0	1.2	-1	0	8.01
t_7	8.01	-1	1.2	0	-1	9.01
t_8	9.01	-1	0.2	1	0	10.01

Tabla 4.1: Evolución de la simulación de (4.1) con el algoritmo LIQSS1.

Si las Ecuaciones (4.1) formasen parte de un sistemas de mayor tamaño con dinámicas más lentas, esas oscilaciones espurias provocarían una gran cantidad de pasos de simulación que resultarían innecesarios. De este modo, LIQSS1 no puede simular eficientemente sistemas *stiff* que tengan el tipo de estructura antes descripto.



(a) Con $t_f = 10$ segundos.



(b) Con $t_f = 100$ segundos.

Figura 4.1: Simulación del sistema dado por (4.1) con LIQSS1.

4.2. Idea básica

A los efectos de evitar las oscilaciones entre pares de variables, se propone verificar si una actualización en un estado cuantificado produce un cambio

de signo en la derivada de algún otro estado. Si eso sucede, se verifica si luego si una eventual actualización del segundo estado cuantificado provoca un cambio de signo en la derivada del anterior estado. Ante esta situación, se espera que ambas variables experimenten oscilaciones espurias, y a los efectos de prevenirlas, se aplicará simultáneamente un cambio en ambos estados cuantificados utilizando un paso linealmente implícito.

Si bien que esta estrategia no resuelve estructuras rígidas genéricas, si evitará la aparición de oscilaciones entre pares de variables, lo cual cubre en la práctica, varios casos.

Para poder efectuar los cambios simultáneos en pares de variables, se hace uso de la semejanza entre los pasos de LIQSS1 con los de Backward Euler. De hecho, un cambio en la variables cuantificada q_i puede ser computado alternativamente como

$$q_i = x_i + h \cdot (A_{i,i} \cdot q_i + u_{i,i}) \tag{4.2}$$

donde h es el pasó máximo tal que

$$|q_i - x_i| \leq \Delta Q_i \tag{4.3}$$

Nótese que, considerando que h es finito, entonces $q_i = x_i + signo(\dot{x}_i) \cdot \Delta Q_i$ como fue asignado en la linea 10 del Algoritmo 3.3. Por otro lado, cuando h es infinito, entonces $(A_{i,i} \cdot q_i + u_{i,i}) = 0$ como fue asignado en la linea 12.

En otras palabras, el cálculo de q_i puede ser pensado como un paso de Backward Euler usando el modelo de la aproximación lineal[1] de la Ecuación (3.22) con un paso h definido como el máximo paso que cumple con la desigualdad dada por la Ecuación (4.3).

Nótese además, que esta formulación no requiere verificar el signo de la derivada de los estados.

4.3. LIQSS1 modificado (mLIQSS1)

Utilizando como base la idea arriba expresada, la modificación introducida al algoritmo LIQSS1 consiste en el agregado de una condición para verificar si luego de actualizar un estado cuantificado el signo de las derivadas de otros estados no cambia. Para comprobar esta condición, para cada

[1] Realizar pasos implícitos utilizando una aproximación lineal es similar a lo realizado por los algoritmos Linealmente Implícitos. Es justamente a esto que se debe el nombre de QSS *Linealmente Implícitos*

par de variables x_i, x_j, tal que cada una influencia directamente la derivada de la otra, se utiliza la siguiente aproximación lineal.

$$\begin{aligned} \dot{x}_i &= A_{ii} \cdot q_i + A_{ij} \cdot q_j + u_{ij} \\ \dot{x}_j &= A_{ji} \cdot q_i + A_{jj} \cdot q_j + u_{ji} \end{aligned} \tag{4.4}$$

Aquí, $A_{i,j} = \frac{\partial f_i}{\partial x_j}(\mathbf{q}, t)$ es el i,j-ésimo término de la matriz Jacobiana, y $u_{ij} = f_i(\mathbf{q}, t) - A_{ii} \cdot q_i - A_{ij} \cdot q_j$ es un coeficiente afín.

Si el nuevo valor de q_i no produce un cambio de signo en la derivada de ningún otro estado, calculada con la aproximación lineal de la Ecuación (4.4), el algoritmo se comporta de manera idéntica a LIQSS1. De otro modo, se propone un nuevo valor para q_j que vaya en la nueva dirección de x_j. Luego, se verifica si este valor propuesto para q_j produce a su vez un cambio en el signo de $\dot{x}_i$. Si no es el caso, se ignora el cambio propuesto para q_j y el algoritmo procede del mismo modo que LIQSS1. En caso contrario, se sabe que pueden tener lugar oscilaciones entre los estados x_i y x_j, por que los estados cuantificados q_i y q_j son computados usando un paso de Backward Euler en el modelo lineal de la Ecuación (4.4).

Definiendo

$$\mathbf{q}_{ij} \triangleq \begin{bmatrix} q_i \\ q_j \end{bmatrix} \ , \ \mathbf{x}_{ij} \triangleq \begin{bmatrix} x_i \\ x_j \end{bmatrix} \ , \ \dot{\mathbf{x}}_{ij} \triangleq \begin{bmatrix} \dot{x}_i \\ \dot{x}_j \end{bmatrix} \tag{4.5}$$

y

$$\mathbf{A}_{ij} = \begin{bmatrix} A_{ii} \ A_{ij} \\ A_{ji} \ A_{jj} \end{bmatrix} \ , \ \mathbf{u}_{ij} \triangleq \begin{bmatrix} u_{ij} \\ u_{ji} \end{bmatrix} \tag{4.6}$$

el paso con Backward Euler está dado por la ecuación

$$\mathbf{q}_{ij} = \mathbf{x}_{ij} + h \cdot \dot{\mathbf{x}}_{ij} = \mathbf{x}_{ij} + h \cdot (\mathbf{A}_{ij} \cdot \mathbf{q}_{ij} + \mathbf{u}_{ij}) \tag{4.7}$$

que puede ser reescrita como

$$\mathbf{q}_{ij} = (\mathbf{I} - h \cdot \mathbf{A}_{ij})^{-1} \cdot (\mathbf{x}_{ij} + h \cdot \mathbf{u}_{ij}) \tag{4.8}$$

donde h es calculado como el máximo paso tal que la diferencia entre los estados y sus versiones cuantificadas esté acotada por el quantum.

Dado un paso de integración h, la Ecuación (4.8) permite calcular los estados cuantificados q_i y q_j. El valor de h debe ser tal que los estados cuantificados no difieran de los estados en un cantidad que exceda al quantum, esto es,

$$|q_i - x_i| \leq \Delta Q_i \ , \ \ |q_j - x_j| \leq \Delta Q_j \tag{4.9}$$

El valor máximo para h que satisface ambas inecuaciones puede ser obtenido analíticamente resolviendo la Ecuación (4.8) para h considerando

$q_i = x_i \pm \Delta Q_i$ y $q_j = x_j \pm \Delta Q_j$ y tomando el mínimo valor entre las diferentes soluciones. Alternativamente, h puede ser hallado por medio de iteraciones en la Ecuación (4.8). En cualquier caso, la Ecuación (4.8) en conjunto con las restricciones impuestas por las Inecuaciones (4.9), implícitamente definen una función que calcula el máximo valor para h que satisface esas restricciones, esto es,

$$h_{\text{máx}} = \text{max\_be\_step}(\mathbf{x}_{ij}) \triangleq \text{máx}(h : |q_i - x_i| \leq \Delta Q_i \wedge |q_j - x_j| \leq \Delta Q_j) \quad (4.10)$$

Así, el algoritmo de simulación mLIQSS1 es idéntico al de LIQSS1 hasta la linea 14. Luego, continua como sigue:

Algoritmo 4.1: mLIQSS1.

```
15  for each j ∈ [1,n] such that (i≠j and A_ij · A_ji ≠ 0)
16    e = t − t_j^x // elapsed time since last xj update
17    x_j^aux = x_j + ẋ_j · e // store updated j−th state value on
          auxiliary variable
18    u_ji = u_jj − A_ji · q_i^− // affine coefficient
19    ẋ_j^+ = A_ji · q_i + A_jj · q_j + u_ji // next j−th state der. est.
20    if (ẋ_j · ẋ_j^+ < 0) // update in qi => change of sign in
          dxj/dt
21      q_j^+ = x_j^aux + sign(ẋ_j^+) · ΔQ_j // update qj in future xj's
            direction
22      u_ij = u_ii − A_ij · q_j // affine coefficient
23      ẋ_i^+ = A_ii · q_i + A_ij · q_j^+ + u_ij // next i−th state der. est.

24      if (ẋ_i · ẋ_i^+ < 0) // update in qj => change of sign in
            dxi/dt
25        // presence of oscillations
26        q_j^− = q_j //store previous value of qj
27        ẋ_j^− = ẋ_j //store previous value of dxj/dt
28        h = MAX_BE_STEP_SIZE(x_i, x_j^aux) // maximum BE step
              size as defined in Eq.\eqref{eq:hmax}
29        [q_i, q_j] = BE_step(x_i, x_j^aux, h) // qi and qj are computed
              using a BE step size h from xi and xj
30        t_j^q = t // last qj update
31        t_j^η = min(τ > t) subject to x_j(τ) = q_j or
              |q_j − x_j(τ)| = 2ΔQ_j // compute the time of the
              next change in the j−th quantized state
32        for each k ∈ [1,n] such that ẋ_k depends on q_j
33          e = t − t_k^x // elapsed time since last xk update
34          x_k = x_k + ẋ_k · e // update k−th state value
35          ẋ_k^− = ẋ_k //store previous value of dxk/dt
```

```
36              $\dot{x}_k = f_k(\mathbf{q}, t)$ // recompute k-th state derivative
37              $t_k^\eta = \mathbf{min}(\tau > t)$ subject to $x_k(\tau) = q_k$ or
                    $|q_k - x_k(\tau)| = 2\Delta Q_k$ // recompute the time of the
                      next change in the k-th quantized state
38              $A_{k,j} = (\dot{x}_k - \dot{x}_k^-)/(q_j - q_j^-)$ // Jacobian
39              if $k \neq j$ then $t_k^x = t$ // last xk update
40            end for
41            $A_{j,j} = (\dot{x}_j - \dot{x}_j^-)/(q_j - q_j^-)$ // Jacobian diagonal entry

42            $u_{j,j} = \dot{x}_j(t) - A_{j,j} \cdot q_j$ // affine coefficient
43          end if
44        end if
45      end for
46      for each $j \in [1, n]$ such that $\dot{x}_j$ depends on $q_i$
47        $e = t - t_j^x$ // elapsed time since last xj update
48        $x_j = x_j + \dot{x}_j \cdot e$ // update j-th state value
49        $\dot{x}_j^- = \dot{x}_j$ //store previous value of dxj/dt
50        $\dot{x}_j = f_j(\mathbf{q}, t)$ // recompute j-th state derivative
51        $t_j^\eta = \mathbf{min}(\tau > t)$ subject to $x_j(\tau) = q_j$ or $|q_j - x_j(\tau)| = 2\Delta Q_j$
              // recompute the time of the next change in the j-
              th quantized state
52        $A_{j,i} = (\dot{x}_j - \dot{x}_j^-)/(q_i - q_i^-)$ // Jacobian
53          if $j \neq i$ then $t_j^x = t$ // last xj update
54      end for
55      // update linear approximation coefficients
56      $A_{i,i} = (\dot{x}_i - \dot{x}_i^-)/(q_i - q_i^-)$ // Jacobian diagonal entry
57      $u_{i,i} = \dot{x}_i(t) - A_{i,i} \cdot q_i$ // affine coefficient
58      $t_i^x = t$ // last xi update
59  end while
```

Nótese que este nuevo algoritmo agrega el cálculo de pasos en simultaneo en los estados x_i y x_j (lineas 28–29), pero estos sólo tienen lugar ante la ocurrencia de oscilaciones. De otro modo, al algoritmo únicamente se adicionan los cálculos necesarios para detectar cambios de signo en las derivadas de los estados, lo cual requiere estimarlas (lineas 19 y 23) y estimar asimismo la amtriz Jacobiana completa (lineas 38, 41, 52 y 56) y diferentes coeficientes afines.

Volvamos entonces al ejemplo previo, pero simulando ahora el sistema de las Ecuaciones (4.1) con el algoritmo mLIQSS1 a partir de las mismas condiciones iniciales.

Como fue previamente descripto, el algoritmo se comporta de manera idéntica a LIQSS1 hasta el punto en el que se detecta la condición de oscilaciones. Esta sitaución tiene lugar en tiempo t_2, cuando actualizar la variable

q_2 provocaría que $\dot{x}_1$ vaya de $\dot{x}_2(t_2^-) = 0$ a $\dot{x}_j(t_2^+) = -1$ (como ocurre con LIQSS1).

Aquí, el algoritmo modificado detecta esta situación y verifica si un subsiguiente cambio en q_1 produce un cambio en el signo de $\dot{x}_2$. Tras proponer $q_1(t_2^+) = x_1 - \Delta Q$, resulta que la derivada de el estado $\dot{x}_2(t_2^+) = -2$ cambia de signo. Así, se predicen futuras oscilaciones entre ambas variables de estado, y se computa un paso de *Backward Euler* simultaneamente en ambas variables. Siendo que los estaos se encuentran próximos a un punto de equilibrio, este paso se computa hasta el tiempo final t_f y los estados cuantificados tomas los valores de equilibrio $q_1 = -0{,}5$ − $q_2 = 0{,}7$. Con estos valores, las derivadas de ambos estados se anulan, y así la simulación termina sin ninguna oscilación, como puede verse en la Fig. 4.2.

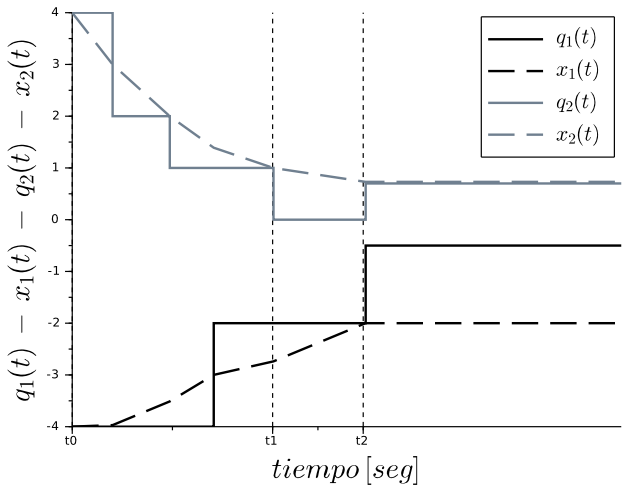


Figura 4.2: Simulación del sistema (4.1) con el algoritmo mLIQSS1.

4.4. Ejemplos y resultados

Esta sección se muestran resultados de simulación, comparando el desempeño del algoritmo modificado con su versión anterior, así como con métodos clásicos(DOPRI and DASSL).

A los efectos de realizar estas comparaciones, se corrieron una serie de experimentos en diferentes modelos bajo las siguientes condiciones:

- Las simulaciones fueron realizadas con una PC AMD A4-3300 APU@2,5GHz Bajo el sistema operativo Ubuntu.
- En todos los casos, medimos el tiempo de CPU, el número de evaluaciones de funciones escalares y el error relativo, calculado como:

$$e_{rr} = \sqrt{\frac{\sum (x[k] - x_{REF}[k])^2}{\sum x_{REF}[k]^2}} \tag{4.11}$$

donde la solución de referencia $x_{REF}[k]$ fue obtenida utilizado el algoritmo DASSL con una tolerancia de error muy pequeña (10^{-9}).

4.4.1. Convertidor Ćuk intercalado

La simulación de convertidores de electrónica conmutada de potencia es un caso donde los métodos LIQSS presentan importantes ventajas por sobre los métodos clásicos. En ellos, el tratamiento eficiente de las discontinuidades y la rigidez son las principales razones de esas ventajas [32]. Sin embargo, como fue reportado en la cita referenciada, cuando aparecen términos grandes en la matriz Jacobiana, a ambos lados de la diagonal principal, los métodos LIQSS fallan debido a la aprición de rápidas oscilaciones entre pares de variables. Un ejemplo donde esto ocurre en en el convertidor Ćuk.

Para verificar que el algoritmo modificado de LIQSS resuelve este inconveniente, simulamos el circuito correspondiente a un convertidor Ćuk intercalado de cuatro etapas, como el mostrado en la Fig. 4.3.

Las ecuaciones de estado de la j–ésima etapa del convertidor de potencia pueden ser escritas como sigue:

$$\begin{aligned}
\frac{\mathrm{d}i_{L_1^j}}{\mathrm{d}t} &= \frac{U - u_{C_1^j} - i_{D^j} \cdot R_{S^j}}{L_1} \\
\frac{\mathrm{d}i_{L_2^j}}{\mathrm{d}t} &= \frac{-u_{C_2} - i_{D^j} \cdot R_{S^j}}{L_2} \\
\frac{\mathrm{d}u_{C_1^j}}{\mathrm{d}t} &= \frac{i_{D^j} - i_{L_2^j}}{C_1}
\end{aligned}$$

con

$$i_{D^j} = \frac{(i_{L_1^j} + i_{L_2^j}) \cdot R_{S^j} - u_{C_1^j}}{R_{S^j} + R_{D^j}}$$

donde R_{S^j} y R_{D^j} son las resistencias de los interruptores y diodos respectivamente, las cuales pueden tomar uno de dos valores posibles, R_{ON} o R_{OFF}

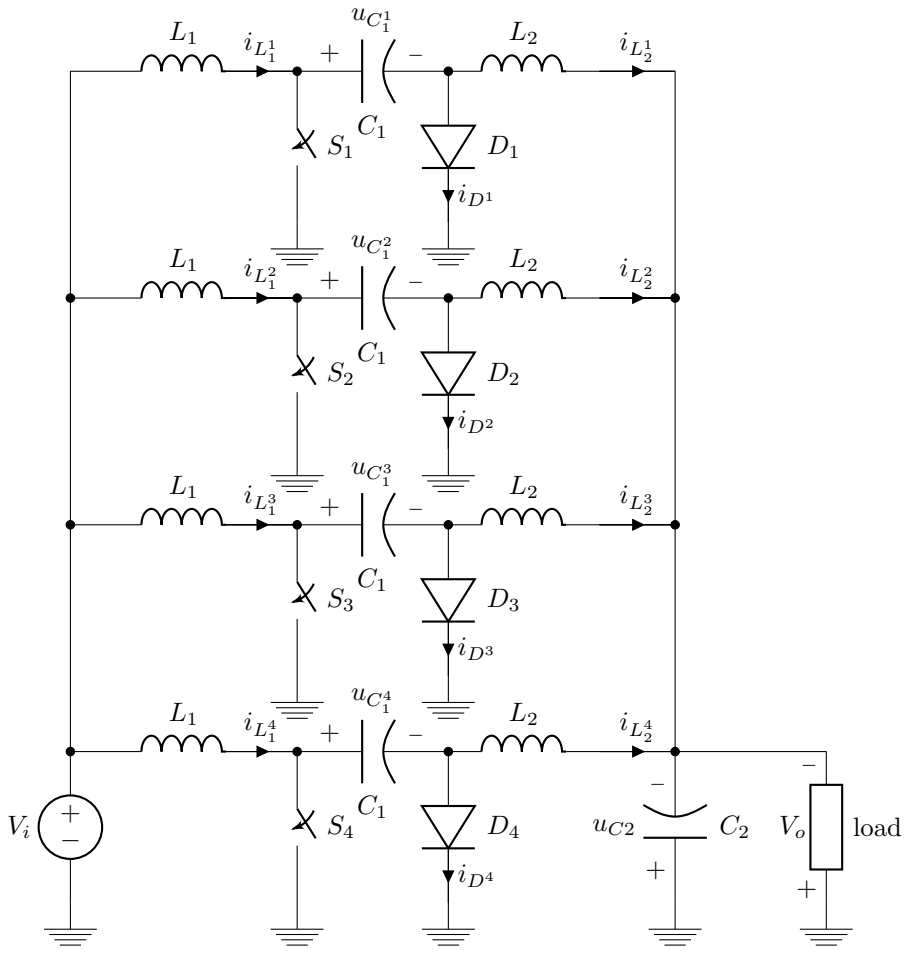


Figura 4.3: Convertidor Ćuk intercalado de cuatro etapas.

dependiendo de el estado en el cual se encuentren. Finalmente, la tensión de salida obedece la siguiente ecuación:

$$\frac{\mathrm{d}u_{C_2}}{\mathrm{d}t} = \frac{\sum_{j=1}^{N} i_{L_2^j} - \frac{u_{C_2}}{R_o}}{C_2}$$

El modelo fue simulado con el siguiente conjunto de parámetros:

- Tensión de entrada: $U = 24V$

- Capacitores: $C_1 = 10^{-4}F$ y $C_2 = 10^{-4}F$
- Inductores: $L_1 = 10^{-4}Hy$ y $L_2 = 10^{-4}Hy$
- Resistencia de carga: $R_o = 10\Omega$
- Resistencia de encendido (interruptor y diodo): $R_{ON} = 10^{-5}\Omega$
- Resistencia de apagado (interruptor y diodo): $R_{OFF} = 10^{5}\Omega$
- Período de la senñal de control: $T = 10^{-4}sec$
- Ciclo de trabajo de la señal de control: $DC = 0{,}25$

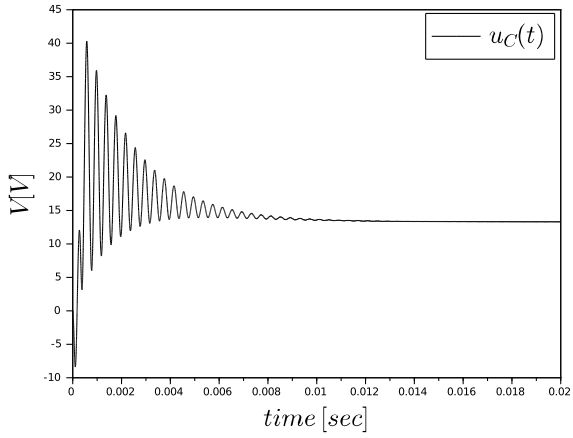
El sistema fue simulado hasta el tiempo final $t_f = 0{,}02sec$, y los resultados se muestran en la Fig.4.4.

