Simulación por Cuantificación de Sistemas Stiff.

Gustavo Migoni

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

Doctor en Ingeniería

Director: Ernesto Kofman

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

Resumen

En esta Tesis se desarrollan nuevos métodos de integración para ecuaciones diferenciales ordinarias. Siguiendo la idea de los métodos de integración basados en cuantificación, es decir, reemplazando la discretización temporal por la cuantificación de los estados, estos nuevos métodos realizan aproximaciones de primer, segundo y tercer orden. Si bien se utilizan algunos principios de los métodos clásicos implícitos, los algoritmos desarrollados son explícitos en su implementación y pueden ser utilizados para la simulación eficiente de diversas clases de sistemas stiff. Así, estos nuevos métodos son los primeros de su tipo que resultan adecuados para el tratamiento de sistemas rígidos.

Se demuestra en el trabajo que estos métodos satisfacen las mismas propiedades teóricas que los métodos de integración basados en cuantificación previamente existentes. Además, se detalla cómo los algoritmos se traducen en una especificación de eventos discretos (DEVS) y se describe su implementación en una herramienta de software de simulación adecuada.

La eficiencia de estos métodos es verificada haciendo comparaciones entre ejemplos simulados con los mismos y con distintas herramientas avanzadas de integración numérica. En particular, se destacan las ventajas que poseen los algoritmos desarrolados con respecto a los métodos clásicos de integración en la simulación de circuitos de electrónica de conmutación.

Agradecimientos

Quiero agradecer en primer lugar a mi director de tesis Ernesto Kofman no solo por su ayuda a la hora de sacar adelante esta Tesis Doctoral sino también por acompañarme en aspectos de la vida que van más allá de lo laboral y profesional. Sin el tiempo que dedicó a orientarme, ayudarme y animarme, este trabajo difícilmente hubiera concluido satisfactoriamente.

Siguiendo con la gente del ambiente académico, tengo que agradecerle a uno de los responsables de que actualmente me este dedicando a la docencia y uno de los que me alentó a realizar este doctorado. Sergio Junco, además, de apoyarme durante el doctorado, le dio un poco de humor a cada día con sus chistes diarios.

No se me debe pasar por alto agradecer a mis campaneros de oficina Mario, Franco, Federico, y Fernando por el apoyo y ayuda diario que me brindaron

Seria imposible dejar de agradecerles Flavia y Alejandro por las infinitas charlas de café y momentos compartidos mientras realizábamos nuestros respectivos Doctorados. Creo que de alguna forma, que nada tiene que ver con lo estrictamente académico hicieron posible que llegase hasta este punto en la carrera.

Yendo aun mas hacia el plano personal, jamas hubiese sido posible estar hoy escribiendo esta Tesis sin el apoyo incondicional de mis padres y esposa, que siempre me alentaron a seguir cualquiera fuese el proyecto que emprendiera, enseñándome la importancia de la libertad de elegir.

Finalmente, y no por eso menos importante, debo agradecer a mi perra Afri por estar siempre junto a mi cuando tuve que trabajar en casa y mientras escribía esta Tesis.

Se que hay mucha gente mas a quien agradecer el apoyo que me brindaron en el transcurso del doctorado pero podría escribir un capitulo más de esta tesis y aun así seguiría quedándome gente sin agradecer.

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), [12, 37, 43]. 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 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 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 $\lambda \cdot h$ [8], 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* en términos del formalismo DEVS [45].

El primero de estos métodos es el de QSS1 (Quantized State Systems) [27], que realiza una aproximación de primer orden. En base a principios similares, se desarrollaron también métodos de segundo orden (QSS2) [26] y de tercer orden

(QSS3) [20], 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 [18]. Sin embargo, no son apropiados para la simulación de sistemas stiff, debido a la aparición de oscilaciones de alta frecuencia [8]. 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.

En esta Tesis se desarrollan, 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. Estos métodos, aunque fueron formulados para EDOs pueden ser también aplicados a sistemas híbridos y a *Ecuaciones Diferenciales Algebraicas*.

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 la simulación de sistemas discontinuos stiff. En particular, se mostrará cómo estos métodos obtienen ventajas sobre los métodos tradicionales en la simulación modelos de circuitos electrónica conmutada cuando en los mismos se modela con más realismo los elementos de conmutación. Estos tipos de modelos resultan ser stiff y presentan en general muchas discontinuidades.

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 segundo capítulo 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 seccion 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.

El tercer capitulo, presenta las ideas principales en las que se basa el resto del trabajo. En las secciones 3.1 y 3.2 se presentan los conceptos originales de cuantificación y DEVS comenzando con un ejemplo motivador, luego se presenta una teoría ad-hoc sobre DEVS y finalmente se muestra la relación entre DEVS y el ejemplo motivador para introducir el concepto de Sistemas Cuantificados. La sección 3.3 presenta entonces los *Sistemas de Estados Cuantificados* (QSS, por Quantized State Systems) y los métodos QSS, siendo estos los que dieron origen a los métodos que se desarrollaron en este trabajo. Además, se presenta en esta sección un software de simulación llamado PowerDEVS [3] (concebido como un simulador DEVS de propósito general) en el cual están implementados

los métodos QSS y los métodos que se desarrollan en esta Tesis. Finalmente, en la seccion 3.4 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.

El capítulo 4 comienza mostrando, a través de un ejemplo, los inconvenientes que tienen los métodos QSS en la simulación de sistemas Stiff. A partir de este punto, el resto de la Tesis contiene los resultados originales del trabajo. En la sección 4.2 se presenta el método BQSS (Backward QSS), siendo este el primer método de integración de eventos discretos que permite integrar de manera eficiente sistemas stiff. 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 4.2.2. Luego, en la sección 4.2.3 se deduce el modelo DEVS de un integrador BQSS reutilizando parte de los conceptos de la sección 3.3.

El desarrollo del método BQSS termina en la sección 4.3 donde se presentan las propiedades teóricas del método. Se realiza aquí un estudio de estabilidad y cota de error basado en una representación perturbada de los modelos obtenidos aplicando el método BQSS. Al igual que en los métodos QSS, el estudio de cota de error arroja un resultado que permite a su vez, en el caso de sistemas LTI, elegir la cuantificación apropiada de acuerdo a la precisión deseada. El principal resultado en este punto concluye que –bajo ciertas condiciones– las soluciones obtenidas mediante este método están finalmente acotadas. Finalmente, en la sección 4.4 se presentan dos ejemplos que permiten comparar los resultados obtenidos con BQSS con los que se obtienen utilizando algunos de los métodos clásicos introducidos en el capítulo 2.

Un resultado colateral de esta Tesis surgió al combinar el método BQSS con el método QSS. El método que surgió de esta manera se llamo CQSS (Centered QSS) y resulta ser el primer método general de estados cuantificados apropiado para la simulación de sistemas marginalmente estables. En la sección 4.5 se presenta la idea de este método y se realiza un estudio muy simplificado de sus propiedades. Luego en la sección 4.6 se presentan dos ejemplos de sistemas marginalmente estables simulados utilizando este método. Si bien los resultados obtenidos parecen muy prometedores, el estudio de métodos para sistemas marginalmente estables utilizando métodos de integración por cuantificación esta siendo llevado a cabo por otro integrante del grupo, escapando el análisis más profundo de este método de los temas de esta Tesis.