La comparación entre los desempeños de los diferentes algoritmos está reportada en la Tabla 4.2 para dos valores de tolerancias de error. Los resultados de la simulación con el algoritmo DOPRI no han sido reportados, puesto que el sistema es demasiado *stiff* y dicho algoritmo ha fallado en proveer resultados en un tiempo de CPU razonable.

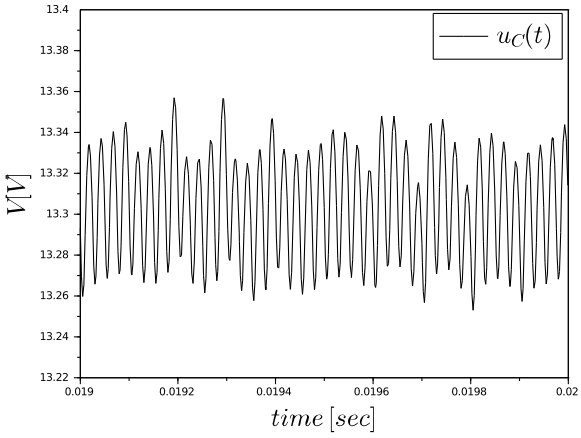
Método de Integración		**Error Relativo**	**Evaluaciones de función f_i**	**CPU [mseg]**
DASSL	$tol = 1 \cdot 10^{-1}$	$1{,}7 \cdot 10^{-1}$	4,928,846	227
	$tol = 1 \cdot 10^{-2}$	$7{,}5 \cdot 10^{-2}$	4,402,996	218
LIQSS1	$\Delta Q_i = 1 \cdot 10^{-1}$	$3{,}0 \cdot 10^{-2}$	80,243,117	9,697
	$\Delta Q_i = 1 \cdot 10^{-2}$	$2{,}8 \cdot 10^{-3}$	102,413,128	12,399
mLIQSS1	$\Delta Q_i = 1 \cdot 10^{-1}$	$4{,}4 \cdot 10^{-2}$	1,043,458	163
	$\Delta Q_i = 1 \cdot 10^{-2}$	$5{,}2 \cdot 10^{-3}$	5,123,830	676

Tabla 4.2: Comparación de resultados para convertidor Ćukl de cuatro etapas.

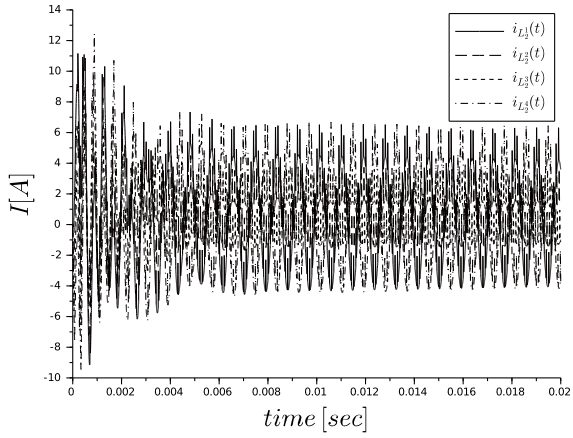
Lo primero que se observa es que el algoritmo LIQSS1 original realiza una cantidad enorme de pasos, que tienen lugar producto de la aparición de oscilaciones espurias entre las variables de estados que computan las corrientes por los inductores, como fue oportunamente observado en [32]. Esta claro que el algoritmo modificado no sufre de este inconveniente y realiza entre 20 y 80 veces menos cantidad de pasos.



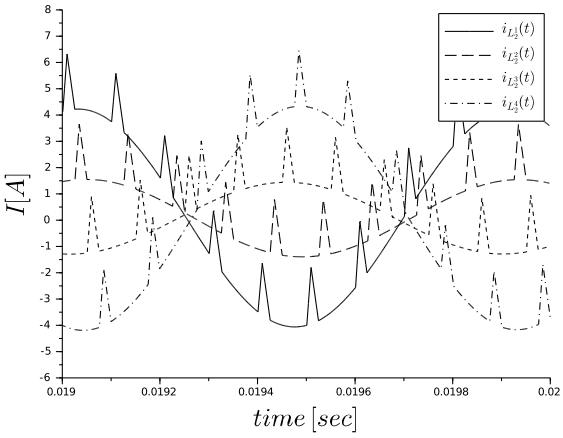
(a) Tensión de salida.



(b) Detalle de la tensión de salida.



(c) Corrientes $i_{L_2^j}$.



(d) Detalle de las corrientes $i_{L_2^j}$.

Figura 4.4: Simulación convertidor Ćuk intercalado de cuatro etapas.

Para una configuración de baja presición (tolerancia = 10^{-1}), LIQSS1 modificado supera los resultados obtenidos con DASSL. Sin embargo cuando se establece una tolerancia de 10^{-2}, DASSL resulta más rápido. Esto puede ser fácilmente explicado debido al hecho de que LIQSS1 es un método sólo de orde uno, y no puede obtener buenas precisiones sin incrementar significativamente el número de pasos. De cualquier modo, puede verse que para

iguales tolerancias, mLIQSS1 es mucho más preciso que DASSL.

Las ventajas de mLIQSS1 con respecto a DASSL se deben al manejo eficiente de las discontinuidades y a la explotación de la dispersión del sistema, así como al tratamiento explícito de la rigidez (sólo se invierten matrices de 2 por 2 independientemente del tamaño n del sistema, mientras que DASSL necesita invertir una matriz de n por n en cada paso).

Estas ventajas deberían ser más notorias a medida que el tamaño del sistema se incrementa. Para verificar este hecho, se ha simulado este modelo variando el tamaño del sistema desde 4 a 32 etapas. En cada no de esos experimentos, se estableció una tolerancia para cada cada método tal que el error medido sea el mismo. De este modo, se compara el tiempo de CPU requerido por cada método para obtener resultados con idénticos errores.

Los resutlados obtenidos se muestran en la Figura. 5.2. Como era de esperarse, las ventajas de mLIQSS1 crecen con el número de etapas, resultando aproximadamente de 10 veces más rápido que DASSL para el caso de 32 etapas.

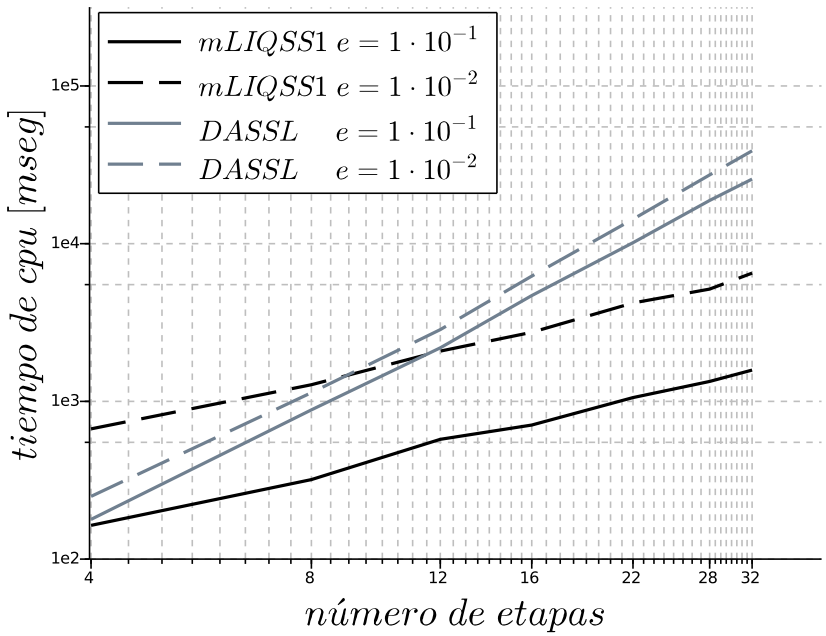


Figura 4.5: Comparación de tiempos de simulación: Convertido Ćuk intercalado para tolerancias de $1 \cdot 10^{-3}$ y $1 \cdot 10^{-5}$.

4.4.2. Modelo de Tyson: intreacciones entre Cdc2 y cyclin

Otra aplicación donde los métodos LIQSS presentan ventajas respecto a lso métodos clásicos de tiempo discreto son algunos modelos biológicos [33]. Sin embargo, en algunos de estos modelos la matriz Jacobiana contiene términos grandes a ambos lados de la diagonal principal, lo que provoca oscilaciones espurias en la famial de métodos LIQSS. Un caso donde esto sucede es en el clásico modelo de Tyson de *interacción entre cdc2 y ciclina*, presentado en [34], que representa la creación y degradación de ciclina en una célula.

Para verificar que el algoritmo mLIQSS1 funciona en este caso, se ha considerado un modelo compuesto por 100 células partiendo de diferentes condiciones iniciales.

Las ecuaciones de una célula son las siguientes:

$$\begin{aligned}
\frac{\mathrm{d}C2}{\mathrm{d}t} &= k_6 \cdot M - k_8 \cdot [P] \cdot C2 + k_9 \cdot CP \\
\frac{\mathrm{d}CP}{\mathrm{d}t} &= -k_3 \cdot CP \cdot Y + k_8 \cdot [P] \cdot C2 - k_9 \cdot CP \\
\frac{\mathrm{d}pM}{\mathrm{d}t} &= k_3 \cdot CP \cdot Y - pM \cdot F(M) + k_5 \cdot [P] \cdot M \\
\frac{\mathrm{d}M}{\mathrm{d}t} &= pM \cdot F(M) - k_5 \cdot [P] \cdot M - k_6 \cdot M \\
\frac{\mathrm{d}Y}{\mathrm{d}t} &= k1 \cdot [aa] - k_2 \cdot Y - k_3 \cdot CP \cdot Y \\
\frac{\mathrm{d}YP}{\mathrm{d}t} &= k_6 \cdot M - k_7 \cdot YP
\end{aligned}$$

donde las concentraciones de $[P]$ y de los aminoácidos $[aa]$ se asume son constantes. Hay seis variables de estado: las concentraciones de cc2 ($C2$), cdc2-p (CP), preMPF = P-ciclina-cdc2-P (pM), MPF activo = P-ciclina-cdc2 (M), ciclina (Y) y ciclina-P (YP). $F(M)$ es una función que describe la retroalimentación autocatalítica del MPF activo en su propia producción. Las definiciones de las constantes $k_{1,\ldots,9}$, así como explicaciones en mayor detalle pueden se encontradas en [34].

El sistema fue simulado con tolerancias de 10^{-3} y 10^{-5}, hasta un tiempo final $t_f = 50$ usando diferentes algoritmos. Los resultados usando DOPRI no fueron reportados debido a que no pudo completarse la simulación con este método como consecuencia de la rigidez del sistema.

Los resultados son reportados en la Table 5.3. Las salidas de esta simulación pueden verse en la Figura 4.6.

A partir de la Tabla 5.3, podemos ver que el algoritmo modificado mLIQSS1 es significativamente más rápido que su versión original. Dado que mLIQSS1 es un método de primer orden, sólo se obtienen resultados decentes para tolerancias de error grandes. De cualquier modo, utilizando una tolerancia de error estándar de 10^{-3}, se obtienen resultados comparables a los obtenidos con DASSL.

4.5. Conclusiones

En este capítulo se presentó el método QSS linealmente implícito modificado de primer orden, mLIQSS1, que surge de combinar de combinar el método LIQSS1 con pasos simultáneos en pares de variables usando el méto-

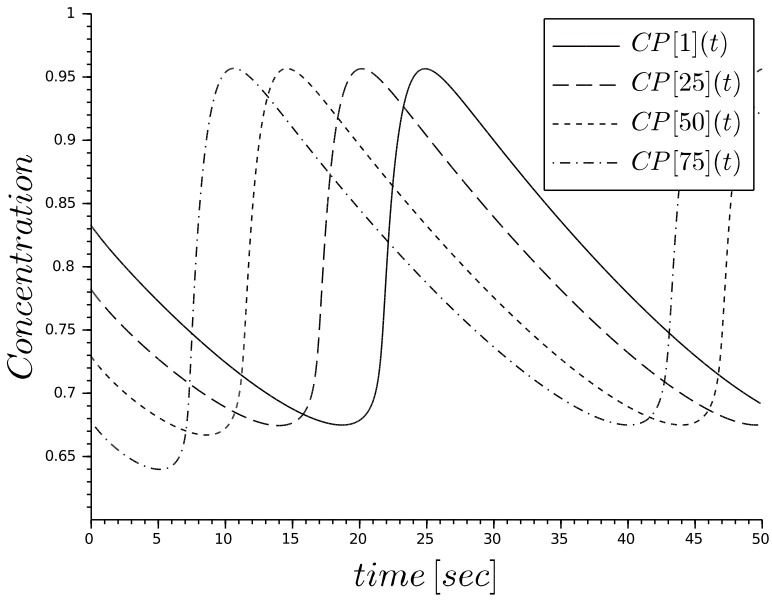


Figura 4.6: Resultado de la simulación del modelo de Tyson.

Método de Integración		**Error Relativo**	**Evaluaciones de función f_i**	**CPU [mseg]**
DASSL	$tol = 1 \cdot 10^{-3}$	$3{,}34 \cdot 10^{-}3$	$44,062,800$	$1,648$
	$tol = 1 \cdot 10^{-5}$	$5{,}18 \cdot 10^{-}4$	$28,478,400$	$2,257$
LIQSS1	$\Delta Q_i = 1 \cdot 10^{-3}$	$7{,}44 \cdot 10^{-}3$	$43,127,543$	$5,704$
	$\Delta Q_i = 1 \cdot 10^{-5}$	$2{,}11 \cdot 10^{-}5$	$134,002,162$	$20,741$
mLIQSS1	$\Delta Q_i = 1 \cdot 10^{-3}$	$7{,}10 \cdot 10^{-}3$	$8,672,317$	$1,704$
	$\Delta Q_i = 1 \cdot 10^{-5}$	$1{,}00 \cdot 10^{-}5$	$56,204,686$	$11,112$

Tabla 4.3: Comparación resultados de la simulación del modelo de Tyson para 100 células.

do Backward Euler. Este método, comparte las propiedades de precisión de los métodos QSS anteriores.

Como fue explicado, los métodos LIQSS pueden integrar eficientemente sistemas donde la rigidez se deba a la presencia de términos grandes en la diagonal principal de la matriz Jacobiana, pero si esta condición estructural no se cumple, los mismos resultan ineficientes. Esta ineficiencia se traduce en la presencia de oscilaciones espurias rápidas producidas por la interacción de variables de estado.

El método mLIQSS1 logra integrar eficientemente estructuras más generales. Particularmente, tiene la capacidad de detectar situaciones en las que potencialmente se producirían oscilaciones entre pares de variables de estados y realizar un paso simultaneo en ambas variables. Es decir, mLIQSS1 puede resolver subsistemas de 2×2 dando un paso utilizando un método clásico implícito en caso de prever posibles oscilaciones entre pares de variables.

A pesar de sus virtudes este método padece de dos grandes falencias. La primera de ellas reside en que sólo realiza una aproximación de primer orden, tanto para los pasos con LIQSS1 como para los pasos simultáneos utilizando Backward Euler, por lo cual está muy limitado en precisión. Y la segunda es que sólo es capaz de resolver la aparición de oscilaciones espurias debidas a la interacción entre pares de variables, pero si las mismas tienen lugar debido a la presencia de más de dos variables, el método resulta ineficiente.

Con la idea de solucionar el primer inconvenientes mencionado, en el próximo capítulo se introduce un método de segundo orden que combina el método LIQSS2 con la idea planteada en el presente capítulo.

Capítulo 5

Método LIQSS modificado de segundo orden

5.1. Versión mejorada de LIQSS2

Antes de presentar el algoritmo LIQSS2 modificado, se expondrá un cambio en la formulación del algoritmo LIQSS2 de [26]. Este cambio, que es necesario para definir el algoritmo mLIQSS2, mejora además el desempeño de LIQSS2 en casos simples donde las oscilaciones espurias no aparecen.

La definición original de LIQSS2 combina las ideas de QSS2 con LIQSS1, donde los estado cuantificados $q_i(t)$ siguen trayectorias seccionalmente lineales, tales que al final de cada segmento alcancen los estados, es decir, $q_i(t+h) = x_i(t+h)$. Sin embargo, la pendiente de los estados cuantificados eran elegidas talees que coincidan con las derivadas de los estados al comienzo del paso, es decir, $\dot{q}_i(t) = \dot{x}_i(t)$.

A los efectos de extender la idea mLIQSS1, necesitamos reformular LIQSS2 tal que la pendiente de los estados cuantificados y la derivadas de los estados coincidas an final del paso, es decir, $\dot{q}_i(t+h) = \dot{x}_i(t+h)$.

Para lograrlo, q_i y $\dot{q}_i$ son computados tales que verifiquen las siguientes ecuaciones:

$$\begin{aligned} \dot{q}_i &= \dot{x}_i + h \cdot \ddot{x}_i = A_{i,i} \cdot q_i + u_{i,i} + h \cdot (A_{i,i} \cdot \dot{q}_i + \dot{u}_{i,i}) \\ q_i + h \cdot \dot{q}_i &= x_i + h \cdot \dot{x}_i + \frac{h^2}{2} \cdot \ddot{x}_i \\ &= x_i + h \cdot (A_{i,i} \cdot q_i + u_{i,i}) + \frac{h^2}{2} \cdot (A_{i,i} \cdot \dot{q}_i + \dot{u}_{i,i}) \end{aligned} \tag{5.1}$$

donde $\dot{u}_{i,i}$ es el coeficiente afín de la pendiente. Nótese que la primera ecua-

ción dice que la pendiente del estado cuantificado, $\dot{q}_i$, es igual a la derivada del estado,$\dot{x}_i$, en tiempo $t+h$. Luego, la segunda ecuación dice que el estado cuantificado q_i es igual al estado x_i en tiempo $t+h$.

Aquí, h es calculado como el máximo paso en la Ec. .(5.1) tal que $|q_i - x_i| \leq \Delta Q_i$. Similarmente a mLIQSS1, este valor de h puede ser hallado analíticamente (es decir, resolviendo la Ec. (5.1) para h usando $q_i = x_i \pm \Delta Q_i$) o numéricamente.