A pesar de las bondades mostradas por el método BQSS, el estudio teórico concluye que no puede alcanzarse una buena precisión sin aumentar significativamente el número de pasos. Por esto se torna necesario formular aproximaciones de orden superior.

Lamentablemente no se pudo generar métodos BQSS de orden superior sin alterar la idea principal del mismo y se observaron algunos problemas en la simulación de algunos sistemas no lineales relacionados con la aparición de falsos puntos de equilibrio. Sin embargo, combinando la idea del método BQSS con los principios de los métodos linealmente implícitos mostrados en la sección 2.4, se logró desarrollar un método denominado LIQSS (QSS Linealmente Implícito) del cual sí se pudo generar versiones de orden mayor.

En el capítulo 5 se presentan finalmente los métodos LIQSS que represen-

tan el resultado más importante de esta Tesis. En primer lugar se introduce en la sección 5.1 el primer método de esta familia denominado LIQSS1. El mismo es un método de primer orden que se obtuvo introduciendo una pequeño cambio en el método BQSS inspirado en los métodos linealmente implícitos. La sección comienza con un ejemplo introductorio que muestra de manera intuitiva el comportamiento de este método para dar lugar luego a la definición formal del mismo. Finalmente, aprovechando que la forma de las trayectorias en este método resultan iguales que las de QSS1, se deduce el modelo DEVS correspondiente.

En base a este nuevo enfoque, en las secciones 5.2 y 5.3 se generaliza la idea mostrada en la seccion anterior y, utilizando los principios de los métodos QSS2 y QSS3 se obtienen los métodos LIQSS2 y LIQSS3 de segundo y tercer orden respectivamente. En cada caso, se comienza dando una idea intuitiva de como funcionan, a continuación se da la definición formal y finalmente se deduce los modelos DEVS correspondientes.

Una vez que los métodos han sido completamente definidos y formalizados, la sección 5.4 se dedica al estudio de las propiedades teóricas de los mismos. En primer lugar se demuestra la legitimidad de los métodos, propiedad que garantiza que las simulaciones nunca se "trabarán". Luego, siguiendo la idea de las demostraciones correspondientes a los métodos de QSS, se hace un análisis de estabilidad y cota de error de los LIQSS. Finalmente, en la sección 5.5 se muestran dos ejemplos de simulación de sistemas continuos (uno lineal y otro no lineal). En los mismos se pone en evidencia en forma práctica los ordenes de los métodos y se muestran sus virtudes comparando los resultados obtenidos con los correspondientes a métodos de tiempo discretos apropiados para el tipo de sistemas mostrados.

A modo de resumen de los resultados presentados en este capítulo, la sección 5.6 presenta a modo de conclusiones, una descripción compacta de las ventajas y limitaciones de estos métodos así como también el tipo de modelos en los cuales utilizar estos métodos resultaría realmente provechoso.

El capitulo 6 se dedica exclusivamente a la simulación de modelos de circuitos de electrónica conmutada. Estos tipos de modelos presentan la particularidad de ser stiff y discontinuos, característica que hace que el uso de los métodos LIQSS para su simulación logre una mejora realmente importante con respecto a los demás métodos.

La Tesis concluye con el capítulo 7, donde se presentan algunos problemas abiertos en los que se está trabajando actualmente y las conclusiones generales del trabajo realizado.

Al final de la Tesis, pueden encontrarse los apéndices con los códigos de los métodos que se desarrollaron e implementaron en PowerDEVS.

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: construcción de los modelos DEVS, estudio de las propiedades teóricas, implementación en el entorno de simulación PowerDEVS,

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, segundo y tercer orden permitiendo simular eficientemente diversos tipos de sistemas stiff. La característica saliente de los métodos desarrollados es que son explícitos, a diferencia del resto de los algoritmos apropiados para simular este tipo de sistemas.

Una contribución colateral fue el desarrollo de un método de primer orden que permite simular correctamente sistemas marginalmente estables

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 [13] y [8].

Los más usados en la práctica son aquellos basados en métodos lineales multipaso, especialmente los métodos BDF [28, 12], por ser muy eficientes para un gran número de problemas de este tipo y los métodos Runge-Kutta implícitos [6], 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 [39] así como los métodos Rosenbrock-Wanner o métodos Rosenbrock modificados, [35, 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 [42, 41]. También existen métodos denominados mixtos, que utilizan fórmulas explícitas para ciertas componentes del sistema y fórmulas implícitas para las restantes [8].

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 [46]. 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 [45].

El primero de estos métodos es el de QSS1 (Quantized State Systems) [27], que realiza una aproximación de primer orden. En base a principios similares, se desarrollaron también métodos de segundo orden (QSS2) [26] y de tercer orden (QSS3) [20], 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 [18]. Sin embargo, no son apropiados para la simulación de sistemas stiff, debido a la aparición de oscilaciones de alta frecuencia [8]. 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 del trabajo de Ernesto Kofman en el tema. Aquí, se señala el problema que presentan los métodos para simular sistemas stiff y se soluciona y el método BQSS resulta entonces el primer algoritmo de integración de ecuaciones diferenciales por eventos discretos que permite simular este tipo de sistemas. Si bien la idea básica ya había sido planteada en [17], nunca había sido formalizada ni implementada.

Además de ser el primer algoritmo tipo backward de la familia de QSS, BQSS tiene la particularidad de no requerir ningún tipo de iteraciones para su implementación. Sin embargo, las limitaciones en precisión impuestas por el bajo orden del método llevaron al desarrollo de los métodos LIQSS.

Una característica heredada de los métodos QSS, es que los métodos de BQSS, LIQSS1, LIQSS2 y LIQSS3 demuestran muy buen desempeño la simulación de sistemas híbridos stiff. Estos casos han sido siempre un problema para los métodos clásicos de tiempo discreto. Por un lado, se conocen las trayectorias del sistema durante todo el tiempo entre dos muestras. Por otro lado, los métodos son asincronicos y aceptan eventos en cualquier instante de tiempo. En consecuencia no hay que modificar nada para tener en cuenta los eventos en el tiempo en que ocurren. Esta cualidad junto con las virtudes de los métodos en la simulación de sistemas stiff hace de estos métodos una excelente opción para la simulación de circuitos de electrónica de conmutación cuando se les da a los modelos de los componentes discontinuos cierto grado de realismo.

Al igual que los métodos QSS, estos métodos tienen características sobresalientes en la manera en que explotan la ralitud. El aprovechamiento de la ralitud de un sistema es punto un punto muy importante cuando los sistemas

son además stiff. En este campo, la comunidad dedicada a problemas de simulación está haciendo mucho esfuerzos para sacar provecho de la ralitud. Por un lado las herramientas como Matlab [43] y Dymola [40] tratan de explotar estas estructuras para tenerlas en cuenta en cada inversión y multiplicación matricial, y los métodos numéricos también buscan sacar provecho de estas propiedades [36].

Por último, cabe mencionar que un resultado colateral que surgió del trabajo de esta Tesis es un método de integración por cuantificación para sistemas Marginalmente Estables. Este método se denominó CQSS y abre un nuevo horizonte de posibilidades para estos métodos.