Así, el algoritmo de simulación LIQSS2 funciona como sigue:

Algoritmo 5.1: LIQSS2.

while $(t < t_f)$ // simulate until final time tf
  $t = \mathbf{min}(t_j^\eta)$ // adavance simulation time
  $i = \mathbf{argmin}(t_j^\eta)$ // the i-th quantized state changes first
  $e = t - t_i^x$ // elapsed time since last xi update
  // update i-th state value and its derivative
  $x_i = x_i + \dot{x}_i \cdot e + 0{,}5 \cdot \ddot{x}_i \cdot e^2$
  $\dot{x}_i = \dot{x}_i + \ddot{x}_i \cdot e$
  $\dot{x}_i^- = \dot{x}_i$ //store previous value of dxi/dt
  $u_{ii} = u_{ii} + e_{xi} \cdot \dot{u}_{ii}$ // affine coefficient projection
  $e = t - t_i^q$ // elapsed time since last qi update
  $q_i^- = q_i + e \cdot \dot{q}_i$ //store previous value of qi projected
  $h = \text{MAX\_2ND\_ORDER\_STEP\_SIZE}(x_i)$
  $[q_i, \dot{q}_i] = \text{2ND\_ORDER\_step}(x_i, h)$
  $t_i^q = t$ // last qi update
  $t_i^\eta = \mathbf{min}(\tau > t)$ subject to $x_i(\tau) = q_i(\tau)$// compute the time of then next change in the i-th quantized state
  for each $j \in [1, n]$ such that $\dot{x}_j$ depends on q_i
    $e = t - t_j^x$ // elapsed time since last xj update
    // update j-th state value and its derivatives
    $x_j = x_j + \dot{x}_j \cdot e + 0{,}5 \cdot \ddot{x}_j \cdot e^2$
    $\dot{x}_j = f_j(\mathbf{q}(t), t)$ // recompute state derivative
    $\ddot{x}_j = \dot{f}_j(\mathbf{q}(t), t)$ // recompute state second derivative
    $t_j^\eta = \mathbf{min}(\tau > t)$ subject to $x_j(\tau) = q_j(\tau)$ or $|q_j(\tau) - x_j(\tau)| = 2\Delta Q_j$ // recompute the time of the next change in the j-th quantized state
    if $j \neq i$ **then** $t_j^x = t$ // last xj update
  end for
  // update linear approximation coefficients
  $A_{i,i} = (\dot{x}_i - \dot{x}_i^-) / (q_i - q_i^-)$ // Jacobian diagonal entry
  $u_{i,i} = \dot{x}_i - A_{i,i} \cdot q_i$ // affine coefficient
  $\dot{u}_{i,i} = \ddot{x}_i - A_{i,i} \cdot \dot{q}_i$ // affine coefficient slope
  $t_i^x = t$ // last xi update
end while

Aquí, se puede observar que los pasos de LIQSS2 agregan algunos cálculos con respecto a los de QSS2. Se calcula el máximo paso de h de orden dos (linea 12) y se lo utiliza para computar el estado cuantificado y su derivada (linea 13). Además se calculan los elementos de la diagonal principal de la matriz Jacobiana $A_{i,i}$ y los parámetros afines $u_{i,i}$ y $\dot{u}_{i,i}$ (lineas 26–28).

Del mismo modo que en LIQSS1, el algoritmo verifica una condición para asegurar que la diferencia entre x_j y q_j se mantenga acotada (por $2\Delta Q_j$) y controla además si un cambio en el valor del estado cuantificado q_i causa que x_j deje de aproximarse a q_j (linea 22).

5.2. LIQSS2 modificado(mLIQSS2)

El algoritmo LIQSS modificado combina las ideas de los métodos LIQSS2 y mLIQSS1. Esto es, el algoritmo funciona de manera idéntica a LIQSS2 hasta que se detecte una cadena de cambios significativos en pares de derivadas de variables de estado x_i y x_j[1]. Decimos que hay una cadena de cambios significativos cuando al actualizar el estado cuantificado q_i se produce un cambio abrupto en $\dot{x}_j$ que provoca un posterior cambio en q_j, el cual su vez producirá un cambio abrupto en $\dot{x}_i$.

Bajo estas cirucunstacias, un cambio simultaneo en los estados cuantificados q_i y q_j, así como en sus pendientes $\dot{q}_i$ y $\dot{q}_j$, es aplicado siguiento una fórmula implítica de segundo orden.

A los efectos de predecir las futuras derivadas primera de los estados, $\dot{x}_i$ y $\dot{x}_j$, se utiliza la aproximación lineal de la Ec. (4.4). Además, las futuras derivadas segundas de los estados, $\ddot{x}_i$ and $\ddot{x}_j$, son estimadas derivando la Ec. (4.4), obteniendo así el siguiente sistema de ecuaciones.

$$\begin{aligned} \ddot{x}_i &= A_{ii} \cdot \dot{q}_i + A_{ij} \cdot \dot{q}_j + \dot{u}_{ij} \\ \ddot{x}_j &= A_{ji} \cdot \dot{q}_j + A_{jj} \cdot \dot{q}_j + \dot{u}_{ji} \end{aligned} \tag{5.2}$$

donde aparecen dos nuevos coeficientes, $\dot{u}_{ij}$ y $\dot{u}_{ji}$, que corresponden a parámetros afines de las pendientes.

Cuando un nuevo valor en el estado cuantificado q_i no causa un cambio significativo en la derivada de nigún otro estado, el algoritmo funciona de

[1] En el algoritmo de segundo orden, no sólo se verifican cambios de signo, sino además cambios significativos en el valor de las derivadas de los estados, ya que estos pueden causar secuencias de pasos peuqeños. El motivo de esto, es que cuando el estado cuantificado q_i produce un cambio abrupto en la derivada de algún otro estado $\dot{x}_j$, esto provoca que x_j y q_j se alejen uno del otro, lo cual tendrá como consecuencia un nuevo paso en q_j. Si este nuevo paso también provoca un cambio significativo en $\dot{x}_i$, entonces tendrán lugar oscilaciones rápidas.

manera idéntica a LIQSS2. Sin embargo, cuando un nuevo valor de q_i produce un cambio significativo en $\dot{x}_j$ o en $\ddot{x}_j$, se propone un nuevo valor para q_j en la nueva dirección de x_j. Entonces, se verifica si con ese nuevo valor de q_j se producen cambios en $\dot{x}_i$ o $\ddot{x}_i$. Si no es así, el algoritmo continua del mismo modo que LIQSS2. En caso contrario, se anticipa la presencia de oscilaciones entre los estados x_i y x_j, por tal motivo se computan ambos estados cuantificados, q_i y q_j y sus correspondientes pendientes, usando un paso implícito de segundo orden según Ec.(4.4).

Usando las definiciones de las Ecs.(4.5)–(4.6), y definiendo además:

$$\dot{\mathbf{q}}_{ij} \triangleq \begin{bmatrix} \dot{q}_i \\ \dot{q}_j \end{bmatrix} \ , \ \ddot{\mathbf{x}}_{ij} \triangleq \begin{bmatrix} \ddot{x}_i \\ \ddot{x}_j \end{bmatrix} \ , \ \dot{\mathbf{u}}_{ij} \triangleq \begin{bmatrix} \dot{u}_{ij} \\ \dot{u}_{ji} \end{bmatrix}$$

el paso implícito está dado por las siguientes ecuaciones

$$\dot{\mathbf{q}}_{ij} = \dot{\mathbf{x}}_{ij} + h \cdot \ddot{\mathbf{x}}_{ij} = \mathbf{A}_{ij} \cdot \mathbf{q}_{ij} + \mathbf{u}_{ij} + h \cdot (\mathbf{A}_{ij} \cdot \dot{\mathbf{q}}_{ij} + \dot{\mathbf{u}}_{ij}) \tag{5.3a}$$

$$\begin{aligned} \mathbf{q}_{ij} + h \cdot \dot{\mathbf{q}}_{ij} &= \mathbf{x}_{ij} + h \cdot \dot{\mathbf{x}}_{ij} + \frac{h^2}{2} \cdot \ddot{\mathbf{x}}_{ij} \\ &= \mathbf{x}_{ij} + h \cdot (\mathbf{A}_{ij} \cdot \mathbf{q}_{ij} + \mathbf{u}_{ij}) + \frac{h^2}{2} \cdot (\mathbf{A}_{ij} \cdot \dot{\mathbf{q}}_{ij} + \dot{\mathbf{u}}_{ij}) \end{aligned} \tag{5.3b}$$

Nótese que la Ec. (5.3a) dice que las pendientes de ambos estados cuantificados $\dot{\mathbf{q}}_{ij}$ coinciden con la derivada primera de los correspondientes estados $\dot{\mathbf{x}}_{ij}$ en tiempo $t + h$. Luego, la Ec. (5.3b) dice que los estados cuantificados $\mathbf{q}_{ij}$ coinciden con los estados $\mathbf{x}_{ij}$ en tiempo $t + h$.

Aquí, h es calculado como el paso máximo en la Ec. (5.3) tal que $|q_i - x_i| \leq \Delta Q_i$ y $|q_j - x_j| \leq \Delta Q_j$. De modo semejante a lo tratado en mLIQSS1, esta valor para h puede ser hallado analíticamente (es decir, resolviendo las Ecs. (5.3) para h usando $q_i = x_i \pm \Delta Q_i$ y $q_j = x_j \pm \Delta Q_j$) o numéricamente.

Luego, el algoritmo resultante para mLIQSS2 se comporta de modo idéntico al de LIQSS2 hasta el punto en el que se actualiza un estado cuantificado, donde se debe verificar la posible ocurrencia de oscilaciones entre pares de variables. Así, luego de la linea 15 del Algoritmo 5.1, el método modificado continua como sigue:

Algoritmo 5.2: mLIQSS2.

```
16    for each j ∈ [1,n] such that (i≠j and A_ij · A_ji ≠ 0)
17      e = t − t_j^x // elapsed time since last xj update
```

18 $x_j^{aux} = x_j + \dot{x}_j \cdot e + 0{,}5 \cdot \ddot{x}_j \cdot e^2$ // store updated j-th state value on auxiliary variable
19 $u_{j,j} = u_{j,j} + e \cdot \dot{u}_{j,j}$ // affine coefficient projection
20 $e = t - t_j^q$ // elapsed time since last qj update
21 $q_j^- = q_j + e \cdot \dot{q}_j$ //store previous value of qj projected
22 $\dot{x}_j^- = \dot{x}_j$ //store previous value of dxj/dt
23 $u_{j,i} = u_{j,j} - A_{j,i} \cdot q_i^-$ // affine coefficient
24 $\dot{x}_j^+ = A_{j,i} \cdot q_i + A_{j,j} \cdot q_j^- + u_{j,i}$ // next j-th state der. est.
25 $\dot{u}_{j,i} = \dot{u}_{j,j} - A_{j,j} \cdot q_i^-$ // affine coefficient
26 $\ddot{x}_j^+ = A_{j,i} \cdot \dot{q}_i + A_{j,j} \cdot \dot{q}_j + \dot{u}_{j,i}$ // next j-th state 2nd der. est.
27 **if** $(|\dot{x}_j - \dot{x}_j^+| > |\dot{x}_j + \dot{x}_j^+|/2$ or $|\ddot{x}_j - \ddot{x}_j^+| > |\ddot{x}_j + \ddot{x}_j^+|/2)$ // update in qi => significant change in dxj/dt or ddxj/dt
28 $q_j^+ = x_j^{aux} - sign(\ddot{x}_j^+) \cdot \Delta Q_j$ // update qj in future xj's direction
29 $\dot{q}_j^+ = (A_{j,i} \cdot (q_i + h \cdot \dot{q}_i) + A_{j,j} \cdot q_j^+ + u_{j,i} + h \cdot \dot{u}_{j,i})/(1 - h \cdot A_{j,j})$
30 $u_{i,j} = u_{i,i} - A_{i,j} \cdot q_j^-$ // affine coefficient
31 $\dot{x}_i^+ = A_{i,i} \cdot q_i + A_{i,j} \cdot q_j^+ + u_{i,j}$ // next i-th state der. est.
32 $\dot{u}_{i,j} = \dot{u}_{i,i} - A_{i,j} \cdot \dot{q}_j^-$ // affine coefficient
33 $\ddot{x}_i^+ = A_{i,i} \cdot \dot{q}_i + A_{i,j} \cdot \dot{q}_j^+ + \dot{u}_{i,j}$ // next i-th state 2nd der. est.
34 **if** $(|\dot{x}_i - \dot{x}_i^+| > |\dot{x}_i + \dot{x}_i^+|/2$ or $|\ddot{x}_i - \ddot{x}_i^+| > |\ddot{x}_i + \ddot{x}_i^+|/2)$ // update in qj => significant change in dxi/dt or ddxi/dt
35 // presence of oscillations
36 $h =$ MAX_2ND_ORDER_STEP_SIZE(x_i, x_j^{aux})
37 $[q_i, \dot{q}_i, q_j, \dot{q}_j] =$ 2ND_ORDER_step(x_i, x_j^{aux}, h)
38 $t_j^q = t$ // last qj update
39 $t_j^\eta =$ **min**$(\tau > t)$ subject to $x_j(\tau) = q_j(\tau)$ or $|q_j(\tau) - x_j(\tau)| = 2\Delta Q_j$ // compute the time of the next change in the j-th quantized state
40 **for each** $k \in [1, n]$ such that $\dot{x}_k$ depends on q_j
41 $e = t - t_k^x$ // elapsed time since last xk update
42 // update k-th state value and its derivatives
43 $x_k = x_k + \dot{x}_k \cdot e + 0{,}5 \cdot \ddot{x}_k \cdot e^2$
44 $\dot{x}_k^- = \dot{x}_k$ //store previous value of dxk/dt
45 $\dot{x}_k = f_j(\mathbf{q}(t), t)$ // recompute state derivative
46 $\ddot{x}_k = \dot{f}_k(\mathbf{q}(t), t)$ // recompute state second derivative
47 $t_k^\eta =$ **min**$(\tau > t)$ subject to $x_k(\tau) = q_k(\tau)$ or $|q_k(\tau) - x_k(\tau)| = 2\Delta Q_j$ // recompute the time of the next change in the k-th quantized state
48 $A_{k,j} = (\dot{x}_k - \dot{x}_k^-)/(q_j - q_j^-)$ // Jacobian

```
49            if k ≠ j then t_k^x = t // last xk update
50          end for
51           // update linear approximation coefficient
52          A_{j,j} = (ẋ_j − ẋ_j^−)/(q_j − q_j^−) // Jacobian diagonal entry

53          u_{j,j} = ẋ_j − A_{j,j} · q_j // affine coefficient
54          t_j^x = t // last xj update
55        end if
56      end if
57    end for
58    for each j ∈ [1, n] such that ẋ_j depends on q_i
59      e = t − t_j^x // elapsed time since last xj update
60      // update j−th state value and its derivatives
61      x_j = x_j + ẋ_j · e + 0,5 · ẍ_j · e^2
62      ẋ_j^− = ẋ_j //store previous value of dxj/dt
63      ẋ_j = f_j(q(t), t) // recompute state derivative
64      ẍ_j = ḟ_j(q(t), t) // recompute state second derivative
65      t_j^η = min(τ > t) subject to x_j(τ) = q_j or |q_j − x_j(τ)| = 2ΔQ_j
            // recompute the time of the next change in the j−
            th quantized state
66      A_{j,i} = (ẋ_j − ẋ_j^−)/(q_i − q_i^−) // Jacobian
67      if j ≠ i then t_j^x = t // last xj update
68    end for
69    // update linear approximation coefficients
70    A_{i,i} = (ẋ_i − ẋ_i^−)/(q_i − q_i^−) // Jacobian diagonal entry
71    u_{i,i} = ẋ_i − A_{i,i} · q_i // affine coefficient
72    u̇_{i,i} = ẍ_i − A_{i,i} · q̇_i // affine coefficient slope
73    t_i^x = t // last xi update
74  end while
```

En comparación a LIQSS2, el algoritmo modificado añade el cálculo de pasos simultáneos en los estados x_i y x_j (lineas 36–37), pero estos sólo tienen lugar ante la predicción de oscilaciones. De otro modo, el algoritmo sólo añade alguno algunos cálculos para realizar la predicción de de cambios significativos en las derivadas de los estados [2], lo cual requiere estimarlas (lineas 24, 26, 31, y estimar además la matriz Jacobiana completa (lineas 48, 52, 66 and 70), así como calcular diferentes parámetros afines.

[2]Se predice un cambio significativo en la derivada de un estado bajo la condición $|\dot{x}_i - \dot{x}_i^+| > |\dot{x}_i + \dot{x}_i^+|/2$. Esta condición equivale a considerar un cambio de signo en las derivadas, o bien un cambio de al menos tres veces (más grande o más chico) con respecto al valor previo. Otra condición podría utilizarse, sin embargo esta establece un buen compromiso entre la correcta detección de cambios que causen oscilaciones y el no realizar actualizaciones simultaneas de pares de estados de manera innecesaria.

5.3. Propiedades de los métodos LIQSS modificados

Del mismo modo que los algoritmo originales, las versiones modificadas de LIQSS1 y LIQSS2 aseguran que la diferencia entre cada estado x_i y su correspondiente estado cuantificado q_i está acotada por $2 \cdot \Delta Q_i$. Así, los nuevos algoritmos proveen una solución analítica para la Ec. (3.7), que, definiendo

$$\Delta \mathbf{x}(t) \triangleq \mathbf{q}(t) - \mathbf{x}(t)$$

puede ser reescrita como

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t) + \Delta \mathbf{x}(t), t) \tag{5.4}$$

Nótese que el sistema original dado por Ec. (3.6), sólo difiere de la aproximación dada por Ec. (5.4) por la presencia de una *perturbación* acotada $\Delta \mathbf{x}(t)$, donde cada componente de esta perturbación está acotado por

$$|\Delta x_i(t)| \leq 2\Delta Q_i$$

Dado que las propiedades de los algoritmos originales de LIQSS1 y LIQSS2 están basadas en la presencia estas cotas, los algoritmos modificados satisfacen las mismas propiedades:

- Convergencia:
  Asumiendo que $\mathbf{f}(\mathbf{x}, t)$ es Lipzschits local en $\mathbf{x}$ y seccionalmente continua en t, entonces, la solución aprosimada de Ec. (5.4) tiende a la solución analítica de Ec. (3.6) cuando el quantum para todas las variables, ΔQ_i, tiende a cero [5].

- Estabilidad práctica:
  Asumiendo que la solución analítica de Ec. (3.6) es asintóticamente estable al rededor de un punto de equilibrio, el uso de un quantum lo suficientemente pequeño asegura que la solución de Ec. (5.4) termina dentro una región arbitrariamente pequeña al rededor del punto de equilibrio [5].

- Cota de error global:
  En el caso de un sistema lineal invariante en el tiempo (LTI, por sus siglas en inglés), cuando

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \tag{5.5}$$

tratándose $\mathbf{A}$ de una matriz Hurwitz(es decir, el sistema LTI es asintóticamente estable), el máximo error cometido en una simulación está acotado por

$$|\mathbf{e}(t)| \preceq |\mathbf{V}||\text{Real}(\mathbf{\Lambda})^{-1}\mathbf{\Lambda}||\mathbf{V}^{-1}|\mathbf{\Delta Q}$$

donde $\mathbf{\Lambda} = \mathbf{V}^{-1}\mathbf{A}\mathbf{V}$ es la descomposición canónica de Jordan de $\mathbf{A}$, $\mathbf{\Delta Q}$ es el vector de quantums, el símbolo '$|\cdot|$' computa el valor absoluto de una matriz o un vector componente a componente, y '$\preceq$' represente una desigualdad componente a componente [26].

5.4. Implementación de los métodos LIQSS modificados

Los algoritmos modificado fueron implementados en el Solver Autónomo QSS (Stand Alone QSS Solver) [31]. Con ese propósito, los pseudocódigos de los algoritmos 4.1 y 5.2 fueron codificados en lenguaje C plano como funciones del solver QSS. Los correspondientes códigos están disponibles en `https://sourceforge.net/projects/qssengine/`.