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 BQSS, la construcción del modelo DEVS y la demostración de las propiedades generales de estabilidad y legitimidad. Estos resultados fueron publicados en primer lugar en una conferencia local [23, 32] y luego en una revista internacional [33] y en el capítulo de un libro de una prestigiosa editorial internacional [44].

Posteriormente surgió como resultado colateral el algoritmo CQSS que sirve de semilla para una nueva familia de métodos para sistemas marginalmente estables. Los resultados sobre el mismo, junto con los de BQSS se encuentran publicados en una conferencia local [5], en una conferencia internacional [7] y forman parte de un capítulo de coautoria de un libro [44].

Ambos resultados (BQSS y CQSS) con sus principales propiedades teóricas forman parte de un artículo actualmente en revisión en la revista *Simulation. Transactions of the SCS*, estando su aceptación sujeta a modificaciones sugeridas por los revisores.

El siguiente paso fue obtener métodos de orden superior para la simulación de sistemas stiff. Si bien en esta Tesis se presentan algoritmos LIQSS de hasta tercer orden, se publicaron resultados de los mismos sólo hasta orden dos en una conferencia local [29, 30] y luego en una revista internacional [31].

Hay también un artículo en preparación para una revista internacional sobre los métodos LIQSS de hasta tercer orden aplicados a la simulación de circuitos de electrónica de conmutación.

Capítulo 2

Metodos Clásicos de Integración y Sistemas Stiff

En este capítulo, basado principalmente en [8], 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 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

Evidentemente, 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.

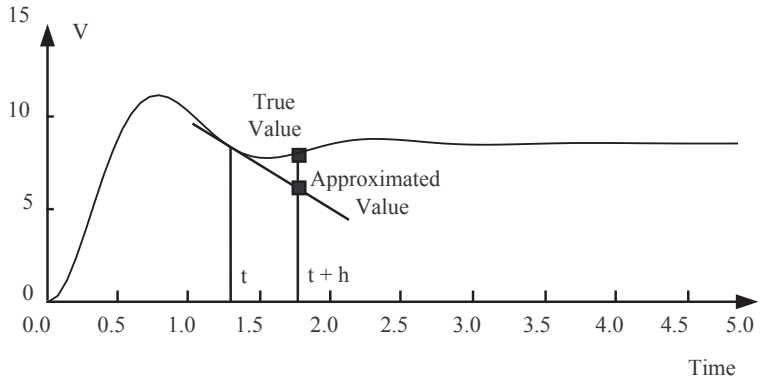

Figura 2.1: Integración numérica utilizando Forward Euler.

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*.

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.

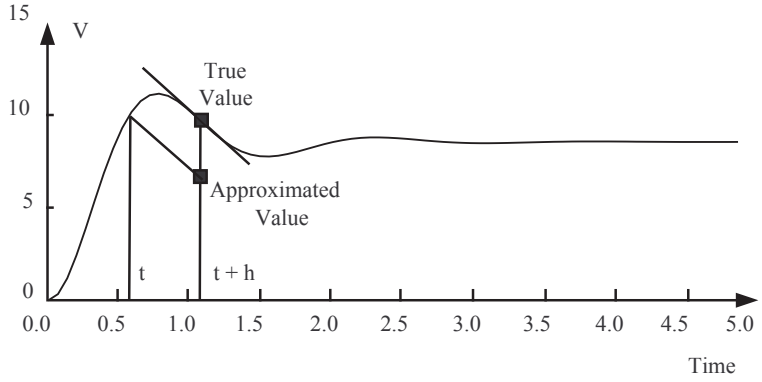

Figura 2.2: Integración numérica utilizando Backward Euler.

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:

$$\mathbb{R}\mathrm{e}\{\mathrm{Eig}(\mathbf{A})\} = \mathbb{R}\mathrm{e}\{\lambda\} < 0{,}0 \tag{2.9}$$

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

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:

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

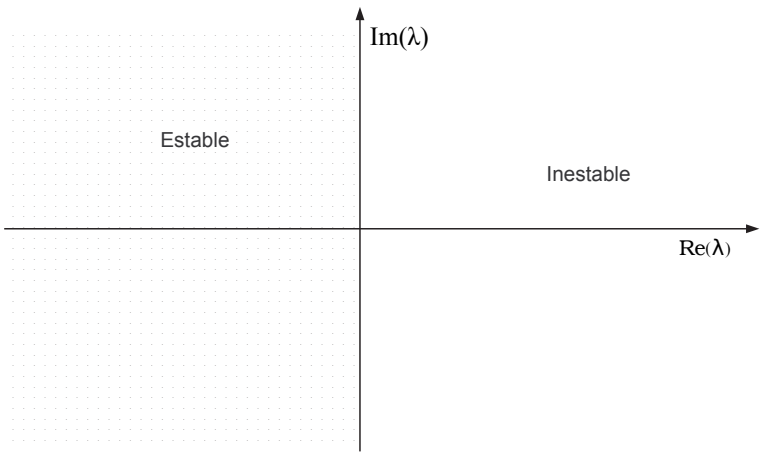

Figura 2.3: Dominio de estabilidad analítica.

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 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.

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:

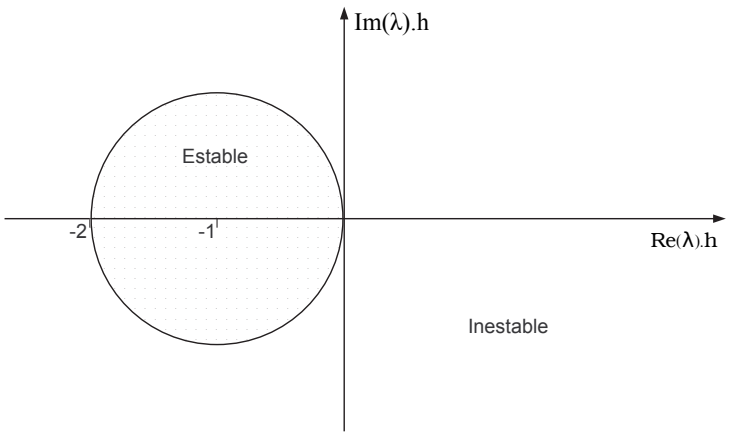

Figura 2.4: Dominio de estabilidad numérica de Forward Euler.

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

o:

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

Luego:

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

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

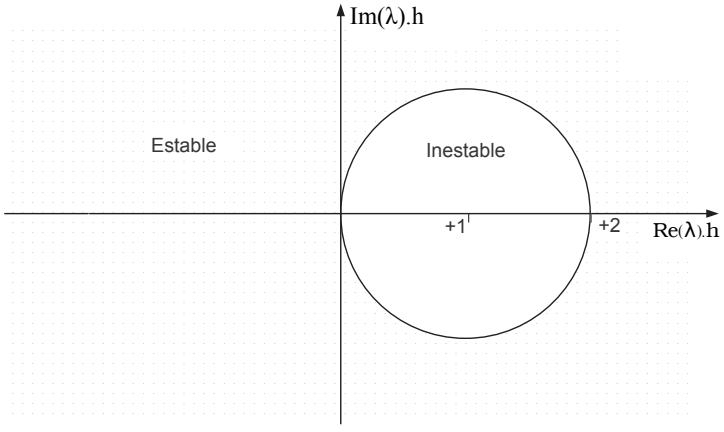

Figura 2.5: Dominio de estabilidad de Backward Euler.

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

con autovalores cuyas partes reales están desparramadas a lo largo del eje real negativo.

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

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

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.

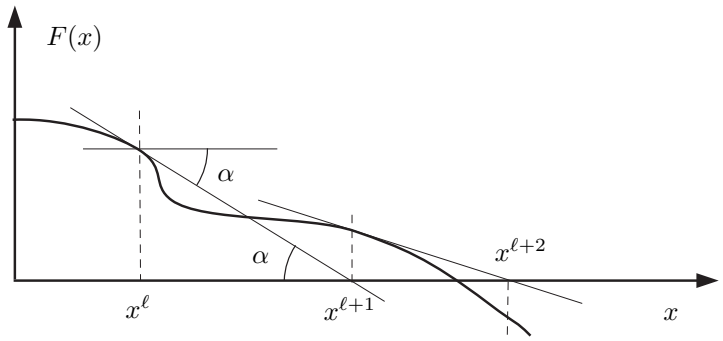

Figura 2.6: Iteración de Newton.

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

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

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

$$x_{k+1}^{\ell+1} = x_{k+1}^{\ell} - \frac{x_k + h \cdot f(x_{k+1}^{\ell}, t_{k+1}) - x_{k+1}^{\ell}}{h \cdot \partial f(x_{k+1}^{\ell}, t_{k+1})/\partial x - 1{,}0} \tag{2.24}$$

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:

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

donde:

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

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:

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

donde:

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

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:

etapa 0: $\mathbf{k_1} = \mathbf{f}(\mathbf{x_k} + b_{1,1} \cdot h \cdot \mathbf{k_1} + \ldots + b_{1,n} \cdot h \cdot \mathbf{k_n}, t_k + a_1 h)$

$\vdots$ $\vdots$

etapa $n-1$: $\mathbf{k_n} = \mathbf{f}(\mathbf{x_k} + b_{n,1} \cdot h \cdot \mathbf{k_1} + \ldots + b_{n,n} \cdot h \cdot \mathbf{k_n}, t_k + a_n h)$

etapa n: $\mathbf{x_{k+1}} = \mathbf{x_k} + c_1 \cdot h \cdot \mathbf{k_1} + \ldots + c_n \cdot h \cdot \mathbf{k_n}$

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	$b_{1,1}$	$\ldots$	$b_{1,n}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
a_n	$b_{n,1}$	$\ldots$	$b_{n,n}$
x	c_1	$\ldots$	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,

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

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

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

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

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

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.

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

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

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 marginal se cierra sobre el semiplano izquierdo del plano $(\lambda \cdot h)$.

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

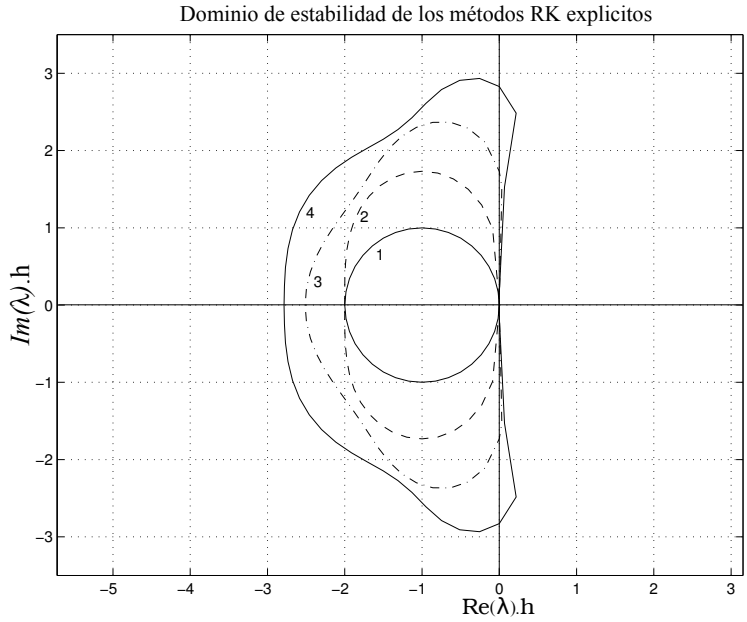

Figura 2.7: Dominios de estabilidad de los métodos explícitos RK.

hacerse más pequeño que el valor indicado por la estima del error local debido a las restricciones impuestas por la región de estabilidad numérica'.

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

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

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

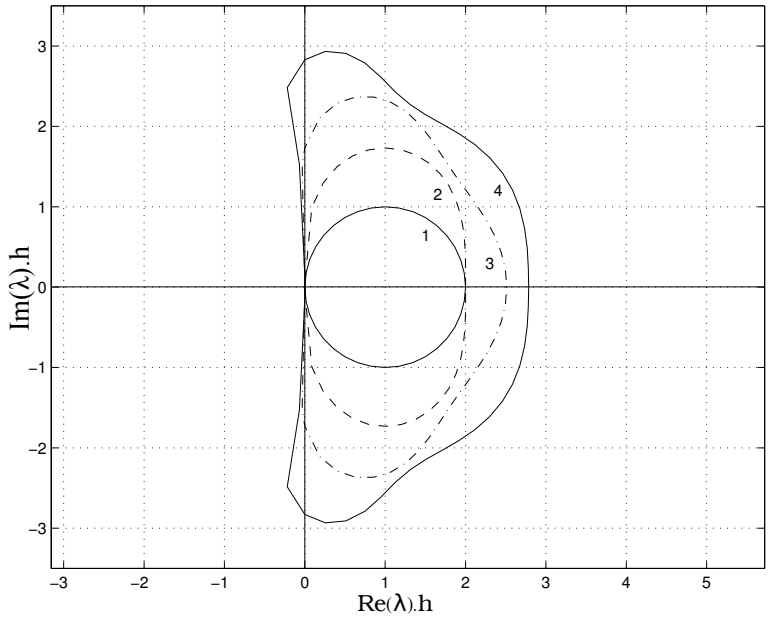

Figura 2.8: Dominios de estabilidad de los métodos básicos de interpolación hacia atrás.

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 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:

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

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

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:

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

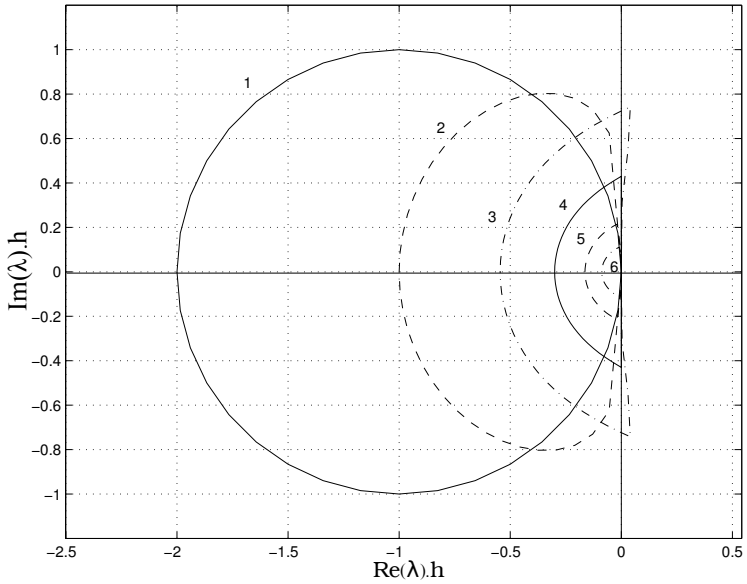

Figura 2.9: Dominios de estabilidad de los algoritmos explícitos AB.