En dicha implementación , el valor de h es hallado por medio de iteraciones, dado que este mecanismo resulta más rápido que obtenerlo analíticamente. De cualquier modo, esas iteraciones no apuntan a encontrar el valor exacto , sino un valor los suficientemente grande para h tal que se verifiquen las desigualdades de Ec. .(4.9). Con ese propósito, primero se verifica si las desigualdades se cumplen para $h = t_f - t$, donde t_f es el tiempo final de simulación. Si no es así, se intenta con un paso de valor $h = \Delta Q_i/|\dot{x}_i|$ (este es el valor de paso que QSS1 calcularía). Si este tamaño para el pasno no verifica las desigualdades, entonces se reduce el paso h usando un aproximación lineal de la dependencia de $|q_i - x_i|$ en h. Si este valor falla nuevamente, el algoritmo realiza un paso de LIQSS1 en un única variable.

La versión actualizada de LIQSS2 y mLIQSS2 usan un enfoque similar.

5.5. Ejemplos y resultados

Esta sección se muestran resultados de simulación, comparando el desempeño del algoritmo modificado con su versión anterior, así como con métodos clásicos(DOPRI and DASSL).

A los efectos de realizar estas comparaciones, se corrieron una serie de experimentos en diferentes modelos bajo las siguientes condiciones:

- Las simulaciones fueron realizadas con una PC AMD A4-3300 APU@2,5GHz Bajo el sistema operativo Ubuntu.
- En todos los casos, medimos el tiempo de CPU, el número de evaluaciones de funciones escalares y el error relativo, calculado como:

$$e_{rr} = \sqrt{\frac{\sum (x[k] - x_{REF}[k])^2}{\sum x_{REF}[k]^2}} \tag{5.6}$$

donde la solución de referencia $x_{REF}[k]$ fue obtenida utilizado el algoritmo DASSL con una tolerancia de error muy pequeña (10^{-9}).

5.5.1. Problema Advección-Reacción-Difusión (ADR) 1D

Las ecuaciones de Advección-difusión proporcionan la base para describir fenómenos de transferencia de calor y masa, así como como los procesos de la mecánica continua, donde la cantidad física de interés podría ser la temperatura en la conducción de calor o la concentración de alguna sustancia química. En varias aplicaciones, estos fenómenos ocurren en presencia de reacciones químicas, lo que lleva a la ecuación ADR, un problema que se encuentra con frecuencia en muchas áreas de las ciencias ambientales, así como en la ingeniería mecánica.

Los problemas de ADR discretizados con el método de líneas conducen a grandes sistemas rígidos de EDOs donde el uso de métodos LIQSS ha mostrado importantes ventajas sobre los algoritmos de tiempo discreto clásicos [35].

El siguiente conjunto de EDOs, tomado de [35], corresponde a la discretización espacial de un problema de ADR 1D:

$$\frac{\mathrm{d}u_i}{\mathrm{d}t} = -a \cdot \frac{u_i - u_{i-1}}{\Delta x} + d \cdot \frac{u_{i+1} - 2 \cdot u_i + u_{i-1}}{\Delta x^2} + r \cdot (u_i^2 - u_i^3)$$

para $i = 1, \ldots, N-1$ y

$$\frac{\mathrm{d}u_N}{\mathrm{d}t} = -a \cdot \frac{u_N - u_{N-1}}{\Delta x} + d \cdot \frac{2 \cdot u_{N-1} - 2 \cdot u_N}{\Delta x^2} + r \cdot (u_N^2 - u_N^3)$$

donde $N = 1000$ es el número de puntos de la discretización, y $\Delta x = \frac{10}{N}$.

Se consideran lo parámetros $a = 1$, $d = 0{,}001$, $r = 1000$, y condiciones iniciales

$$u_i(x, t = 0) = \begin{cases} 1 & \text{si } i \in [1, N/5] \\ 0 & \text{cada otro caso} \end{cases}$$

Se ha simulado este sistema hasta el tiempo final $t_f = 10$ usando los solvers clásicos DASSL y DOPRI, así como las versiones original y modificada de LIQSS2. El resultado de estas simulaicones se muestra en la Fig. 5.1. En todos los casos, simulamos utilizando dos configuraciones de tolerancia diferentes. Los resultados están reportados en la Tabla 5.1.

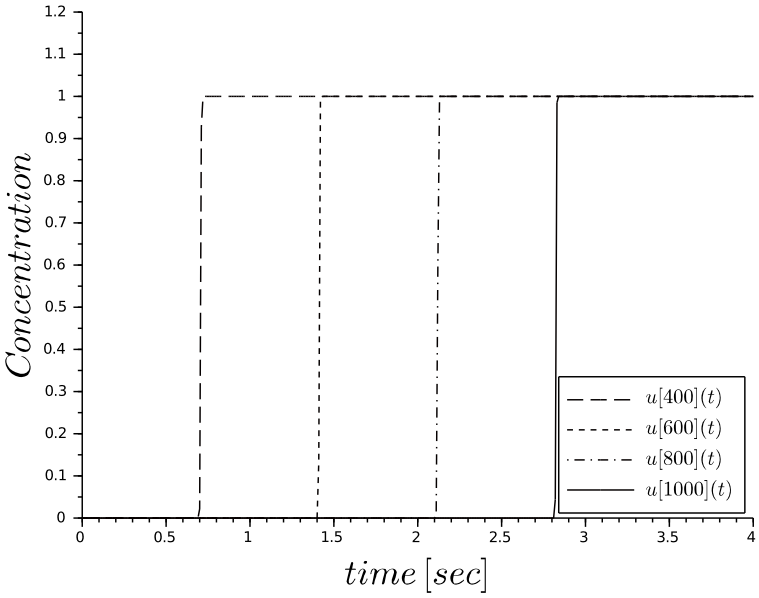


Figura 5.1: Reultado de la simulación del problema ADR-1D.

Como ya fue mencionado en [35], LIQSS2 supera los desempeños de DOPRI y DASSL. Sin embargo, la versión modificada de LIQSS2 presentada en este trabajo es más de dos veces más rápida que la original, extendiendo así las ventajas antes expuestas. Así, para la tolerancia *estándar* de 10^{-3}, mLIQSS2 es casi 50 veces más rápido que DOPRI, y al rededor de 2000 veces más rápido que DASSL. Al ajustar la tolerancia utilizada (al valor 10^{-5}) las ventajas se hacen menos notorias. Estos se debe a que mLIQSS2 es sólo de segundo orden, mientras que DASSL y DOPRO son algoritmos de quinto orden.

En este caso, la ventaja de mLIQSS2 por sobre LIQSS2 no se debe a la estructura de la matriz Jacobiana. El motivo es que mLIQSS2 no sólo utiliza el valor futuro del estado, sino también el valor futuro de la derivada del

Método de Integración		Error Relativo	Evaluaciones de función f_i	CPU [mseg]
DASSL	$tol = 1 \cdot 10^{-3}$	$8{,}28 \cdot 10^{-}2$	539,685	42,870
	$tol = 1 \cdot 10^{-5}$	$4{,}95 \cdot 10^{-}4$	35,186	31,237
DOPRI	$tol = 1 \cdot 10^{-3}$	$8{,}86 \cdot 10^{-}4$	36,184	1,032
	$tol = 1 \cdot 10^{-5}$	$1{,}69 \cdot 10^{-}4$	41,890	1,289
old LIQSS2	$\Delta Q_i = 1 \cdot 10^{-3}$	$1{,}59 \cdot 10^{-}3$	405,104	55
	$\Delta Q_i = 1 \cdot 10^{-5}$	$4{,}35 \cdot 10^{-}5$	2,764,382	362
mLIQSS2	$\Delta Q_i = 1 \cdot 10^{-3}$	$2{,}82 \cdot 10^{-}3$	140,812	22
	$\Delta Q_i = 1 \cdot 10^{-5}$	$1{,}98 \cdot 10^{-}5$	1,084,484	157

Tabla 5.1: Comparación de resultados para el problema ADR-1D.

estado para computar la trayectoria del estado cuantificado.

5.5.2. Convertidor Ćuk Interleaved

La comparación de rendimiento de los diferentes algoritmos se informa en la Tabla 5.2 para dos configuraciones de tolerancia diferentes. Los resultados de DOPRI no se informaron porque el sistema es demasiado rígido y no se obtuvieron resultados en un tiempo de CPU razonable.

Aquí podemos ver, como era de esperarse, que LIQSS2 tiene un desempeño bastante pobre. Sin embargo, mLIQSS2 es más rápido y más preciso que DASSL para ambas configuraciones de tolerancia. Más aún, para la tolerancia *estándar* de 10^{-3}, mLIQSS2 es más de cuatro veces más rápido y diez veces más preciso. Luego, para una tolerancia de 10^{-5} mLIQSS2 resulta aún más rápido y preciso, pero como en el ejemplo anterior, la diferencia es menos notable. Nuevamente, el motivo de esto es que la tolerancia de 10^{-5} no es apropiada para un algoritmo de segundo orden.

A los efectos de verificar como el costo computacional crece con la complejidad del circuito, se ha simulado el el modelo variando el tamaño de 4 a 32 etapas. A los efectos de realizar una comparación justa, se fijó la tolerancia en cada simulación de modo de obtener el mismo error en cada una de ellas. Los resultados pueden observarse en la Figura 5.2. Allí, puede verse que el tiempo de CPU de mLIQSS2 crece aproximadamente de modo lineal

	Método de Integración	Error Relativo	Evaluaciones de función f_i	CPU [mseg]
DASSL	$tol = 1 \cdot 10^{-3}$	$1{,}74 \cdot 10^{-}2$	$5,697,757$	285
	$tol = 1 \cdot 10^{-5}$	$1{,}52 \cdot 10^{-}4$	$7,056,634$	390
old LIQSS2	$\Delta Q_i = 1 \cdot 10^{-3}$	$1{,}96 \cdot 10^{-}3$	$216,086,632$	$33,747$
	$\Delta Q_i = 1 \cdot 10^{-5}$	$3{,}10 \cdot 10^{-}5$	$335,675,912$	$49,069$
mLIQSS2	$\Delta Q_i = 1 \cdot 10^{-3}$	$1{,}23 \cdot 10^{-}3$	$341,928$	60
	$\Delta Q_i = 1 \cdot 10^{-5}$	$1{,}64 \cdot 10^{-}5$	$2,021,402$	349

Tabla 5.2: Comparación de resultados para convertidor Ćukl de cuatro etapas.

con el número de etapas, mientras DASSL lo hace de forma cuadrática.

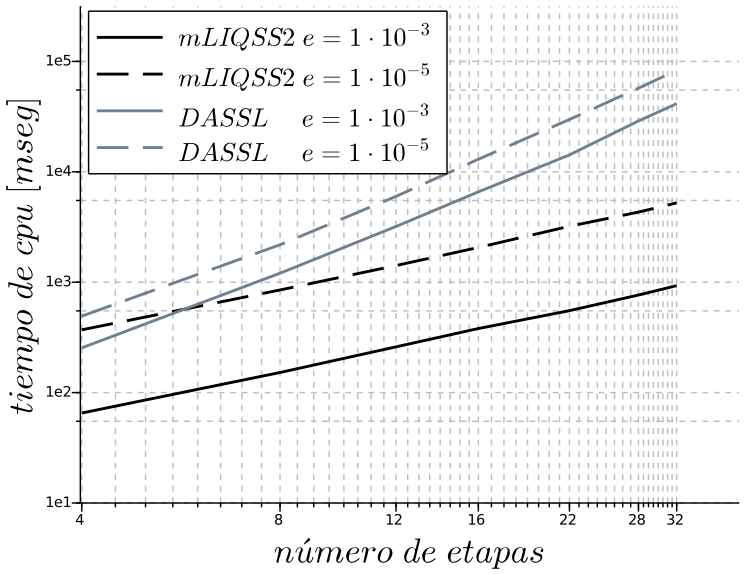


Figura 5.2: Comparación de tiempos de simulación: Convertido Ćuk intercalado para tolerancias de $1 \cdot 10^{-3}$ y $1 \cdot 10^{-5}$.

5.5.3. Modelo de Tyson: intreacciones entre Cdc2 y cyclin

Del mismo modo que en los caos anteriores, este sistema fue simulado para tolerancias de 10^{-3} y 10^{-5}, hasta un tiempo final $t_f = 50$ usando diferentes algoritmos de integración. Los resultados de DOPRI no están reportados, puesto que falló la simulación dada la rigidez del sistema.

Los resultados pueden encontrase en la Tabla 5.3.

Método de Integración		**Error Relativo**	**Evaluaciones de función f_i**	**CPU [mseg]**
mLIQSS1	$\Delta Q_i = 1 \cdot 10^{-3}$	$7{,}10 \cdot 10^{-}3$	$8,672,317$	$1,704$
	$\Delta Q_i = 1 \cdot 10^{-5}$	$1{,}00 \cdot 10^{-}5$	$56,204,686$	$11,112$
old LIQSS2	$\Delta Q_i = 1 \cdot 10^{-3}$	$6{,}17 \cdot 10^{-}3$	$55,720,144$	$9,945$
	$\Delta Q_i = 1 \cdot 10^{-5}$	$1{,}36 \cdot 10^{-}5$	$64,628,190$	$11,376$
mLIQSS2	$\Delta Q_i = 1 \cdot 10^{-3}$	$7{,}53 \cdot 10^{-}3$	$185,970$	42
	$\Delta Q_i = 1 \cdot 10^{-5}$	$1{,}21 \cdot 10^{-}5$	$1,471,838$	320

Tabla 5.3: Comparación resultados de la simulación del modelo de Tyson para 100 células.

En la Tabla 5.3, podemos observar que el algoritmo modificado mLIQSS2 es significativamente más veloz que su verisón original. Es más, para la tolerancia de error 10^{-3}, mLIQSS2 es más de 25 veces más rápido que DASSL obteniendo error semejantes. Luego, para la tolerancia de 10^{-5}, mLIQSS2 es también más rápido y más preciso. Como era de esperarse, el algoritmo LIQSS2 original resulta mucho menos eficiente que los demás.

5.6. Conclusiones

En este capítulo se presentó en primera instancia una mejora al método QSS linealmente implícito de segundo orden LIQSS2. Luego, teniendo esta mejora como base se presentó el método QSS linealmente implícito modificado de segundo orden, mLIQSS2, que surge de combinar el método LIQSS2 con pasos simultáneos en pares de variables usando un método implícito de segundo orden. Este método, comparte las propiedades de precisión de los métodos QSS anteriores.

El método mLIQSS2 logra integrar eficientemente estructuras más generales que el método original. Particularmente, tiene la capacidad de detectar situaciones en las que potencialmente se producirían oscilaciones entre pares de variables de estados y realizar un paso simultaneo en ambas variables. Es decir, mLIQSS2 puede resolver subsistemas de 2×2 dando un paso utilizando un método clásico implícito en caso de prever posibles oscilaciones entre pares de variables. Además, al ser un método de segundo orden, mejora notablemente los resultados obtenidos con el método de primer orden mLIQSS1.

Sin embargo, pese a los resultado obtenidos, este método sólo es capaz de resolver la aparición de oscilaciones espurias debidas a la interacción entre pares de variables, pero si las mismas tienen lugar debido a la presencia de más de dos variables, el método resulta ineficiente.

Con la idea de solucionar este inconvenientes mencionado, en el capítulo7 se introduce una familia de métodos que combinan los métodos QSS con los métodos clásicos.

A continuación, en el capítulo siguiente veremos algunos ejemplos de aplicación de los métodos mLIQSS en circuitos de electrónica conmutada en presencia de elementos parásitos, evaluando los resultados obtenidos y estableciendo sus posibles limitaciones.

Capítulo 6

Simulación de Circuitos de Electrónica Conmutada en Presencia de Elementos Parásitos

6.1. Introducción

Los circuitos de electrónica conmutada son utilizados para realizar conversiones de potencia de forma eficiente, y pueden ser encontrados en la gran mayoría de los dispositivos electrónicos de uso cotidiano e industrial. Estos circuitos utilizan uno o varios interruptores cuyos ciclos de trabajo son controlados para regular la tensión de salida deseada. A los efectos de minimizar el *ripple* en dicha salida, los elementos de conmutación deben ser operados a altas frecuencias (del orden los kHz).

La simulación de este tipo de circuitos no es trivial [32]. Por un lado, las altas frecuencias de conmutación implican discontinuidades frecuentes en los modelos a simular. Esto implica que los métodos de integración numérica deben detectar los momentos precisos donde dichas discontinuidades tienen lugar y reiniciar la simulación ante cada una de ellas, ya que no se puede integrar a través de una discontinuidad sin cometer un error inaceptable. Estos reinicios frecuentes son costosos computacionalmente y limitan el paso de integración utilizado. Por otro lado, el uso de modelos realistas para los elementos de conmutación conduce a modelos *stiff*, con dinámicas lentas y rápidas simultáneas. Los modelos con estas características deben integrarse con *métodos implícitos* ya que los explícitos resultarán inestables a menos

que se disminuya considerablemente el paso de integración [4]. Los métodos implícitos tienen un costo elevado ya que requieren iterar para resolver un sistema de ecuaciones en cada paso (costo que aumenta notablemente a medida que el tamaño del modelo se incrementa).

Como hemos visto en los capítulos anteriores, los métodos de integración denominados QSS que pueden resolver eficientemente sistemas con discontinuidades [4], y entre estos métodos encontramos los QSS Linealmente Implícitos (LIQSS) [26], que pueden además integrar muy eficientemente sistemas *stiff* sin necesidad de realizar iteraciones (siempre y cuando la rigidez de estos sistemas se deban a cierta estructura particular). Por este motivo, los LIQSS permiten simular sistemas de electrónica conmutada de manera muy eficiente siempre y cuando el circuito cuente con ciertas características [32], lo que se cumple en casi todas las topologías de las fuentes conmutadas (excepto en la fuente Ćuk). La versión modificada de estos métodos introducida en esta tesis, como hemos visto, permite integrar de manera eficiente topologías más generales (incluyendo el caso de la fuente Ćuk) [22, 23]. En todos los casos analizados hasta aquí, sin embargo, se omitió la presencia de elementos parásitos. Dado que los elementos dinámicos (inductancias y capacitores) cambian la estructura de las ecuaciones, es de esperar que la presencia de los mismos altere la condición requerida por los LIQSS y mLIQSS para funcionar eficientemente.

Este capítulo tiene como objetivo estudiar el desempeño de estos algoritmos cuando a los modelos de fuentes conmutadas se les agrega la presencia de ciertos elementos parásitos (inductancias en particular) y analizar si en este caso siguen ofreciendo ventajas frente a los métodos clásicos de integración.

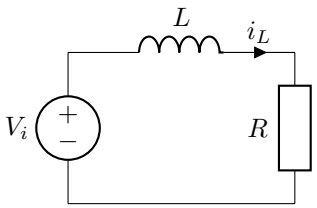
6.2. Rigidez en circuitos electrónicos

Un primer problema surge al simular sistemas usando el método QSS1, como vimos ejemplificado por el sistema dado por la Ec. 3.24. En la Figura3.7 se ve el resultado de dicha simulación, en donde pueden observarse oscilaciones espurias en torno al punto de equilibrio del sistema. Esto llevado al campo de la electrónica puede observarse en un circuito muy simple como el del inductor en serie con una resistencia grande ilustrado en la Figura 6.1a.

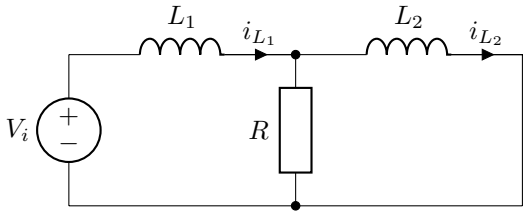
Al simular este circuito con un método QSS1, y recordando que los estados cuantificados sólo pueden tomar determinados valores, un pequeño cambio en el valor cuantificado de la corriente (como consecuencia de no poder arribar a un punto de equilibrio) producirá un gran cambio en la tensión

que cae en la resistencia. Este gran cambio en esa tensión producirá a su vez un cambio en la derivada de la corriente, dando lugar así a oscilaciones espurias rápidas, obteniendo un resultado como el mostrado en la Figura 3.7. Si en cambio, este sistema fuese simulado utilizando un método LIQSS1, el valor cuantificado de la corriente quedaría fijado en el punto de equilibrio, eliminando así las oscilaciones como vimos en la Figura 3.8.