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

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

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

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.

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:

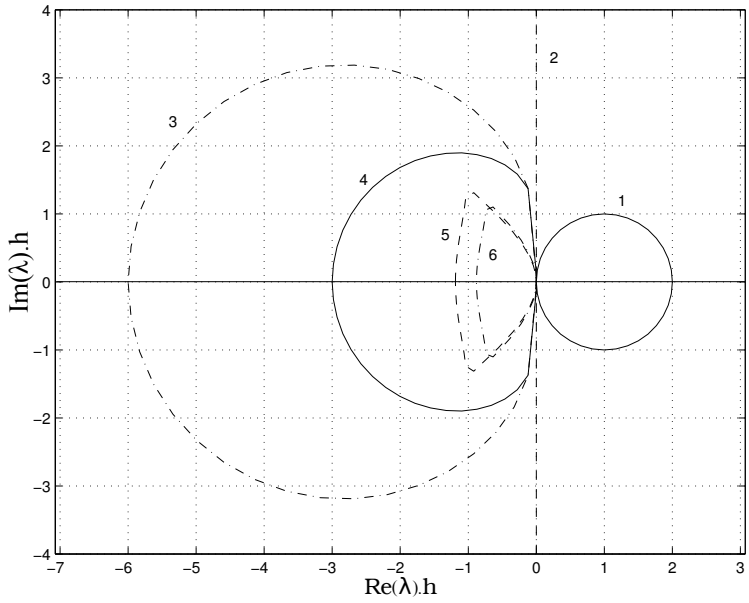

Figura 2.10: Dominios de estabilidad de los algoritmos implícitos de AM.

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

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.

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*).

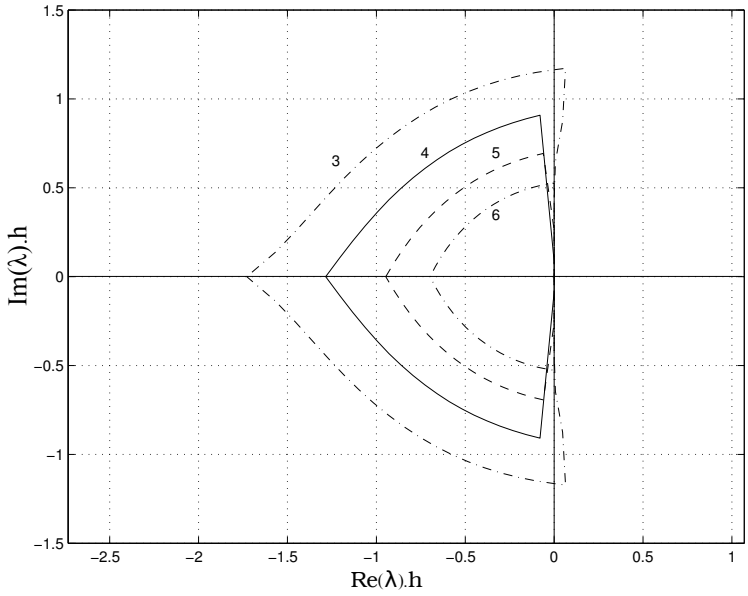

Figura 2.11: Dominios de estabilidad de los algoritmos predictor–corrector ABM.

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

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

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

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.

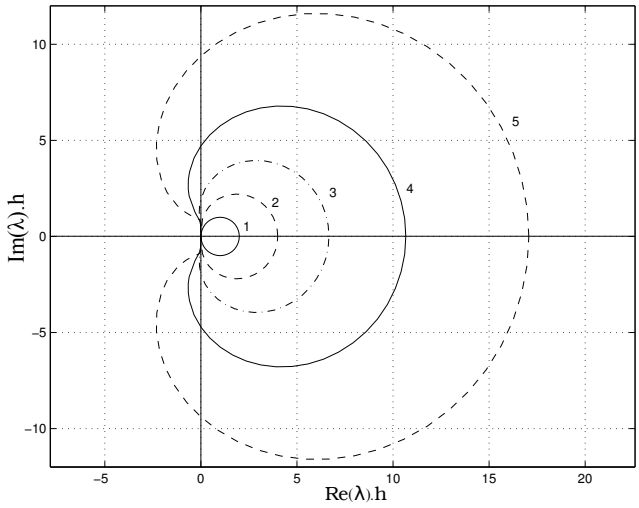

Figura 2.12: Dominios de estabilidad de los métodos implícitos BDF.

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 realizar 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 $h = 0{,}0005$.

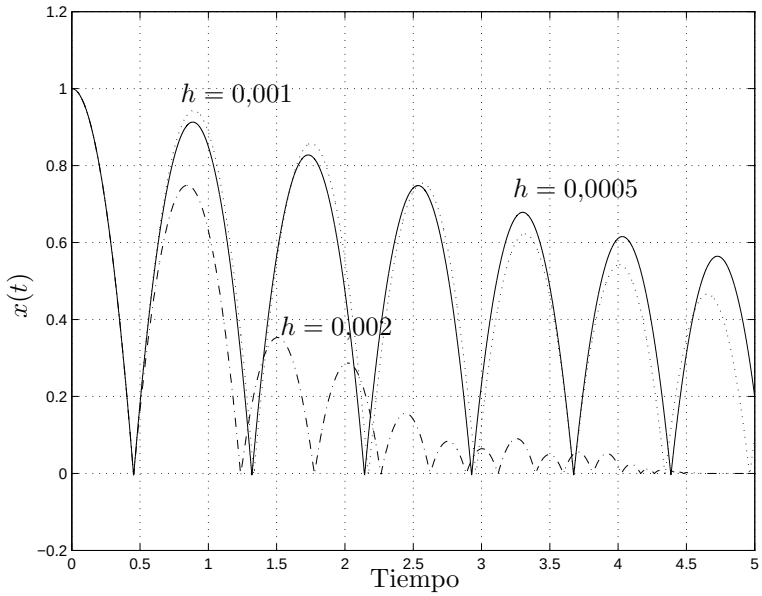

Figura 2.13: Simulación con RK4 de la pelotita rebotando.

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}.

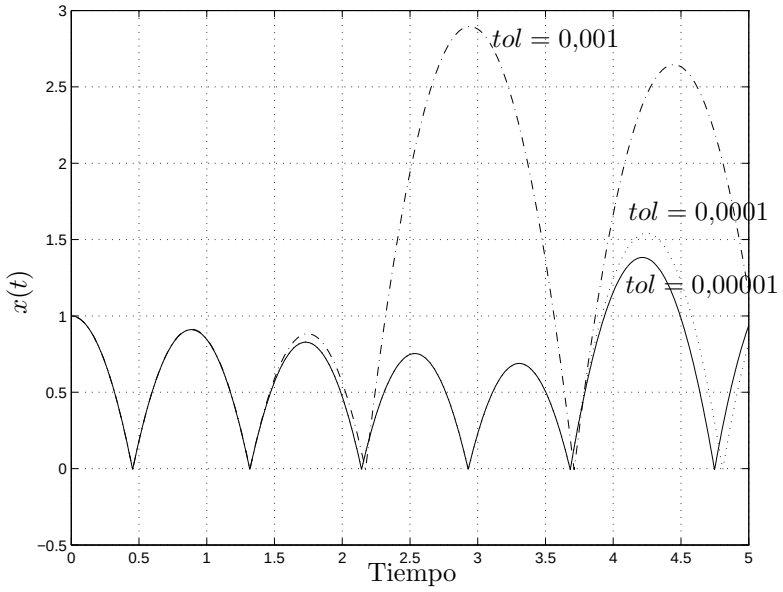

Figura 2.14: Simulación con RK23 de la pelotita rebotando.

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 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 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

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.

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).

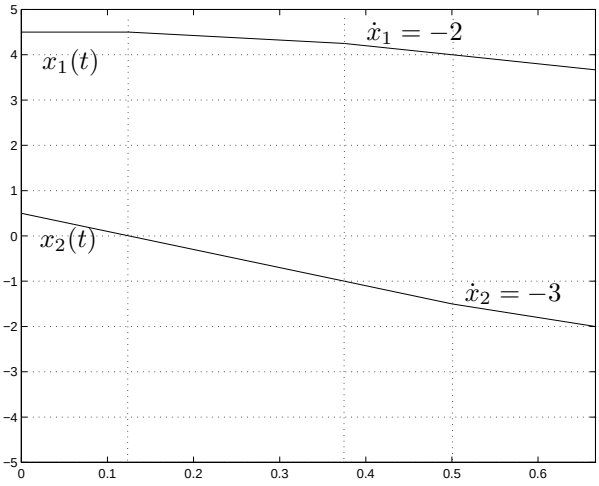

Figura 3.1: Simulación de las trayectorias del sistema de la Ec.(3.2) (Comienzo).

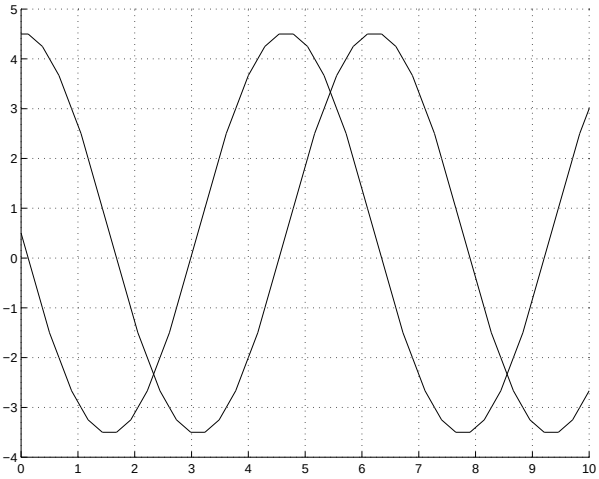

Figura 3.2: Simulación de las trayectorias del sistema de la Ec.(3.2) (Comienzo).

3.2. Sistemas de Eventos Discretos y DEVS

Todos los métodos de tiempo discreto aproximan las ecuaciones diferenciales por sistemas de tiempo discreto (ecuaciones en diferencia) de la forma:

$$\mathbf{x}(t_{k+1}) = \mathbf{f}(\mathbf{x}(t_k), t_k) \tag{3.3}$$

Con la nueva idea de cuantificar las variables de estado, sin embargo, se obtienen sistemas que son discretos (ya que realizan un número finito de cambios), pero no son de tiempo discreto.

Se verá que esta nueva manera de aproximar las ecuaciones nos lleva a *sistemas de eventos discretos*, dentro del formalismo DEVS.

DEVS, cuyas siglas provienen de *Discrete EVent System specification*, fue

introducido por Bernard Zeigler a mediados de la década de 1970. DEVS permite representar todos los sistemas cuyo comportamiento entrada/salida pueda describirse mediante secuencias de eventos.

En este contexto, un *evento* es la representación de un cambio instantáneo en alguna parte del sistema. Como tal, puede caracterizarse mediante un valor y un tiempo de ocurrencia. El valor puede ser un número, un vector, una palabra, o en general, un elemento de algún conjunto.

La trayectoria definida por una secuencia de eventos toma el valor ϕ (o *No Event*) en casi todos los instantes de tiempo, excepto en los instantes en los que hay eventos. En estos instantes, la trayectoria toma el valor correspondiente al evento. La Figura 3.3 muestra una trayectoria de eventos que toma los valores x_2 en el tiempo t_1, luego toma el valor x_3 en t_2, etc.

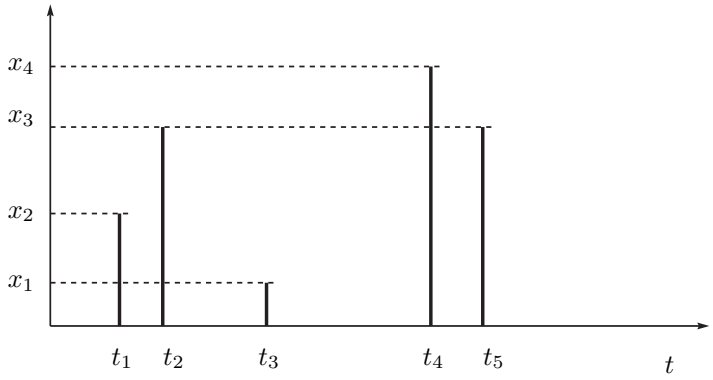

Figura 3.3: Trayectoria de eventos.

Un modelo DEVS procesa una trayectoria de eventos y, de acuerdo a dicha trayectoria y a sus propias condiciones iniciales, provoca una trayectoria de eventos de salida. Este comportamiento entrada/salida se muestra en la Fig.3.4.

Figura 3.4: Comportamiento Entrada/Salida de un modelo DEVS.

El comportamiento de un modelo DEVS *atómico* se explicita mediante distintas funciones y conjuntos, definidos en la siguiente estructura:

$$M = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta)$$

donde:

- X es el conjunto de los valores de los eventos de entrada, es decir el conjunto de todos los valores que un evento de entrada puede tomar;

- Y es el conjunto de valores de los eventos de salida;
- S es el conjunto de los valores del estado;
- δ_{int}, δ_{ext}, λ y ta son funciones que definen la dinámica del sistema.

La Figura 3.5 nos muestra el comportamiento de un modelo DEVS.