Notar que en este caso, cuando la resistencia $R \to \infty$, se traduce en una restricción sobre una variable de estado $i_L(t) = 0$.



(a)



(b)

Figura 6.1: Circuitos RL.

El caso de las oscilaciones espurias entre dos variables, en tanto, aparece en el circuito de la Figura 6.1b donde nuevamente la resistencia R es grande. Aquí, un pequeño cambio en la versión cuantificada de la corriente por una inductancia se verá reflejado como un gran cambio en la derivada de la corriente por la otra inductancia, lo que al simularse mediante QSS se reflejará en oscilaciones espurias entre ambas variables. Este caso tampoco lo resolverá LIQSS, donde se obtendrán resultados similares a los mostrados en la Figura 4.1a, pero sí lo resolverá mLIQSS.

Notar que este segundo caso, cuando la resistencia $R \to \infty$, se traduce en una restricción sobre dos variables de estado: $i_{L_1} - i_{L_2} = 0$.

Veremos entonces algunas topologías circuitales correspondientes a fuentes conmutadas en las cuales se dan algunas de estas situaciones, observando el desempeño de los métodos LIQSS y mLIQSS. En particular, utilizaremos los métodos de segundo orden, lo cuales muestran el mejor desempeño a la hora de simular circuitos de electrónica conmutada.

6.3. Análisis sobre un Convertidor Buck

Un circuito de electrónica conmutada muy utilizado es el convertidor *Buck*. Se trata de un convertidor DC-DC, es decir, que tiene como entrada una determinada tensión continua (usualmente proveniente de la rectificación de una tensión alterna), y se obtiene a la salida una tensión continua de otro valor. Este circuito es también conocido como *reductor*, puesto que la tensión de salida resulta menor a la tensión de entrada. Podemos ver un diagrama de dicho circuito en la Figura 6.2, donde tanto el diodo D como la llave S serán modelados como una resistencia muy grande (R_{off}) cuando el elemento esté en condición de corte, y como una resistencia muy pequeña (R_{on}) cuando esté en conducción.

En este caso, en ausencia de elementos parásitos sabemos que los métodos de LIQSS funcionan eficientemente (y notablemente mejor que los métodos clásicos [32]). Esto puede verse topológicamente en el circuito ya que en la única situación que habría una restricción sobre variables de estado es cuando el diodo y la llave están abiertos simultáneamente (con $R_{\text{off}} \to \infty$), pero esta restricción se dará sobre una sola variable (i_L).

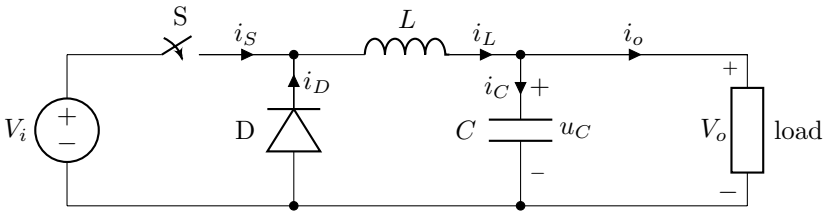


Figura 6.2: Circuito convertidor Buck.

Estudiaremos a continuación los resultados de simular dicho circuito en presencia de distintos elementos parásitos. Para ello, se mostrarán los resultados de simular el mismo circuito tanto con métodos clásicos (DASSL[36], CVODE[37] e IDA[38]) como con un método QSS linealmente implícito

(LIQSS). En este caso utilizaremos el método LIQSS2 que es de segundo orden y que es con el cual, dentro de la familia de métodos QSS, se obtienen los mejores resultados a la hora de simular este tipo de circuitos. Otros resultados de simular distintas topologías pueden encontrarse en [32, 23].

Todas las simulaciones se realizarán con los siguientes parámetros:

- $U = 24\,V$ - tensión de entrada
- $C = 10\,mF$ - capacitor
- $L = 10\,mH$ - inductor
- $R = 10\,\Omega$ - Resistencia de carga
- $f = 10\,kHz$ - frecuencia de conmutación
- $DC = 0,35$ - ciclo de trabajo
- $R_{\text{off}} = 1 \cdot 10^5\,\Omega$ - resistencia de corte
- $R_{\text{on}} = 1 \cdot 10^{-5}\,\Omega$ - resistencia de conducción
- $t_f = 10\,mseg$ - tiempo final de simulación
- $tol = 1 \cdot 10^{-3}$ - tolerancias absolutas y relativas de error de simulación

6.3.1. Inductancia parásita en la llave

Un caso de estudio interesante es la presencia de una inductancia parásita modelada en serie con el elemento de conmutación como se muestra en la Figura 6.3. Esta situación es particularmente interesante debido a que la presencia de dicha inductancia produce efectos indeseados (afecta la velocidad de conmutación del dispositivo [39, 40, 41], o puede producir sobretensiones).

Figura 6.3: Modelo de llave con inductancia parásita.

La Tabla 6.1 muestra los resultados de simular el convertidor Buck con la inductancia parásita mencionada de valor $L_s = 10nH$.

Tabla 6.1: Comparación tiempos de simulación para convertidor Buck con inductancia parásita en la llave.

Método de Integración	**Evaluaciónes de función f_i**	**CPU [mseg]**
DASSL	123,030	17
CVODE	40,494	15
IDA	112,572	18
LIQSS2	34,911,158	5,511
mLIQSS2	27,038	8

Podemos observar en la tabla que LIQSS2 no es capaz de simular el circuito con un desempeño comparable a los métodos clásicos. Esto se debe a que cuando la llave se cierra y el diodo está abierto, la inductancia L del circuito y la inductancia parásita L_s quedan con una configuración similar a la del circuito de la Figura 6.1b donde el diodo abierto juega el papel de la resistencia grande y si $R_D \to \infty$ aparecería una restricción entre las dos variables de estado. Como mencionamos en el Capítulo 4, este caso stiff no es manejado eficientemente por los LIQSS ya que los mismos introducen oscilaciones espurias entre dos variables. Sin embargo, el método mLIQSS2 sí logra integrar eficientemente el circuito, y además supera los resultados obtenidos con cualquiera de los métodos clásicos.

6.3.2. Inductancia parásita en el diodo

Similarmente a lo visto en el caso anterior, se suele modelar también la presencia de inductancia parásita en serie con el diodo, como puede verse en la Figura 6.4.

La Tabla 6.2 muestra los resultados de simular el convertidor Buck con una inductancia parásita en paralelo con el diodo de valor $L_d = 10nH$.

Encontramos aquí resultados similares a los del caso anterior. La situación análoga a la del circuito de la Fig.6.1b se da aquí mientras el diodo conduce y la llave está abierta, motivo por el cual LIQSS2 tampoco tiene resultados eficientes y mLIQSS2 sigue siendo la mejor opción.

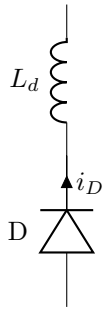


Figura 6.4: Modelo de diodo con inductancia parásita.

Tabla 6.2: Comparación tiempos de simulación para convertidor Buck con inductancia parásita en el diodo.

Método de Integración	**Evaluaciónes de función f_i**	**CPU [mseg]**
DASSL	38,938	18
CVODE	15,908	17
IDA	36,064	19
LIQSS2	1,410,530	194
mLIQSS2	35,584	6

6.3.3. Buck interleaved

Veremos a continuación la influencia del tamaño del sistema en estudio. Para ello, simularemos el circuito denominado *Buck interleaved*. Se trata de un convertidor Buck de varias etapas, donde cada una de ellas entra en conducción alternadamente. Estudiaremos el comportamiento de este circuito en presencia de una inductancia parásita modelada en serie con las llaves de cada etapa.

Se muestran en la Figura 6.5 los resultados de simular dicho circuito, variando el número de etapas que lo componen, con una inductancia parásita en las llaves de $L_s = 10nH$. Allí podemos observar que LIQSS2 no logra simular eficientemente el circuito, y sólo logra igualar a los métodos clásicos cuando el número de etapas se incrementa a 32, esto es, cuando se incre-

menta mucho el número de discontinuidades del sistema. Esto se debe a que si bien este método presenta las oscilaciones espurias mencionadas en el Capítulo 4, la familia de métodos QSS presenta mayores ventajas sobre los métodos clásicos a medida que el sistema tenga discontinuidades más frecuentes. Por otro lado, vemos que mLIQSS2 logra integrar eficientemente el circuito, superando el desempeño de los métodos clásicos. De hecho, se puede observar que la diferencia entre mLIQSS2 y CVODE(el método clásico con el que se obtienen mejores resultados) va creciendo a medida que se incrementa el tamaño del circuito, llegando a ser hasta 15 veces más rápido al simular 32 etapas.

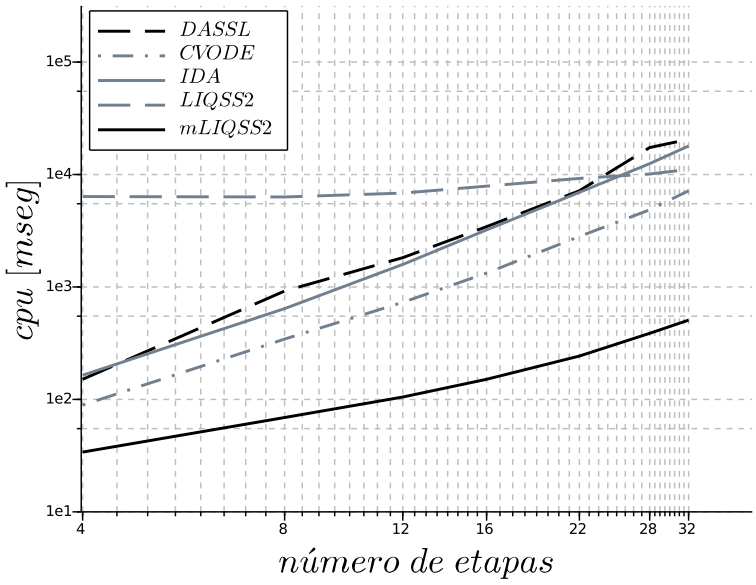


Figura 6.5: Buck interleaved con inductancia parásita en las llaves.

6.4. Convertidor Ćuk

Otro circuito de electrónica conmutada importante es el convertidor *Ćuk*. Al igual que en el caso del *Buck*, es un convertidor DC-DC. Este circuito es algo más complejo que el presentado anteriormente, ya que posee cuatro

elementos almacenadores de energía (por lo que se lo denomina un convertidor de orden cuatro) a diferencia de los dos presentes en el convertidor *Buck*. Podemos ver un diagrama de dicho circuito en la Figura 6.6. En él, tanto la llave como el diodo serán modelados del mismo modo que ya ha sido descripto previamente.

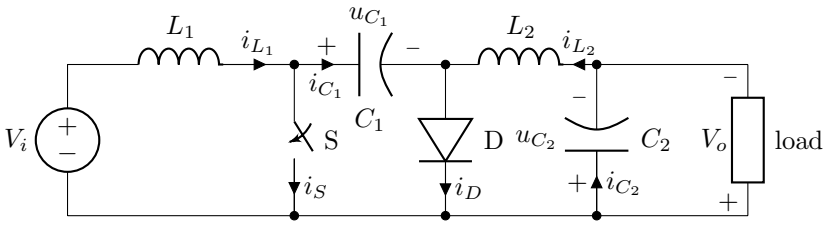


Figura 6.6: Circuito convertidor Ćuk.

En este caso, en ausencia de inductancias parásitas, los métodos de LIQSS no funcionan eficientemente, mientras que los mLIQSS sí lo hacen [23]. La razón es que si consideramos que la llave y el diodo están abiertos simultáneamente (con $R_{\text{off}} \to \infty$) entonces aparece una restricción sobre dos variables de estado: $i_{L_1} + i_{L_2} = 0$.

Estudiaremos entonces el caso en que exista una inductancia parásita modelada en serie con la llave de valor $L_s = 10nH$. Para las simulaciones utilizaremos además los siguientes parámetros:

- $U = 24\,V$ - tensión de entrada
- $C_1 = C_2 = 10\,mF$ - capacitores
- $L_1 = L_2 = 10\,mH$ - inductores
- $R = 10\,\Omega$ - Resistencia de carga
- $f = 10\,kHz$ - frecuencia de conmutación
- $DC = 0,35$ - ciclo de trabajo
- $R_{\text{off}} = 1 \cdot 10^5\,\Omega$ - resistencia de corte
- $R_{\text{on}} = 1 \cdot 10^{-5}\,\Omega$ - resistencia de conducción
- $t_f = 10\,mseg$ - tiempo final de simulación

- $tol = 1 \cdot 10^{-3}$ - tolerancias absolutas y relativas de error de simulación

Los resultados obtenidos con los distintos métodos de integración se muestran en la Tabla 6.3.

Tabla 6.3: Comparación tiempos de simulación para convertidor Ćuk con inductancia parásita en la llave.

Método de Integración	**Evaluaciónes de función f_i**	**CPU [mseg]**
DASSL	6,556,560	586
CVODE	480,190	16
IDA	152,825	16
LIQSS2	119,920,338	17,153
mLIQSS2	43,744,462	6,710

Vemos aquí que, aunque la simulación con el método mLIQSS2 mejora lo obtenido con LIQSS, los resultados siguen siendo ineficientes en comparación a los obtenidos utilizando métodos clásicos. Esto se debe a que un pequeño cambio en la corriente por cada uno de los inductores provoca un cambio grande en la derivada de los otros dos inductores, dando lugar a oscilaciones espurias que involucran tres variables de estado, situación que no contempla mLIQSS2. Otra forma de ver esto es que, considerando que el diodo está abierto (con $R_D \to \infty$) y la llave cerrada, aparece una restricción sobre tres variables de estado: $i_{L_1} + i_{L_2} = i_{L_s}$.

6.5. Conclusiones

En este capítulo hemos analizado el desempeño de diferentes algoritmos de integración numérica a la hora de simular algunos circuitos de electrónica conmutada en presencia de elementos parásitos. Se han comparado no sólo algunos de los métodos clásicos más utilizados, si no también algunos métodos de la familia QSS. Particularmente, se han utilizado los métodos LIQSS2 y su versión mejorada mLIQSS2.

Del análisis de los resultados, concluimos que:

- La presencia de elementos parásitos en diversos convertidores DC-DC se traduce en un tipo de rigidez que los métodos LIQSS no son capaces de integrar eficientemente, ya que aparecen oscilaciones espurias entre pares de variables de estado.

- Si el circuito en cuestión cumple con algunas restricciones de topología (que un pequeño cambio en una variable de estado no produzca un gran cambio en las derivadas de más de dos variables de estado), entonces la utilización de mLIQSS2 supera los resultados obtenidos tanto con LIQSS2 como con métodos clásicos.

- Estas condiciones sobre la topología pueden analizarse en función del número de variables de estado que pueden quedar involucradas en una restricción al hacer $R_{\text{off}} \to \infty$.

- Las ventajas de utilizar el método mLIQSS2 se hace más significativa a medida que el tamaño del circuito a simular aumenta, y como consecuencia de esto, las discontinuidades en el el modelo se hacen más frecuentes y la matriz Jacobiana del mismo crece en tamaño. Esto se debe a que, como los demás miembros de la familia QSS, este método logra integrar a través de discontinuidades de manera muy eficiente.

Capítulo 7

Integración mixta basada en cuantificación

7.1. Introducción

La condición para que los algoritmos LIQSS pueda integrar eficientemente un sistema stiff es que la rigidez se deba a la presencia de términos grandes en la diagonal principal de la matriz Jacobiana. Pese a que esta restricción fue relajada, domo fue explicado en los capítulos 4 y 5, y a que existen muchos problemas stiff en la práctica en los que los métodos LIQSS resultan eficientes, también existen casos en los que estos algoritmos son claramente superados en su desempeño por métodos clásicos. Un ejemplo de esto ocurre al simular ecuaciones parabólicos (la ecuación de calor, por ejemplo) discretizadas con el Método de Lineas (MOL), de la que se obtiene un sistema de EDOs stiff, pero en el cual no hay términos grandes en la matriz Jacobiana.

En general, los algoritmos QSS son más eficientes que los métodos clásicos en presencia de discontinuidades frecuentes y en la simulación de grandes sistemas que exhiben actividad no homogénea, es decir, cuando solo unas pocas variables tienen cambios significativos durante un intervalo de tiempo dado. Esto se explica por el hecho de que los métodos QSS solo realizan cálculos en las variables de estado que experimentan cambios significativos. Por el contrario, en sistemas con actividad homogénea y sin discontinuidades frecuentes, los solucionadores de EDO clásicos suelen ser más eficientes [42].

Hay sistemas que combinan subsistemas donde los algoritmos QSS son mejores, con otros subsistemas en los que los métodos clásicos resultan la

mejor opción. Para estos casos, en este capítulo se propone un enfoque de modo mixto que utiliza QSS para integrar algunos subsistemas y métodos clásicos para los otros subsistemas. Al igual que en la mayoría de los enfoques de integración de modo mixto, el principal problema es definir la interfaz entre ambos algoritmos.

Además de proponer una metodología general de modo mixto para QSS y métodos clásicos, en este capítulo también se estudian las propiedades de convergencia y estabilidad de la aproximación resultante, y se describen dos implementaciones particular, una de primer orden que combina LIQSS1 con Backwar Euler, y otra de segundo orden que combina LIQSS2 con un algoritmo BDF2, para ambos métodos clásicos se utiliza un paso variable. Además, se presentan dos ejemplos de simulación, el primero corresponde a un modelo que representa el calentamiento alrededor de un convertidor electrónico conmutado de potencia , y el segundo corresponde a un problema de advección–difusión–reacción unidimensional donde el coeficiente de difusión cambia con el espacio.

7.2. Integración mixta basada en cuantificación (QMM)

Como mencionamos anteriormente en la introducción, los algoritmos QSS son eficientes en la simulación de sistemas con frecuentes discontinuidades, y LIQSS también son eficientes en la integración numérica de algunos sistemas rígidos. Sin embargo, hay varios casos en los que los métodos numéricos clásicos son claramente superiores. Los sistemas complejos generalmente combinan subsistemas con diferentes características, y puede ocurrir que alguna parte del modelo sea más adecuada para los algoritmos QSS, mientras que el resto puede resolverse de manera más eficiente mediante los métodos de integración numérica clásicos.

Para estos casos, proponemos un esquema de modo mixto en el que un subconjunto de variables de estado de un sistema se integra utilizando un algoritmo QSS, mientras que el resto de los estados se integran utilizando un algoritmo de tiempo discreto clásico.

Para definir la metodología, dada la EDO de la Ec. (3.6), primero la dividimos como el acoplamiento de dos subsistemas,

$$\dot{\mathbf{x}}_{\mathbf{1}}(t) = \mathbf{f}_{\mathbf{1}}(\mathbf{x}_{\mathbf{1}}(t), \mathbf{x}_{\mathbf{2}}(t), t) \tag{7.1a}$$

$$\dot{\mathbf{x}}_{\mathbf{2}}(t) = \mathbf{f}_{\mathbf{2}}(\mathbf{x}_{\mathbf{1}}(t), \mathbf{x}_{\mathbf{2}}(t), t) \tag{7.1b}$$

donde $\mathbf{x}(t) = [\mathbf{x}_{\mathbf{1}}(t), \mathbf{x}_{\mathbf{2}}(t)]^T$, y $\mathbf{f}(\cdot) = [\mathbf{f}_{\mathbf{1}}(\cdot), \mathbf{f}_{\mathbf{2}}(\cdot)]^T$.