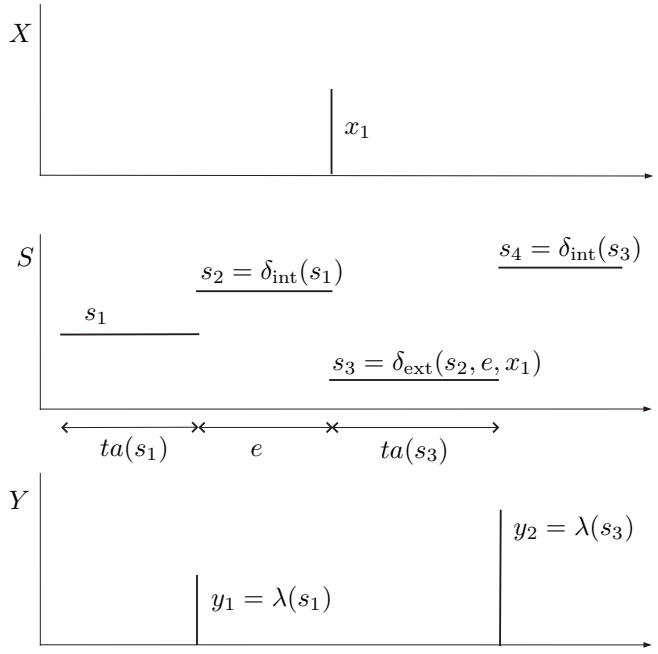

Figura 3.5: Trayectorias de un modelo DEVS.

Cada estado posible s ($s \in S$) tiene asociado un *tiempo de avance* calculado por la *función de avance de tiempo* $ta(s)$ ($ta(s) : S \to \Re_0^+$). El avance de tiempo es un número real no negativo, que determina cuánto tiempo debe permanecer el sistema en un estado en ausencia de eventos de entrada.

Luego, si el estado toma el valor s_1 en el instante t_1, tras $ta(s_1)$ unidades de tiempo (o sea, en el instante $t_1+ta(s_1)$), el sistema realiza una *transición interna*, alcanzando el nuevo estado s_2. El nuevo estado se calcula como $s_2 = \delta_{\text{int}}(s_1)$. La Función δ_{int} ($\delta_{\text{int}} : S \to S$) se denomina *función de transición interna*.

Cuando el estado cambia su valor desde s_1 a s_2, se produce un evento de salida con el valor $y_1 = \lambda(s_1)$. La función λ ($\lambda : S \to Y$) se denomina *función de salida*. De esta manera, las funciones ta, δ_{int} y λ definen el comportamiento autónomo de un modelo DEVS.

Cuando llega un evento de entrada, el estado cambia instantáneamente. El nuevo valor del estado depende no sólo del valor del evento de entrada, sino también del valor anterior del estado y del tiempo transcurrido desde la última transición. Si el sistema toma el valor s_2 en el instante t_2, y llega un evento en el instante $t_2 + e < ta(s_2)$ con valor x_1, el nuevo estado se calcula como $s_3 = \delta_{\text{ext}}(s_2, e, x_1)$. En este caso, se dice que el sistema realiza una *transición externa*. La función δ_{ext} ($\delta_{\text{ext}} : S \times \Re_0^+ \times X \to S$) se denomina *función de*

transición externa. Durante la transición externa, no se produce ningún evento de salida.

En muchos casos, además, los conjuntos X e Y (de donde toman los valores los eventos de entrada y salida) se forman por el producto cartesiano de un conjunto arbitrario (que contiene los valores de los eventos propiamente dichos) y un conjunto que contiene un número finito de *puertos* de entrada y salida respectivamente (luego se verá que estos puertos se utilizarán para acoplar modelos DEVS atómicos).

Sea por ejemplo un sistema que calcula una función estática $f(u_0, u_1)$, donde u_0 y u_1 son trayectorias seccionalmente constantes.

Una trayectoria seccionalmente constante puede ser representada mediante una secuencia de eventos si se relacion cada evento con un cambio en el valor de la trayectoria. Utilizando esta idea, se puede construir el siguiente modelo DEVS atómico:

$$
\begin{aligned}
&M_F = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta), \text{ donde} \\
&X = \Re \times \{0, 1\} \\
&Y = \Re \times \{0\} \\
&S = \Re^2 \times \Re_0^+ \\
&\delta_{\text{int}}(s) = \delta_{\text{int}}(u_0, u_1, \sigma) = (u_0, u_1, \infty) \\
&\delta_{\text{ext}}(s, e, x) = \delta_{\text{ext}}(u_0, u_1, \sigma, e, x_v, p) = \tilde{s} \\
&\lambda(s) = \lambda(u_0, u_1, \sigma) = (f(u_0, u_1), 0) \\
&ta(s) = ta(u_0, u_1, \sigma) = \sigma
\end{aligned}
$$

con:

$$
\tilde{s} = \begin{cases} (x_v, u_1, 0) & \text{si } p = 0 \\ (u_0, x_v, 0) & \text{en otro caso} \end{cases}
$$

En este modelo, se incluyó una variable σ en el estado s que coincide con la función ta. Esto se suele hacer siempre, ya que así se hace más fácil la obtención de un modelo DEVS.

Como se dijo anteriormente, cada evento de entrada y de salida incluye un número entero que indica el correspondiente puerto de entrada o salida. En los eventos de entrada, el puerto p puede ser 0 o 1 (es decir, hay dos puertos de entrada, uno para u_0y el otro para u_1). En los eventos de salida, hay un sólo puerto de salida (0).

3.2.1. Modelos DEVS Acoplados

DEVS es un formalismo muy general, que puede utilizarse para describir sistemas muy complejos. Sin embargo, representar un sistema muy complejo utilizando funciones de transición y avance de tiempo es muy difícil ya que para describir dichas funciones se debe tener en cuenta todas las posibles situaciones en el sistema.

Los sistemas complejos generalmente se piensan como el acoplamiento de sistemas más simples. A través del acoplamiento, los eventos de salida de unos subsistemas se convierten en eventos de entrada de otros subsistemas. La teoría

de DEVS garantiza que el modelo resultante de acoplar varios modelos DEVS atómicos es equivalente a un nuevo modelo DEVS atómico, es decir, DEVS es cerrado frente al acoplamiento (clausura del acoplamiento). Esto permite el acoplamiento jerárquico de modelos DEVS, o sea, la utilización de modelos acoplados como si fueran modelos atómicos que a su vez pueden acoplarse con otros modelos atómicos o acoplados.

Existen diversas formas de acoplar modelos DEVS. Una de ellas, y es la que utiliza el software PowerDEVS utilizado en esta Tesis, es el acoplamiento mediante puertos de entrada/salida.

La Figura 3.6 muestra un modelo DEVS acoplado N, resultado de acoplar los modelos M_a y M_b. De acuerdo a la propiedad de clausura de DEVS, el modelo N puede usarse de la misma forma que si fuera un modelo atómico, y puede ser acoplado con otros modelos atómicos y/o acoplados.

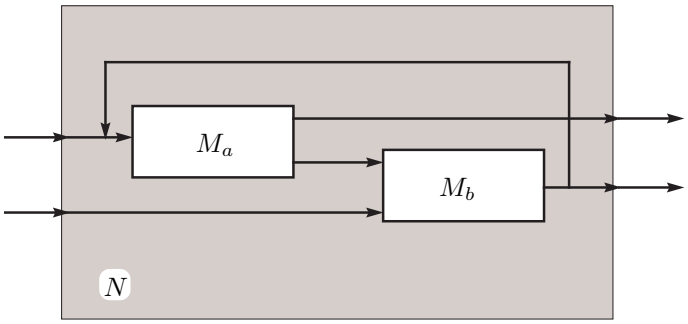

Figura 3.6: Modelo DEVS acoplado.