Luego, proponemos integrar $\mathbf{x_1}(t)$ usando un método QSS y $\mathbf{x_2}(t)$ utilizando algún algoritmo clásico con control de paso $h_k = t_{k+1} - t_k$, esto es,

$$\dot{\mathbf{x}}_\mathbf{1}(t) = \mathbf{f_1}(\mathbf{q_1}(t), \tilde{\mathbf{x}}_\mathbf{2}(t), t) \tag{7.2a}$$

$$\mathbf{x_2}(t_{k+1}) = \mathbf{F_2}(\mathbf{q_1}(t_k), \mathbf{x_2}(t_k), t_k, h_k) \tag{7.2b}$$

donde Eq. (7.2b) representa el caso de un método monopaso explícito. Para un caso más general, podría ser el resultado de

$$\tilde{\mathbf{F}}_\mathbf{2}(\mathbf{q_1}(t_k), \mathbf{x_2}(t_{k+1}), \mathbf{x_2}(t_k), \mathbf{x_2}(t_{k-1}), \ldots, t_k, h_k) = 0 \tag{7.3}$$

que también puede representar un método multipaso explícito.

La interfaz entre ambos algoritmos se realiza mediante el muestreo de los estados cuantificados $\mathbf{q_1}(t)$ en los pasos de tiempo t_k del algoritmo clásico y mediante la extrapolación de los estados del algoritmo clásico $\mathbf{x_2}(t_k)$ según el polinomio

$$\tilde{\mathbf{x}}_\mathbf{2}(t) = \mathbf{x_2}(t_k) + \mathbf{f_2}(\mathbf{q_1}(t_k), \mathbf{x_2}(t_k), t_k) \cdot (t - t_k) + \ddot{\tilde{\mathbf{x}}}_\mathbf{2}(t_k) \cdot \frac{(t-t_k)^2}{2!} + \ldots \tag{7.4}$$

para $t \in [t_k, t_{k+1})$. El polinomio de extrapolación tiene el mismo orden que el estado cuantificado del algoritmo QSS correspondiente (es decir, 0 para QSS1 y LIQSS1, 1 para QSS2 y LIQSS2, etc.). En la Ec. (7.4) $\ddot{\tilde{\mathbf{x}}}_\mathbf{2}(t_k)$ representa una aproximación de $\ddot{\mathbf{x}}_\mathbf{2}(t_k)$..

7.2.1. Convergencia de métodos mixtos QSS–clásicos

El siguiente teorema establece condiciones suficientes para la convergencia de las soluciones del esquema de modo mixto a la solución analítica cuando el quantum del algoritmo QSS y el tamaño de paso del algoritmo clásico tienden simultáneamente a cero.

Teorema (Convergencia del Esquema de Modo Mixto)**.** *Considere el sistema de la Ec.* (7.1) *donde* $\mathbf{f}(\mathbf{x}, t) = [\mathbf{f_1}(\cdot), \mathbf{f_2}(\cdot)]^T$ *es localmente Lipschitz en* $\mathbf{x}$ *en de un conjunto cerrado* $D \subset \mathbb{R}^n$*. Sea* $\mathbf{x_a}(t)$ *una solución de la Ec.* (3.6) *a partir de condiciones iniciales* $\mathbf{x_a}(t_0)$ *tal que* $\mathbf{x_a}(t)$ *se mantenga en el interior de* D *durante el intervalo* $[t_0, t_f]$*. Sea* $\mathbf{x}(t_k) = [\mathbf{x_1}(t_k), \mathbf{x_2}(t_k)]^T$ *la solución del esquema de modo mixto de las Ecs .*(7.1)–(7.4) *a aprtir de las mismas condiciones iniciales* $\mathbf{x}(t_0) = \mathbf{x_a}(t_0)$*, donde el algoritmo QSS utilizan quantos* $\Delta Q_i < \Delta Q_{\text{máx}}$ *y el algoritmo clásico es al menos de primer*

orden de precisión y utiliza un paso de integración limitado superiormente $h_k < h_{max}$. *Asumama además que los coeficientes de extrapolación polinomial de la Ec.* (7.4) *están acotados en* $D \times [t_0, t_f]$.

Entonces,

$$\lim_{\Delta Q_{\text{máx}}, h_{\text{máx}} \to 0} \mathbf{x}(t_k) = \mathbf{x_a}(t_k) \tag{7.5}$$

Para todo $t_k \in [t_0, t_f]$.

Demostración. Dado que $\mathbf{f}$ es localmente Lipschitz en D, existe una constante $L > 0$ tal que

$$\| \mathbf{f}(\mathbf{x_c}, t) - \mathbf{f}(\mathbf{x_b}, t) \| < L \cdot \| \mathbf{x_c} - \mathbf{x_b} \| \tag{7.6}$$

para todo $\mathbf{x_a}, \mathbf{x_b} \in D$ y para todo $t \in [t_0, t_f]$. Aquí, el símbolo $\|\mathbf{x}\|$ denota la norma infinita del vector $\mathbf{x}$.

La condición de la Ec. (7.6) implica que existe una constante $F_{\text{máx}} > 0$ tal que

$$\| \mathbf{f}(\mathbf{x}, t) \| < F_{\text{máx}} \tag{7.7}$$

para $t \in [t_0, t_f]$ y $\mathbf{x} \in D$.

Asumiremos que que el paso de integración está limitado superiormente por una cierta constante $\bar{h}$. Luego, teniendo en cuenta que el método clásico es de al menos primer orden de precisión, podemos escribir:

$$\mathbf{x_2}(t_{k+1}) = \mathbf{x_2}(t_k) + h_k \cdot \mathbf{f_2}(\mathbf{q_1}(t_k), \mathbf{x_2}(t_k), t_k) + \frac{h_k^2}{2} \cdot \mathbf{r_h}(t_k) \tag{7.8}$$

donde

$$\| \mathbf{r_h}(t_k) \| < r_{\text{máx}} \quad \forall t_k \in [t_0, t_f] \tag{7.9}$$

y donde $r_{\text{máx}}$ es una constante que acota el termino resto $\mathbf{r_h}(t_k)$ para cualquier paso de integración $h_k \leq \bar{h}$, y para cualquier $[\mathbf{q_1}(t_k), \mathbf{x_2}(t_k)]^T \in D$. Nótese que $r_{\text{máx}}$ puede depender del método numérico utilizado, de la expresión de $\mathbf{f_2}$, y de la cota máxima del paso de integración $\bar{h}$.

Dadp que $\mathbf{x_2}$ en la Ec. (7.2b) sólo está definido para tiempos discretos t_k,, extendemos su definición para el intervalo $[t_0, t_f]$ commpleto como sigue:

$$\mathbf{x_2}(t) \triangleq \mathbf{x_2}(t_k) + \frac{\mathbf{x_2}(t_{k+1}) - \mathbf{x_2}(t_k)}{h_k} \cdot (t - t_k) \qquad t_k \leq t < t_{k+1} \tag{7.10}$$

Reemplazando el término $\mathbf{x_2}(t_{k+1}) - \mathbf{x_2}(t_k)$ de esta última definición en la Ec. (7.8), obtenemos

$$\mathbf{x_2}(t) = [\mathbf{f_2}(\mathbf{q_1}(t_k), \mathbf{x_2}(t_k), t_k) + h_k \cdot \mathbf{r_h}(t_k)](t - t_k),$$

una expresión que también se puede obtener después de integrar ambos lados de la EDO

$$\dot{\mathbf{x}}_{\mathbf{2}}(t) = \mathbf{f}_{\mathbf{2}}(\mathbf{q}_{\mathbf{1}}(t_k), \mathbf{x}_{\mathbf{2}}(t_k), t_k) + h_k \cdot \mathbf{r}_{\mathbf{h}}(t_k) \tag{7.11}$$

en el intervalo $[t_k, t]$ con $t_k \leq t < t_{k+1}$. De ese modo, $\mathbf{x}_{\mathbf{2}}(t)$ definido en Ec. (7.10) es la solución de la EDO de la Ec.(7.11) a aprtir de la condición inicial $\mathbf{x}_{\mathbf{2}}(t_0)$.

En consecuencia, el sistema dado por la Ec. (7.2) tiene una solución idéntica a la dada por

$$\dot{\mathbf{x}}_{\mathbf{1}}(t) = \mathbf{f}_{\mathbf{1}}(\mathbf{q}_{\mathbf{1}}(t), \tilde{\mathbf{x}}_{\mathbf{2}}(t), t) \tag{7.12a}$$
$$\dot{\mathbf{x}}_{\mathbf{2}}(t) = \mathbf{f}_{\mathbf{2}}(\mathbf{q}_{\mathbf{1}}(t_k), \mathbf{x}_{\mathbf{2}}(t_k), t_k) + h_k \cdot \mathbf{r}_{\mathbf{h}}(t_k) \tag{7.12b}$$

en tiempos discretos t_k. Usando este hecho, probaremos que $\mathbf{x}(t) = [\mathbf{x}_{\mathbf{1}}(t), \mathbf{x}_{\mathbf{2}}(t)]^T \rightarrow \mathbf{x}_{\mathbf{a}}(t)$, lo cual implica Ec.(7.5).

Definiendo

$$\boldsymbol{\Delta}_{\mathbf{1,1}}(t) \triangleq \mathbf{q}_{\mathbf{1}}(t) - \mathbf{x}_{\mathbf{1}}(t), \quad \boldsymbol{\Delta}_{\mathbf{1,2}}(t) \triangleq \tilde{\mathbf{x}}_{\mathbf{2}}(t) - \mathbf{x}_{\mathbf{2}}(t)$$
$$\boldsymbol{\Delta}_{\mathbf{2,1}}(t) \triangleq \mathbf{q}_{\mathbf{1}}(t_k) - \mathbf{x}_{\mathbf{1}}(t), \quad \boldsymbol{\Delta}_{\mathbf{2,2}}(t) \triangleq \mathbf{x}_{\mathbf{2}}(t_k) - \mathbf{x}_{\mathbf{2}}(t)$$

reescribimos Ec. (7.12) como sigue

$$\dot{\mathbf{x}}_{\mathbf{1}}(t) = \mathbf{f}_{\mathbf{1}}(\mathbf{x}_{\mathbf{1}}(t) + \boldsymbol{\Delta}_{\mathbf{1,1}}(t), \mathbf{x}_{\mathbf{2}}(t) + \boldsymbol{\Delta}_{\mathbf{1,2}}(t), t) \tag{7.13a}$$
$$\dot{\mathbf{x}}_{\mathbf{2}}(t) = \mathbf{f}_{\mathbf{2}}(\mathbf{x}_{\mathbf{1}}(t) + \boldsymbol{\Delta}_{\mathbf{2,1}}(t), \mathbf{x}_{\mathbf{2}}(t) + \boldsymbol{\Delta}_{\mathbf{2,2}}(t), t_k) + h_k \cdot \mathbf{r}_{\mathbf{h}}(t_k) \tag{7.13b}$$

Lso términos de perturbación $\boldsymbol{\Delta}_{\mathbf{i,j}}(t)$ pueden ser acotados como sigue:

$$\|\boldsymbol{\Delta}_{\mathbf{1,1}}(t)\| \leq \Delta Q_{\text{máx}} = c_{1,1} \Delta Q_{\text{máx}} \tag{7.14}$$

A partir de las Ecs. (7.4) y (7.10), obtenemos

$$\boldsymbol{\Delta}_{\mathbf{1,2}}(t) = [\mathbf{f}_{\mathbf{2}}(\mathbf{q}_{\mathbf{1}}(t_k), \mathbf{x}_{\mathbf{2}}(t_k), t_k) - \frac{\mathbf{x}_{\mathbf{2}}(t_{k+1}) - \mathbf{x}_{\mathbf{2}}(t_k)}{h_k}] \cdot (t - t_k) +$$
$$+ \ddot{\tilde{\mathbf{x}}}_{\mathbf{2}}(t_k) \cdot \frac{(t - t_k)^2}{2!} + \dots$$

Luego, usando la Ec. (7.8), resulta

$$\boldsymbol{\Delta}_{\mathbf{1,2}}(t) = [\mathbf{f}_{\mathbf{2}}(\mathbf{q}_{\mathbf{1}}(t_k), \mathbf{x}_{\mathbf{2}}(t_k), t_k) - \mathbf{f}_{\mathbf{2}}(\mathbf{q}_{\mathbf{1}}(t_k), \mathbf{x}_{\mathbf{2}}(t_k), t_k) - \frac{h_k}{2} \cdot \mathbf{r}_{\mathbf{h}}(t_k)] \cdot (t - t_k) +$$
$$+ \ddot{\tilde{\mathbf{x}}}_{\mathbf{2}}(t_k) \cdot \frac{(t - t_k)^2}{2!} + \dots$$
$$= [\ddot{\tilde{\mathbf{x}}}_{\mathbf{2}}(t_k) \cdot \frac{(t - t_k)}{2!} + \dots - \frac{h_k}{2} \cdot \mathbf{r}_{\mathbf{h}}(t_k)] \cdot (t - t_k)$$

Entonces, dado que los coeficientes polinomiales de extrapolación están acotados en D, y recordando la Ec. (7.9) y el hecho de que $t - t_k \leq h_{\text{máx}} \leq \bar{h}$, resulta que

$$\|\boldsymbol{\Delta}_{\mathbf{1,2}}(t)\| \leq c_{1,2} \cdot h_{\text{máx}} \tag{7.15}$$

donde $c_{1,2}$ está acotado superiormente por $\|\ddot{\tilde{\mathbf{x}}}_{\mathbf{2}}(t_k) \cdot \frac{(t-t_k)}{2!} + \cdots - \frac{h_k}{2} \cdot \mathbf{r_h}(t_k)\|$. Nótese que esta cota es válida bajo el supuesto de que $\mathbf{f_1}$ y $\mathbf{f_2}$ in Ec. (7.13) son evaluados dentro de D.

Respecto a la cota para $\boldsymbol{\Delta}_{\mathbf{2,1}}(t)$, sabemos que existe $t^* \in [t_k, t]$ tal que, de acuerdo a la desigualdad del valor medio, resulta que

$$\|\mathbf{x_1}(\mathbf{t}) - \mathbf{x_1}(\mathbf{t_k})\| \leq \|\dot{\mathbf{x}}_{\mathbf{1}}(t^*)\| \cdot (t - t_k) \leq F_{\text{máx}} \cdot h_k$$

y luego, recordando que $\boldsymbol{\Delta}_{\mathbf{2,1}}(t) = \mathbf{q_1}(t_k) - \mathbf{x_1}(t) = \mathbf{q_1}(t_k) - \mathbf{x_1}(t_k) + \mathbf{x_1}(t_k) - \mathbf{x_1}(t)$, obtenemos

$$\|\boldsymbol{\Delta}_{\mathbf{2,1}}(t)\| \leq \Delta Q_{\text{máx}} + F_{\text{máx}} \cdot h_{\text{máx}} \leq c_{2,1}\, \text{máx}(\Delta Q_{\text{máx}}, h_{\text{máx}}) \tag{7.16}$$

una cota que también es válida bajo el supuesto de que $\mathbf{f_1}$ y $\mathbf{f_2}$ en Ec. (7.13) son evaluadas dentro de D.

El último término de perturbación, $\boldsymbol{\Delta}_{\mathbf{2,2}}(t)$, puede ser calculado de las Ec. (7.8) y (7.10) como

$$\begin{aligned}\boldsymbol{\Delta}_{\mathbf{2,2}}(t) = \mathbf{x_2}(t) - \mathbf{x_2}(t_k) = \frac{\mathbf{x_2}(t_{k+1}) - \mathbf{x_2}(t_k)}{h_k}(t - t_k) = \\ = [\mathbf{f_2}(\mathbf{q_1}(t_k), \mathbf{x_2}(t_k), t_k) + \frac{h_k}{2} \cdot \mathbf{r_h}(t_k)] \cdot (t - t_k)\end{aligned}$$

and then,

$$\|\boldsymbol{\Delta}_{\mathbf{2,2}}(t)\| \leq (F_{\text{máx}} + h_{\text{máx}} \cdot r_{\text{máx}}) \cdot h_{\text{máx}} = c_{2,2} \cdot h_{\text{máx}} \tag{7.17}$$

Esta desigualdad también asume que $\mathbf{f_1}$ y $\mathbf{f_2}$ en Ec. (7.13) son evaluadas dentro de D.

El hecho de que la solución analítica $\mathbf{x_a}(t)$ esté en el interior de D para $t \in [t_0, t_f]$ implica que existe una constante $d > 0$ tal que

$$\text{dist}(\mathbf{x_a}(t), \partial D) > d \quad \forall t \in [t_0, t_f] \tag{7.18}$$

Tomemos $T \in (0, t_f - t_0]$ tal que

$$T < \frac{d}{4 \cdot (F_{\text{máx}} + \bar{h} \cdot r_{\text{máx}})} \tag{7.19}$$

y sean $\Delta Q_{\text{máx}}$ y $h_{\text{máx}}$ lo suficientemente pequeños como para que el lado derecho de las Ecs. (7.14),(7.15),(7.16), y (7.17) sean menores que $d/4$.

Sea t_a algúns instante de tiempo en el que $\text{dist}(\mathbf{x}(t_a), \partial D) > d/2$, y sea $t_e \in (t_a, t_a + T)$ el primer instante de tiempo en el que $\text{dist}(\mathbf{x}(t_e), \partial D) \leq d/4$. Así, tanto $\mathbf{f_1}$ como $\mathbf{f_2}$ son calculados dentro de D en Ec. (7.13) durante el intervalo $[t_a, t_e)$ y las cotas dadas para $\|\mathbf{\Delta_{i,j}}(t)\|$ por las Ecs. (7.14),(7.15),(7.16), and (7.17) son válidas en ese intervalo. let $t_e \in (t_a, t_a + T)$ be the first instant of time at which $\text{dist}(\mathbf{x}(t_e), \partial D) \leq d/4$. Thus, $\mathbf{f_1}$ and $\mathbf{f_2}$ are both computed inside D in Eq.(7.13) during the interval $[t_a, t_e)$ and the bounds given for $\|\mathbf{\Delta_{i,j}}(t)\|$ by Eqs.(7.14),(7.15),(7.16), and (7.17) are valid in that interval.

Luego, de acuerdo a la desigualdad del valor medio, existe $t^* \in [t_a, t_e]$ tal que

$$\|\mathbf{x}(t_e) - \mathbf{x}(t_a)\| \leq \|\dot{\mathbf{x}}(t^*)\| \cdot (t_e - t_a) \leq (F_{\text{máx}} + \bar{h} \cdot r_{\text{máx}}) \cdot T < \frac{d}{4}$$

Dado que $\text{dist}(\mathbf{x}(t_a), \partial D) > d/2$, la última desigualdad implica que $\text{dist}(\mathbf{x}(t_e), \partial D) > d/4$, contradiciendo la definición de t_e y mostrando que la distancia desde el estado $\mathbf{x}(t)$ a la frontera de D es de al menos $d/4$ en $[t_a, t_a + T]$.

Sea $\varepsilon > 0$ y considere algún $\delta > 0$ lo suficientemente pequeño tal que $\Delta Q_{\text{máx}} < \delta$ y $h_{\text{máx}} < \delta$ implica que el lado derecho de las desigualdades (7.14),(7.15),(7.16), y (7.17) son menores a $d/4$.

Supongamos ahora que hay un tiempo $t_b \geq t_0$, con $t_b \leq t_f - T$, tal que

$$\|\mathbf{x}(t) - \mathbf{x_a}(t)\| < \text{mín}(\varepsilon, d/4) \tag{7.20}$$

para todo $t \in [t_0, t_b]$. A continuación demostraremos que podemos encontrar $\delta > 0$ de tal manera que las condiciones $\Delta Q_{\text{máx}} < \delta$ y $h_{\text{máx}} < \delta$ impliquen que $\|\mathbf{x}(t) - \mathbf{x_a}(t)\| < \varepsilon$ for all $t \in [t_0, t_b + T]$.

Nótese que $\|\mathbf{x}(t_b) - \mathbf{x_a}(t_b)\| < d/4$ implica que $\text{dist}(\mathbf{x}(t_b), \partial D) > d/2$, lo que a su vez implica que $\text{dist}(\mathbf{x}(t), \partial D) > d/4$ para todo $t \in [t_0, t_b + T]$. Luego recordando que δ es lo suficientemente pequeño para que $\|\mathbf{\Delta_{i,j}}(t)\| < d/4$, resulta que ambas funciones $\mathbf{f_1}$ y $\mathbf{f_2}$ en Ec. (7.13) son calculas dentro de D, lo que a su vez implica la validez de las Ecs. (7.14),(7.15),(7.16), y (7.17).

Recordando que $\mathbf{x_a}(t) = [\mathbf{x_{a,1}}(t), \mathbf{x_{a,2}}(t)]^T$ es la solución de la Ec. (7.1), y sustrayendo Ec. (7.1a) de Eq.(7.13a), obtenemos

$$\dot{\mathbf{x}}_\mathbf{1}(t) - \dot{\mathbf{x}}_{\mathbf{a,1}}(t) = \mathbf{f_1}(\mathbf{x_1}(t) + \mathbf{\Delta_{1,1}}(t), \mathbf{x_2}(t) + \mathbf{\Delta_{1,2}}(t), t) - \mathbf{f_1}(\mathbf{x_{a,1}}(t), \mathbf{x_{a,2}}(t), t)$$

o, equivalentemente,

$$\mathbf{x_1}(t)-\mathbf{x_{a,1}}(t) = \int_{t_0}^{t} [\mathbf{f_1}(\mathbf{x_1}(\tau)+\mathbf{\Delta_{1,1}}(\tau), \mathbf{x_2}(\tau)+\mathbf{\Delta_{1,2}}(\tau), \tau)-\mathbf{f_1}(\mathbf{x_{a,1}}(\tau), \mathbf{x_{a,2}}(\tau), \tau)]\mathrm{d}\tau$$

y, usando la condición de Lipschitz en D,

$$\begin{aligned}
\|\mathbf{x_1}(t) - \mathbf{x_{a,1}}(t)\| &\leq \int_{t_0}^{t} L \cdot [\|\mathbf{x_1}(\tau) + \mathbf{\Delta_{1,1}}(\tau) - \mathbf{x_{a,1}}(\tau)\| + \|\mathbf{x_2}(\tau) + \mathbf{\Delta_{1,2}}(\tau) - \mathbf{x_{a,2}}(\tau)\|]\mathrm{d}\tau \\
&\leq \int_{t_0}^{t} 2L \cdot \|\mathbf{x}(\tau) - \mathbf{x_a}(\tau)\|\mathrm{d}\tau + L[c_{1,1}\Delta Q_{\text{máx}} + c_{1,2}h_{\text{máx}}] \cdot (t - t_0) \\
&\leq \int_{t_0}^{t} 2L \cdot \|\mathbf{x}(\tau) - \mathbf{x_a}(\tau)\|\mathrm{d}\tau + L[c_{1,1} + c_{1,2}] \cdot \delta \cdot (t - t_0)
\end{aligned} \tag{7.21}$$

para todo $t \in [t_0, t_b + T]$.

Procediendo de una manera similar y restando ahora Ec. (7.1b) de Ec. (7.13b), obtenemos

$$\begin{aligned}
\dot{\mathbf{x}}_{\mathbf{2}}(t) - \dot{\mathbf{x}}_{\mathbf{a,2}}(t) =& \mathbf{f_2}(\mathbf{x_1}(t) + \mathbf{\Delta_{2,1}}(t), \mathbf{x_2}(t) + \mathbf{\Delta_{2,2}}(t), t) + h_k \cdot \mathbf{r_h}(t_k) \\
&- \mathbf{f_2}(\mathbf{x_{a,1}}(t), \mathbf{x_{a,2}}(t), t)
\end{aligned}$$

y,

$$\begin{aligned}
\|\mathbf{x_2}(t) - \mathbf{x_{a,2}}(t)\| &\leq \int_{t_0}^{t} L \cdot [\|\mathbf{x_1}(\tau) + \mathbf{\Delta_{2,1}}(\tau) - \mathbf{x_{a,1}}(\tau)\| + \|\mathbf{x_2}(\tau) + \mathbf{\Delta_{2,2}}(\tau) - \mathbf{x_{a,2}}(\tau)\|]\mathrm{d}\tau + \\
&+ h_{\text{máx}} \cdot r_{\text{máx}}(t - t_0) \\
&\leq \int_{t_0}^{t} 2L \cdot \|\mathbf{x}(\tau) - \mathbf{x_a}(\tau)\|\mathrm{d}\tau + L[c_{2,1}\,\text{máx}(\Delta Q_{\text{máx}}, h_{\text{máx}}) + c_{2,2}h_{\text{máx}}] \cdot (t - t_0) \\
&+ h_{\text{máx}} \cdot r_{\text{máx}} \cdot (t - t_0) \\
&\leq \int_{t_0}^{t} 2L \cdot \|\mathbf{x}(\tau) - \mathbf{x_a}(\tau)\|\mathrm{d}\tau + [c_{2,1} + c_{2,2} + r_{\text{máx}}] \cdot \delta(t - t_0)
\end{aligned} \tag{7.22}$$

Luego, de las Ecs. (7.21) y (7.22), obtenemos

$$\|\mathbf{x}(t)-\mathbf{x_a}(t)\| \leq \int_{t_0}^{t} 4L\cdot\|\mathbf{x}(\tau)-\mathbf{x_a}(\tau)\|\mathrm{d}\tau+L[c_{1,1}+c_{1,2}+c_{2,1}+c_{2,2}+r_{\text{máx}}]\cdot\delta(t-t_0)$$

Aplicando la desigualdad de Grönwall–Bellman en la última expresión, obtenemos

$$\|\mathbf{x}(t) - \mathbf{x_a}(t)\| \leq L[c_{1,1} + c_{1,2} + c_{2,1} + c_{2,2} + r_{\text{máx}}] \cdot \delta(t - t_0) \cdot e^{4L(t-t_0)}$$

De ese modo, tomando

$$\delta < \frac{\text{mín}(\varepsilon, d/4)}{L[c_{1,1} + c_{1,2} + c_{2,1} + c_{2,2} + r_{\text{máx}}] \cdot (t_f - t_0) \cdot e^{4L(t_f - t_0)}}$$

resulta que

$$\|\mathbf{x}(t) - \mathbf{x_a}(t)\| < \text{mín}(\varepsilon, d/4) \leq \varepsilon \tag{7.23}$$

para todo $t \in [t_0, t_b + T]$.

Dado que la condición de la Ec. (7.20) es verificada con $t_b = t_0$, y teniendo en cuenta que esto implica Ec.(7.23), este razonamiento puede ser aplicado recursivamente con $t_b = t_0 + T$, $t_b = t_0 + 2T$, etc.. Cuando $t_0 + m \cdot T > t_f - T$ podemos tomar $t_b = t_f - T$ y aplicar el procedimiento para el último tiempo concluyendo que Ec.(7.23) se verifica para todo $t \in [t_0, t_f]$.

De este modo, dado $\varepsilon > 0$ podemos hallar $\delta > 0$ tal que $\Delta Q_{\text{máx}} \leq \delta$ y $h_{\text{máx}} \leq \delta$ implican que la Ec.(7.23) se verifica. Esto implica (7.5) concluyendo la demostración □

Después de probar la convergencia del esquema de modo mixto, la siguiente sección estudia la estabilidad de la metodología propuesta.

7.2.2. Estabilidad de métodos mixtos QSS–clásicos

Para analizar la estabilidad del esquema de modo mixto, primero consideraremos un sistema lineal invariante en el tiempo de la forma

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

que puede ser dividido como

$$\begin{bmatrix} \dot{\mathbf{x}}_\mathbf{1}(t) \\ \dot{\mathbf{x}}_\mathbf{2}(t) \end{bmatrix} = \begin{bmatrix} \mathbf{A_{11}} & \mathbf{A_{12}} \\ \mathbf{A_{21}} & \mathbf{A_{22}} \end{bmatrix} \begin{bmatrix} \mathbf{x_1}(t) \\ \mathbf{x_2}(t) \end{bmatrix}$$

Consideraremos que se usa un método QSS para calcular $\mathbf{x_1}$ y que el algoritmo de Backward Euler se usa para calcular $\mathbf{x_2}$. El análisis se puede ampliar fácilmente para considerar otros algoritmos clásicos.

El uso de un algoritmo QSS en $\mathbf{x_1}$ implica que

$$\dot{\mathbf{x}}_\mathbf{1}(t) = \mathbf{A_{11}} \cdot \mathbf{x_1}(t) + \mathbf{A_{12}} \cdot \mathbf{x_2}(t_k) + \mathbf{A_{12}} \cdot \mathbf{\Delta x_1}(t)$$

para $t \in [t_k, t_{k+1})$, donde $\mathbf{\Delta x_1}(t) = \mathbf{q}(t) - \mathbf{x_1}(t)$.

Podemos resolver esta última expresión para $t = t_{k+1}$, obteniendo

$$\begin{aligned}\mathbf{x_1}(t_{k+1}) &= e^{\mathbf{A_{11}}\cdot h}\cdot\mathbf{x_1}(t_k) + \int_{t_k}^{t_{k+1}} e^{\mathbf{A_{11}}\cdot(t-\tau)}\cdot\mathbf{A_{12}}\cdot\mathbf{x_2}(t_k)d\tau \\ &\quad + \underbrace{\int_{t_k}^{t_{k+1}} e^{\mathbf{A_{11}}\cdot(t-\tau)}\cdot\mathbf{A_{11}}\cdot\mathbf{\Delta x_1}(t)d\tau}_{\mathbf{\Delta}(t_k)} \\ &= e^{\mathbf{A_{11}}\cdot h}\cdot\mathbf{x_1}(t_k) + \mathbf{A_{11}}^{-1}\cdot e^{\mathbf{A_{11}}\cdot h}\cdot\mathbf{A_{12}}\cdot\mathbf{x_2}(t_k) + \mathbf{\Delta}(t_k)\end{aligned} \tag{7.24}$$

Luego, el uso de Backward Euler en el segundo subsistema lleva a

$$\mathbf{x_2}(t_{k+1}) = \mathbf{x_2}(t_k) + h\cdot\mathbf{A_{21}}\cdot\mathbf{x_1}(t_k) + h\cdot\mathbf{A_{22}}\cdot\mathbf{x_2}(t_{k+1})$$

que puede ser rescrito como

$$\mathbf{x_2}(t_{k+1}) = (\mathbf{I} - h\cdot\mathbf{A_{22}})^{-1}\cdot h\cdot\mathbf{A_{21}}\cdot\mathbf{x_1}(t_k) + (\mathbf{I} - h\cdot\mathbf{A_{22}})^{-1}\cdot\mathbf{x_2}(t_k) \tag{7.25}$$

Luego, colocando juntas las ecuaciones Ecs. (7.24) y (7.25), finalmente obtenemos

$$\mathbf{x}(t_{k+1}) = \mathbf{F}\cdot\mathbf{x}(t_k) + \mathbf{G}\cdot\mathbf{\Delta}(t_k) \tag{7.26}$$

donde

$$\mathbf{F} \triangleq \begin{bmatrix} e^{\mathbf{A_{11}}\cdot h} & \mathbf{A_{11}}^{-1}\cdot e^{\mathbf{A_{11}}\cdot h}\cdot\mathbf{A_{12}} \\ (\mathbf{I} - h\cdot\mathbf{A_{22}})^{-1}\cdot h\cdot\mathbf{A_{21}} & (\mathbf{I} - h\cdot\mathbf{A_{22}})^{-1} \end{bmatrix} \qquad \mathbf{G} \triangleq \begin{bmatrix}\mathbf{0}\\ \mathbf{I}\end{bmatrix} \tag{7.27}$$

Nótese que $\|\mathbf{\Delta x_1}\|$ está limitado por el quantum máximo utilizado $\Delta Q_{\text{máx}}$. Luego, el término $\mathbf{\Delta}(t)$ definido en la Ec. (7.24) también está limitado por una cantidad que es independiente en el estado $\mathbf{x}$. De esa manera, siempre que los autovalores de la matriz $\mathbf{F}$ en la Ec. (7.27) estén todos dentro del círculo unitario, la aproximación numérica finalmente tiene soluciones acotadas (es decir, tiene estabilidad práctica) para cualquier valor de h y $\Delta Q_{\text{máx}}$.

De esa manera, la estabilidad del esquema resultante puede depender del tamaño del paso h, pero es independiente del quamtum máximo $\Delta Q_{\text{máx}}$.

7.3. Implementación de los algoritmos de modo mixto

Se desarrollaron e implementaron dos nuevos algoritmos en el *stand–alone QSS solver*, ya mencionado. Estos algoritmos son un método de primer orden (QMM1), que combina LIQSS1 y Backward Euler, y un método

de segundo orden (QMM2), que combina LIQSS2 con BDF2 (fórmula de diferenciación hacia atrás de segundo orden).

Ambos algoritmos combinan un método clásico implícito con un método QSS linealmente implícito, para integrar eficientemente sistemas sitff. Además, ambos tienen control de paso para su parte clásica. En el caso de QMM1, el control de paso se realiza comparando los pasos de Backwar Euler con los de Forward Euler. Luego, en QMM2, este control se implementa mediante la comparación entre BDF2 y AB2 (método de Adams Bashforth de segundo orden).

En ambos casos, antes de iniciar la simulación propiamente dicha, se realiza una partición en el sistema original, donde es posible elegir qué parte del sistema se simulará con cada método. Además, para cada variable de estado que se simulará utilizando el enfoque clásico, se calcula simbólicamente sus términos correspondientes en la matriz Jacobiana, los cuales no se vuelve a calcular durante la simulación. Esto significa que los métodos clásicos solo se pueden usar en subsistemas donde el Jacobiano es constante, lo que limita la naturaleza de los sistemas que son adecuados para ser simulados con estos métodos.

7.4. Ejemplos y resultados

7.4.1. Convertidor Buck con disipador de calor

En esta sección, primero analizamos la limitación de rendimiento tanto de QSS como de los algoritmos clásicos al simular sistemas rígidos-discontinuos en los que la rigidez se debe a la presencia de un subsistema con una dinámica muy rápida en comparación con el resto del sistema. Se pueden encontrar ejemplos típicos de esto en sistemas físicos de varios dominios, ya que los componentes de diferentes dominios físicos usualmente involucran distintas constantes de tiempo.

Primero, presentaremos los resultados de ambos enfoques cuando simulemos un sistema de difusión de calor unidimensional. Este sistema está representado por un conjunto de ecuaciones diferenciales parciales (PDE), que se pueden transformar en el siguiente conjunto de EDOs mediante el uso del método de líneas,

$$\begin{aligned} \dot{u}_1 &= (300 - 2 \cdot u_1 + u_2) \cdot N \\ \dot{u}_N &= (298 - 2 \cdot u_N + u_{N-1}) \cdot N \\ \dot{u}_i &= (u_{i-1} - 2 \cdot u_i + u_{i+1}) \cdot N \quad where\ i = 2, N-1 \end{aligned} \tag{7.28}$$

donde u_i con $i = 1, N$ son las N variables de estado en las que se divide el sistema. En este caso, la temperatura ambiente es de $298K$ y la primera sección u_1 se calienta a una temperatura constante de $300K$. A partir de la Ec. (**??**), podemos observar que la matriz Jacobiana de este sistema tiene la forma

$$\begin{bmatrix} -2N & N & 0 & 0 & 0 & \cdots & 0 \\ N & -2N & N & 0 & 0 & \cdots & 0 \\ 0 & N & -2N & N & 0 & \cdots & 0 \\ \vdots & & & \ddots & & & \vdots \\ 0 & \cdots & \cdots & 0 & N & -2N & N \\ 0 & \cdots & \cdots & 0 & 0 & N & -2N \end{bmatrix} \tag{7.29}$$

Dado este tipo de estructura, los métodos implícitos clásicos no presentan inconvenientes, y el sistema se puede simular sin más problemas. En cambio, los métodos QSS (y, en particular, LIQSS) presentan oscilaciones espurias debido a la presencia de grandes entradas fuera de la diagonal principal de la matriz Jacobiana [23], que resultan en grandes costos computacionales. Por lo tanto, usar un método clásico es, por supuesto, una mejor opción en este escenario.

Por otra parte, el rendimiento de ambos enfoques cuando se simulan sistemas que presentan discontinuidades frecuentes ya se ha estudiado en [32] (en particular para circuitos de electrónica conmutada de potencia), donde las ventajas de usar métodos QSS estaban claramente expuestas. Entonces, si tuviéramos que estudiar un sistema que combinara ambos tipos de dinámica, ni los métodos clásicos ni los métodos QSS resultarían una buena opción por sí mismos. Una buena idea sería utilizar cada enfoque en la parte del sistema en la que resulten ventajosos.

Consideremos ahora el siguiente sistema: un convertidor Buck DC-DC, como se muestra en la Figura 7.1, que tiene un disipador de calor en su interruptor. Las pérdidas de energía en el interruptor se deben principalmente a dos razones. Hay pérdidas de conducción que son el producto de la tensión y la corriente en el interruptor en estados estacionarios, es decir, cuando está en su *estado encendido* (corriente alta, pero muy baja tensión) o cuando está en su *estado apagado* (alta tensión, pero muy baja corriente). Pero también hay pérdidas por conmutación, que tienen lugar durante pequeños períodos de tiempo cuando el interruptor pasa de un estado a otro, en los que tanto la tensión como la corriente tienen valores significativos. En nuestro ejemplo, estas pérdidas se modelan como pulsos de energía enviados al disipador de calor en cada conmutación durante los tiempos de encendido y de apagado.

El circuito será simulado utilizando el siguiente conjunto de parámetros.

- $U = 24\,V$ - tensión de entrada
- $C = 10\,mF$ - capacitor
- $L = 10\,mH$ - inductor
- $R = 10\,\Omega$ - resistencia de carga
- $f = 10\,kHz$ - frecuencia
- $DC = 0,35$ - ciclo de trabajo
- $R_{\text{off}} = 1 \cdot 10^5\,\Omega$ - resistencia de estado apagado
- $R_{\text{on}} = 1 \cdot 10^{-5}\,\Omega$ - resistencia de estado encendido
- $tol = 1 \cdot 10^{-3}$ - tolerancias absoluta y relativa

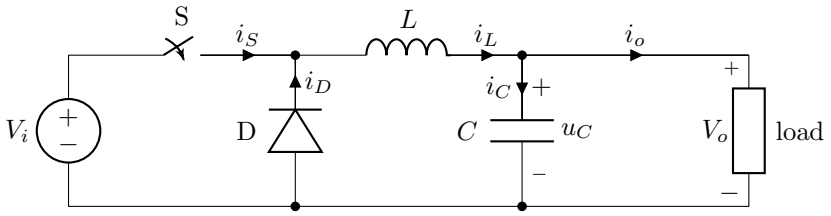


Figura 7.1: Convetidor Buck.

Primero, compararemos el uso de los métodos de integración clásicos con los de LIQSS y QMM (en ambos casos, usamos métodos de segundo orden). Para hacerlo, debemos simular este sistema durante un breve período de tiempo, ya que, como se mencionó anteriormente, los métodos clásicos son altamente ineficientes cuando se simulan circuitos de electrónica de potencia conmutada. Para ello, se simulará el sistema hasta un tiempo final de 2 segundos y el disipador de calor se dividirá en 100 secciones. Los resultados de estas simulaciones se pueden ver en la Tabla 7.1. Allí podemos observar que los métodos clásicos son claramente superados tanto por LIQSS como por QMM. Sin embargo, 2 segundos no son suficiente para que el disipador de calor alcance una temperatura estable. Por lo tanto, simularemos el mismo sistema hasta un tiempo final de 60 segundos y compararemos los resultados de LIQSS2 y QMM. En la Figura 7.2 podemos observar la comparación de

ambos enfoques cuando el número de secciones en que se divide el subsistema de disipación de calor. De allí podemos observar que cuando se usan 1000 secciones, QMM es más de 15 veces más rápido que LIQSS2.

Método de integración	**CPU [mseg]**
DASSL	52.520
CV ODE	27.795
IDA	57.436
LIQSS2	307
QMM2	245

Tabla 7.1: Convertidor Buck con disipador de calor.100 sections - $t_f = 2$ seg.

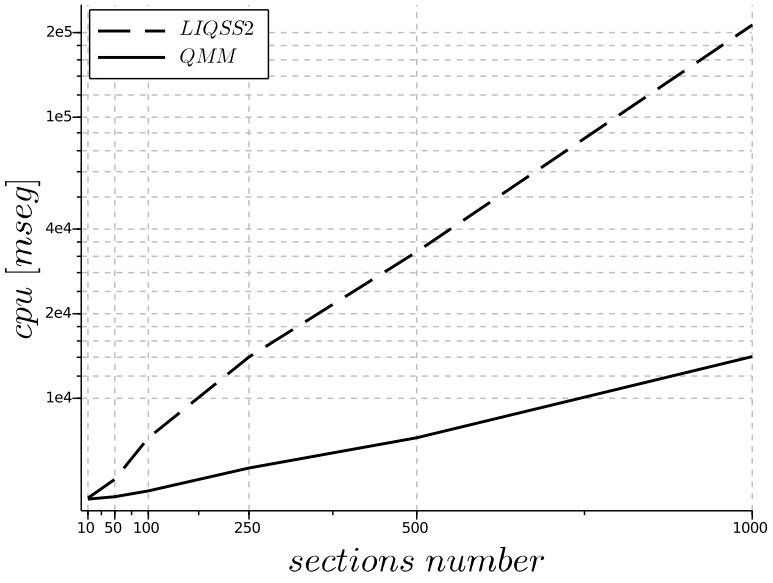


Figura 7.2: Buck DC-DC converter circuit.

7.4.2. Problema Advección-Reacción-Difusión (ADR) 1D

Las ecuaciones de advección-difusión proporcionan la base para describir fenómenos de transferencia de calor y masa, así como procesos de mecánica continua, donde la cantidad física de interés podría ser la temperatura en la conducción de calor o la concentración de alguna sustancia química. En varias aplicaciones, estos fenómenos ocurren en presencia de reacciones químicas, lo que lleva a la ecuación ADR, un problema que se encuentra con frecuencia en muchas áreas de las ciencias ambientales, así como en la ingeniería mecánica.

Los problemas de ADR discretizados con el método de líneas conducen a grandes sistemas rígidos de EDOs donde el uso de métodos LIQSS ha mostrado importantes ventajas sobre los algoritmos de tiempo discreto clásicos [35].

El siguiente conjunto de EDOs, tomado de [35], corresponde a la discretización espacial de un problema de ADR 1D:

$$\frac{\mathrm{d}u_i}{\mathrm{d}t} = -a \cdot \frac{u_i - u_{i-1}}{\Delta x} + d_i \cdot \frac{u_{i+1} - 2 \cdot u_i + u_{i-1}}{\Delta x^2} + r \cdot (u_i^2 - u_i^3)$$

para $i = 1, \ldots, N-1$ y

$$\frac{\mathrm{d}u_N}{\mathrm{d}t} = -a \cdot \frac{u_N - u_{N-1}}{\Delta x} + d_N \cdot \frac{2 \cdot u_{N-1} - 2 \cdot u_N}{\Delta x^2} + r \cdot (u_N^2 - u_N^3)$$

donde $N = 1000$ es el número de puntos de la malla, y $\Delta x = \frac{10}{N}$.

Consideramos parámetros $a = 1$, $r = 1000$,

$$d_i = \begin{cases} 10 & \text{if } i \in [1, N/10 - 1] \\ 1 \cdot 10^{-3} & \text{otherwise} \end{cases}$$

y condiciones iniciales $u_i(x, t = 0) = 0$.

Se puede observar que la primera décima parte tiene un coeficiente de difusión mucho mayor que el resto del sistema. Para esta primera parte del sistema, el uso de QSS no resulta conveniente, y el uso de un método clásico es una mejor opción. Pero para el resto del sistema, donde la difusión es menor, QSS supera los métodos clásicos. Por lo tanto, el uso de cada metodología aplicada convenientemente a cada subsistema da como resultado un mejor rendimiento de simulación, en comparación con el uso de un enfoque único para todo el sistema.

Este sistema se simuló hasta un tiempo final $t_f = 10$ utilizando los métodos clásicos DASSL, CVODE e IDA, así como LIQSS2 y QMM2. El

resultado de estas simulaciones se muestra en la Figura 5.1. Los resultados se informan en la Tabla 7.2, los resultados de IDA no se informaron porque la simulación con este método falló.

Método de integración	**CPU [mseg]**
DASSL	50.592
CV ODE	2.368
IDA	—
LIQSS2	1.538
QMM2	188

Tabla 7.2: Comparación de resultados para problema de Advección-Reacción-Difusión 1D.

7.5. Conclusiones

En este capítulo se presentaron los primeros algoritmos en combinar los métodos clásicos con los métodos por cuantificación de estados. La idea tras ello fue aprovechar las propiedades y beneficios que cada enfoque pueda tener a nivel de simulación para diferentes partes de un mismo sistema. Existen casos en que las diferentes dinámicas de los subsistemas que componen al sistema que se desea estudiar se comportan de manera muy distinta. Por tanto, puede resultar conveniente por eficiencia de simulación, o bien por cuestiones de estabilidad, simular un subsistema aplicando un método clásico, y otra subsistema utilizando algún miembro de la familia QSS.

Se propuso a nivel teórico el funcionamiento de un método mixto que combina ambos enfoques, y se demostraron sus principales propiedades teóricas, léase convergencia y estabilidad. Luego se describió la implementación de dos métodos, por un lado un método de primer orden, QMM1, que combina el algoritmo LIQSS1 con el método Backwar Euler, y luego un método de segundo orden, QMM2, que combina el algoritmo LIQSS2 con el método BDF2. En ambos casos el método clásico es implícito y cuentan con control de paso. La elección de métodos implícitos tanto en lo que respecta a la metodología clásica como a la de estados cuantificados responde al interés que supone la simulación eficiente de sistemas stiff.

Finalmente, a través de dos ejemplos se mostró el desempeño de esta metodología. Para ello, se utilizó el algoritmo de segundo orden QMM2. En primera instancia se simuló un sistema multidominio, que mezcla el dominio

electrónico y el térmico. Aquí se optó por simular el subsistema electrónico usando LIQSS2 y el subsistema térmico usando BDF2. Luego se simuló un problema de advección-difusión-reacción unidimensional discretizado utilizando el método de lineas. En el mismo el coeficiente de difusión es variable, de modo que se obtienen dos subsistemas con dinámicas distintas. En ambos ejemplos puede verse las ventajas de utilizar el enfoque de métodos mixtos por sobre la elección de cualquier de las anteriores metodologías por si solas.

Capítulo 8

Trabajo en curso, futuro y conclusiones

Como se mencionó en la introducción, esta tesis puede ser vista como una contribución al desarrollo de métodos de integración por cuantificación de ecuaciones diferenciales ordinarias resolviendo los problemas que los mismos tenían en la simulación de sistemas stiff.

Teniendo en cuenta el vasto estudio existente sobre los métodos de integración de tiempo discreto para sistemas stiff y sus años de desarrollo, los métodos presentados en esta tesis no pretenden mejorar todos los resultado obtenidos con los mismos en sólo 4 años de desarrollo.

Simplemente se buscó dar solución a algunos de los problemas presentes en los métodos de integración por cuantificación QSS e intentar mejorar los resultados obtenidos por los métodos clásicos en algunos casos particulares. Puntualmente, se mejoraron sensiblemente los resultados de los métodos clásicos en la simulación de sistemas stiff discontinuos, particularmente los resultantes del modelado de circuitos de electrónica conmutada. Además, se dejaron abiertas nuevas puertas para poder ampliar el espectro de problemas en los cuales estos métodos podrían encontrar soluciones más eficientes que los métodos clásicos.

Teniendo en cuenta estas observaciones, este último capítulo comenzará enumerando problemas aún no resueltos. Finalmente, se presentarán las conclusiones generales de la tesis.

8.1. Problemas abiertos y trabajo futuro

Como se dijo en los capítulos 4 y 5, los métodos modificados propuestos puede resolver eficientemente la simulación de sistemas stiff donde se cumplen una de dos condición posibles:

1. Cuando la rigidez se debe sólo a términos grandes en la diagonal principal de la matriz Jacobiana del sistema. Este es el caso que los métodos LIQSS originales podían tratar correctamente.

2. Cuando la rigidez del sistema se debe a la presencia de subsistemas de 2×2 desacoplados unos de otros. Es decir, cuando las oscilaciones presentes en los métodos LIQSS se deben a la interacción entre pares de variables.

Este segundo caso es el que fue agregado mediante el desarrollo de los métodos mLIQSS. Como vimos, sistemas con ese tipo de estructuras aparece con frecuencia en la práctica, fundamentalmente en las aplicaciones de electrónica de potencia.

Sin embargo, existen casos donde la rigidez del sistema tiene lugar debido a la interacción de una cierta cantidad m de variables, donde $m > 2$. En estos casos, y como vimos en la sección 6.4, estos métodos fallan.

Luego, en el capítulo 7 se planteó la utilización de métodos mixtos para la simulación de ciertos sistemas. Sin embargo, en el caso en que existan un subsistema de $m \times m$ variables que sean causal de rigidez dentro de un sistema de $n \times n$ (donde $n > m$), y donde además esas m variables presenten discontinuidades frecuentes, los métodos QMM tampoco lograran integrarlo eficientemente. Esto se debe a que el hecho de que esas m variables presenten discontinuidades frecuentes indica que lo más conveniente sería simular ese subsistema con algún método de la familia QSS, sin embargo, vimos que siendo $m > 2$, ningún miembro de la familia podrá integrar dicho subsistema de manera eficiente. Queda entonces abierto el planteo de cómo proceder en estos casos.

Por otro lado, en lo que respecta a los métodos mistos, actualmente el particionado de los subsistemas para determinar qué subsistema se integrará con cada metodología es manual. Resta entonces determinar cómo automatizar este proceso de manera eficiente, o bien sea *offline* al comienzo de la simulación, o bien *online*, durante la ejecución de la simulación, lo cual sugiere que un mismo subsistema podría conmutar entre metodología según resulte conveniente.

Una primera idea, desde el enfoque *offline*, es realizar un análisis a nivel del modelo, para determinar qué derivadas de variables de estado $dotx_i$

dependen de variables de estado x_j involucradas en discontinuidades. Este subconjunto e variables, tanto x_i como x_j sería elegido para ser integrado usando un método QSS. Esta elección tiene su fundamento en las ya mencionadas ventajas de estos métodos para integrar a través de discontinuidades respecto a los métodos clásicos.

8.2. Conclusiones Generales

Si bien la mayor parte de las conclusiones se encuentran al final de cada capítulo, se debe destacar a modo de síntesis que esta tesis presentó una contribución al desarrollo de métodos de integración cuantificados y a su vez nuevas herramientas para la integración numérica de sistemas stiff.

Se propusieron cuatro métodos de aproximación llamados mLIQSS1, mLIQSS2, QMM1 y LIQSS2 (dos de primer orden y otros dos de segundo respectivamente). En particular, el desarrollo de los mismos basado en los principios de los métodos implícitos y linealmente implícitos permitió que estos métodos resulten muy eficientes en la simulación de sistemas stiff.

Como pudo verse, los métodos mLIQSS extienden el tipo de rigidez que los métodos LIQSS originales pueden integrar de manera eficiente. El tipo de rigidez que los mismos pueden ingrar se encuentra en diversos campos de aplicación. Resultan particularmente eficientes en la simulación de circuitos de electrónica conmutada, incluso en el caso de simular en presencia de elementos parásitos. Esto permite realizar un estudio muy útil desde el punto de vista del diseño de protecciones de elementos de los elementos de conmutación.

Pude verse también, que como era de esperarse, las ventajas de lo métodos mLIQSS por sobre los métodos clásicos, se incrementan a medida que el sistema a simular crece en tamaño. Esto es por dos motivos, por un lado, si el sistema a simular presenta discontinuidades frecuentes, al incrementar el tamaño del sistema, aumentará el número de discontinuidades presentes. Como vimos, cualquier método QSS se ve favorecido por esta situación, y los mLIQSS no son la excepción. Por otro lado, al tratarse de sistemas stiff, estos deben ser integrados usando métodos implícitos. Por ello, a medida que el sistema crece, la matriz a invertir por los métodos también crece, incrementando así mucho el costo computacional de los algoritmos clásicos.

Finalmente, en esta tesis se presentó por primera vez un algoritmo que combina los métodos clásicos con los métodos por cuantificación de estados. La idea tras ello fue aprovechar las propiedades y beneficios que cada enfoque pueda tener a nivel de simulación para diferentes partes de un mismo

sistema. Existen casos en que las diferentes dinámicas de los subsistemas que componen al sistema que se desea estudiar se comportan de manera muy distinta. Por tanto, puede resultar conveniente por eficiencia de simulación, o bien por cuestiones de estabilidad, simular un subsistema aplicando un método clásico, y otra subsistema utilizando algún miembro de la familia QSS.

Bibliografía

[1] E. Hairer, S. Nørsett, and G. Wanner, *Solving Ordinary Dfferential Equations I. Nonstiff Problems.* Berlin: Springer, 2nd ed., 1993.

[2] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling, "Numerical recipes.," *Cambridge University Press*, 1986.

[3] L. Shampine and M. Reichelt, "The matlab ode suite.," *SIAM Journal on Scientific Computing*, vol. 18, no. 1, pp. 1–22, 1997.

[4] F. Cellier and E. Kofman, *Continuous System Simulation.* New York: Springer, 2006.

[5] E. Kofman and S. Junco, "Quantized State Systems. A DEVS Approach for Continuous System Simulation," *Transactions of SCS*, vol. 18, no. 3, pp. 123–132, 2001.

[6] E. Kofman, "A Second Order Approximation for DEVS Simulation of Continuous Systems," *Simulation: Transactions of the Society for Modeling and Simulation International*, vol. 78, no. 2, pp. 76–89, 2002.

[7] E. Kofman, "A Third Order Discrete Event Simulation Method for Continuous System Simulation," *Latin American Applied Research*, vol. 36, no. 2, pp. 101–108, 2006.

[8] E. Kofman, "Discrete Event Simulation of Hybrid Systems," *SIAM Journal on Scientific Computing*, vol. 25, no. 5, pp. 1771–1797, 2004.

[9] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems.* Berlin: Springer, 1991.

[10] F. Cellier and E. Kofman, *Continuous System Simulation.* New York: Springer, 2006.

[11] Lambert, *Numerical Methods for Ordinary Differential Systems. The Initial Value Problem*. 1991.

[12] J. Butcher, "The numerical analysis of ordinary differential equations: Runge-kutta and general linear methods," *Wiley,Chichester*, 1987.

[13] H. Rosenbrock, "Some general implicit processes for the numerical solution of differential equations.," *Computer Journal*, 1963.

[14] S. Nørsett and A. Wolfbrandt, "Order conditions for rosenbrock type methods," *Numerische Mathematik*, no. 32, pp. 1–15, 1979.

[15] P. Kaps, S. Poon, and T. Bui, "Rosenbrock methods for stiff odes: A comparison of richardson extrapolation and embedding technique," *Computing*, no. 34, pp. 17–40, 1985.

[16] P. Kaps and G. Wanner, "A study of rosenbrock-type methods of high order," *Numerische Mathematik*, no. 38, pp. 279–298, 1981.

[17] V. Savcenco, *Multirate Numerical Integration for Ordinary Differential Equations*. PhD thesis, Universiteit van Amsterdam, 2007.

[18] R. Savcenco, V.and Mattheij, "A multirate time stepping strategy for stiff ordinary differential equations," *BIT Numerical Mathematics*, vol. 47, pp. 137–155, 2007.

[19] B. Zeigler and J. Lee, "Theory of quantized systems: formal basis for DEVS/HLA distributed simulation environment," in *SPIE Proceedings*, pp. 49–58, 1998.

[20] B. Zeigler, T. G. Kim, and H. Praehofer, *Theory of Modeling and Simulation. Second edition*. New York: Academic Press, 2000.

[21] E. Kofman, "Relative error control in quantization based integration," *Proceedings of RPIC'07*, 2007.

[22] F. Di Pietro, G. Migoni, and E. Kofman, "Improving a Linearly Implicit Quantized State System Method," in *Proceedings of the 2016 Winter Simulation Conference*, (Arlington, Virginia, USA), 2016.

[23] F. D. Pietro, G. Migoni, and E. Kofman, "Improving linearly implicit quantized state system methods," *Simulation: Transactions of the Society for Modeling and Simulation International*.

[24] F. Di Pietro, E. Kofman, and M. Romero, "Simulación Eficiente de Circuitos de Electrónica Conmutada en Presencia de Elementos Parásitos," in *Proceedings of AADECA 2018*, (Buenos Aires, Argentina), 2018.

[25] J. Nutaro and B. Zeigler, "On the Stability and Performance of Discrete Event Methods for Simulating Continuous Systems," *J. Computational Physics*, vol. 227, no. 1, pp. 797–819, 2007.

[26] G. Migoni, M. Bortolotto, E. Kofman, and F. Cellier, "Linearly Implicit Quantization-Based Integration Methods for Stiff Ordinary Differential Equations," *Simulation Modelling Practice and Theory*, vol. 35, pp. 118–136, 2013.

[27] F. Bergero and E. Kofman, "PowerDEVS. A Tool for Hybrid System Modeling and Real Time Simulation," *Simulation: Transactions of the Society for Modeling and Simulation International*, vol. 87, no. 1–2, pp. 113–132, 2011.

[28] T. Beltrame and F. E. Cellier, "Quantised state system simulation in dymola/modelica using the devs formalism," in *Proceedings 5th International Modelica Conference*, pp. 73–82, 2006.

[29] M. C. D'Abreu and G. A. Wainer, "M/cd++: modeling continuous systems using modelica and devs," in *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2005. 13th IEEE International Symposium on*, pp. 229–236, IEEE, 2005.

[30] G. Quesnel, R. Duboz, É. Ramat, and M. K. Traoré, "Vle: a multimodeling and simulation environment," in *Proceedings of the 2007 summer computer simulation conference*, pp. 367–374, Society for Computer Simulation International, 2007.

[31] J. Fernández and E. Kofman, "A Stand-Alone Quantized State System Solver for Continuous System Simulation.," *Simulation: Transactions of the Society for Modeling and Simulation International*, vol. 90, no. 7, pp. 782–799, 2014.

[32] G. Migoni, F. Bergero, E. Kofman, and J. Fernández, "Quantization-Based Simulation of Switched Mode Power Supplies.," *Simulation: Transactions of the Society for Modeling and Simulation International*, vol. 91, no. 4, pp. 320–336, 2015.

[33] R. Assar and D. J. Sherman, "Implementing biological hybrid systems: Allowing composition and avoiding stiffness," *Applied Mathematics and Computation*, vol. 223, pp. 167–179, 2013.

[34] J. J. Tyson, "Modeling the cell division cycle: cdc2 and cyclin interactions.," *Proceedings of the National Academy of Sciences*, vol. 88, no. 16, pp. 7328–7332, 1991.

[35] F. Bergero, J. Fernández, E. Kofman, and M. Portapila, "Time Discretization versus State Quantization in the Simulation of a 1D Advection-Diffusion-Reaction Equation.," *Simulation: Transactions of the Society for Modeling and Simulation International*, vol. 92, no. 1, pp. 47–61, 2016.

[36] L. R. Petzold, "Description of dassl: a differential/algebraic system solver," tech. rep., Sandia National Labs., Livermore, CA (USA), 1982.

[37] A. C. Hindmarsh, R. Serban, and D. R. Reynolds, "User documentation for cvode v3. 1.1 (sundials v3. 1.1)," 2018.

[38] A. C. Hindmarsh, R. Serban, and A. Collier, "User documentation for ida v3. 1.1 (sundials v3. 1.1)," 2018.

[39] A. Elbanhawy, "Effect of parasitic inductance on switching performance," *Proc. PCIM Europe 2003*, pp. 251–255, 2003.

[40] B. Yang and J. Zhang, "Effect and utilization of common source inductance in synchronous rectification," in *Applied Power Electronics Conference and Exposition, 2005. APEC 2005. Twentieth Annual IEEE*, vol. 3, pp. 1407–1411, IEEE, 2005.

[41] S. x Bhat and H. Nagaraja, "Effect of parasitic elements on the performance of buck-boost converter for pv systems," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 4, no. 6, pp. 831–836, 2014.

[42] B. P. Zeigler, A. Muzy, and E. Kofman, *Theory of modeling and simulation*. Elsevier, 2018.