Para poder mostrar las conexiones entre los distintos modelos DEVS acoplados, se utilizarán en esta Tesis *Diagramas en Bloques* similares al mostrado en la Figura 3.6.

3.2.2. Simulación de Modelos DEVS y PowerDEVS

Una de las características más importantes de DEVS es su facilidad para implementar simulaciones. Un modelo DEVS puede simularse con un programa ad–hoc escrito en cualquier lenguaje. De hecho, la simulación de un modelo DEVS no es mucho más complicada que la de un modelo de tiempo discreto.

Un algoritmo básico que puede utilizarse para simular un modelo DEVS acoplado puede describirse por los siguientes pasos:

1. Identificamos el modelo atómico que, de acuerdo a su tiempo de avance y al tiempo transcurrido, deba ser el próximo en realizar la transición interna. Llamamos d^* a este sistema y t_n al tiempo de dicha transición.

2. Avanzamos el reloj de la simulación t a $t = t_n$, y ejecutamos la función de transición interna del modelo d^*.

3. Propagamos el evento de salida provocado por d^* a todos los modelos atómicos conectados al puerto de salida y ejecutamos las funciones de transición externas correspondientes. Luego, volvemos al paso 1.

Aunque, la implementación de una simulación de DEVS es muy simple, en general los modelos que se utilizan en la práctica están compuestos por muchos subsistemas y, hacer un programa ad–hoc de todos estos modelos suele tornarse muy trabajoso.

En esta Tesis, utilizaremos la herramientas de software denominada PowerDEVS para la simulación de los modelos DEVS. Este software, a pesar de ser un simulador de DEVS de propósito general, fue concebido especialmente para facilitar la simulación de sistemas continuos.

PowerDEVS tiene una interfase gráfica que permite crear modelos DEVS acoplados con las herramientas típicas de drag and drop. Cuenta también con librerías que tienen varios modelos atómicos ya definidos (muchos de ellos para realizar simulaciones con métodos de integración que veremos luego).

Además, los modelos atómicos pueden crearse y modificarse fácilmente utilizando el *editor de modelos atómicos*, donde el usuario sólo debe definir las funciones de transición, de salida y de avance de tiempo utilizando sintaxis C++.

3.2.3. DEVS y Simulación de Sistemas Continuos

En la Sección 3.2, se vio que las trayectorias seccionalmente constantes podían ser representadas mediante secuencias de eventos. Esta simple idea constituye en realidad la base del uso de DEVS en la simulación de sistemas continuos.

En esa sección se vio que un modelo DEVS puede representar el comportamiento de una función estática con una entrada seccionalmente constante. En los sistemas continuos las trayectorias no tienen esta forma, pero pueden ser alterados para que las trayectorias sean así. De hecho, esto es lo que se hizo con el sistema de la Ec.(3.1), donde se utilizó la función 'floor' para convertirlo en el sistema de la Ec.(3.2).

En este ejemplo, se puede dividir la Ec.(3.2) de la siguiente forma:

$$\dot{x}_1(t) = q_2(t) \tag{3.4a}$$
$$\dot{x}_2(t) = d_1(t) \tag{3.4b}$$
$$q_i(t) = \text{floor}[x_i(t)] \tag{3.4c}$$

y:

$$d_1(t) = -q_1(t) \tag{3.5}$$

Se puede representar este sistema usando el Diagrama en Bloques de la Fig.3.7.

Como se dijo anteriormente, el subsistema F_1 (ecuación (3.5)) -siendo una ecuación estática- puede ser representado por la función M_F presentada en la sección 3.2.

Cada subsistema QI_i integra una trayectoria de entrada $\dot{x}_i(t)$ seccionalmente constante para calcular la trayectoria seccionalmente lineal del estado $x_i(t)$ y genera una trayectoria de salida cuantificada $q_i(t) = \text{floor}[x_i(t)]$ seccionalmente constante. Esta salida cambia solamente cuando la trayectoria de estado $x_i(t)$ seccionalmente lineal cruza algún nivel de discretización

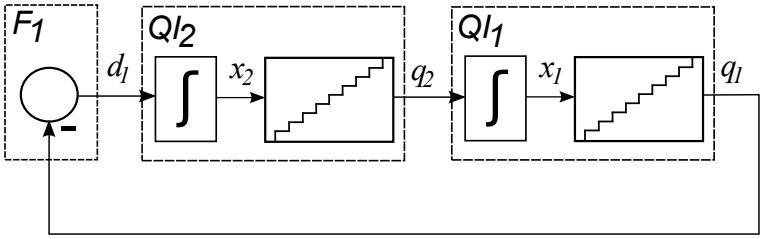

Figura 3.7: Representación en diagrama en Bloques de las Ecs.(3.4)-(3.5).

Este comportamiento también puede ser traducido fácilmente en un modelo DEVS.

$$
\begin{aligned}
&M_{QI} = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta), \text{ donde} \\
&X = Y = \Re \times \mathbb{N} \\
&S = \Re^2 \times \mathbb{Z} \times \Re_0^+ \\
&\delta_{\text{int}}(s) = \delta_{\text{int}}(x, d_x, q, \sigma) = (x + \sigma \cdot d_x, d_x, q + \text{sign}(d_x), \frac{1}{|d_x|}) \\
&\delta_{\text{ext}}(s, e, x) = \delta_{\text{ext}}(x, d_x, q, \sigma, e, x_v, p) = (x + e \cdot d_x, x_v, q, \tilde{\sigma}) \\
&\lambda(s) = \lambda(x, d_x, q, \sigma) = (q + \text{sign}(d_x), 0) \\
&ta(s) = ta(x, d_x, q, \sigma) = \sigma
\end{aligned}
$$

con:

$$
\tilde{\sigma} = \begin{cases} \frac{q+1-x}{x_v} & \text{si } x_v > 0 \\ \frac{q-x}{x_v} & \text{si } x_v < 0 \\ \infty & \text{en otro caso} \end{cases}
$$

El subsistema QI_j formado por un integrador junto con el bloque en forma de escalera (cuantificador) representa las ecuaciones

$$
\begin{aligned}
\dot{x}_i &= q_i \\
q_i &= \text{floor}[x_i]
\end{aligned}
$$

y es lo que Zeigler llamó *integrador cuantificado*. Aquí, la función 'floor' actúa como una *función de cuantificación*. En general, una función de cuantificación mapea un dominio continuo de números reales en un conjunto discreto de los reales.

Un sistema que relacione su entrada y su salida mediante cualquier tipo de función de cuantificación será entonces llamado *cuantificador*. En el caso mostrado, el bloque con forma de escalera es un caso particular de cuantificador con cuantificación uniforme.

Un integrador cuantificado es entonces un integrador concatenado con un cuantificador.

Luego, si se tiene un sistema general invariante en el tiempo [1]:

[1] Utilizando x_a para referirse a las variables de estado del sistema original, tal que $x_a(t)$ es la solución analítica.

$$\begin{aligned} \dot{x}_{a_1} &= f_1(x_{a_1}, x_{a_2}, \cdots, x_{a_n}, u_1, \cdots, u_m) \\ \dot{x}_{a_2} &= f_2(x_{a_1}, x_{a_2}, \cdots, x_{a_n}, u_1, \cdots, u_m) \\ &\vdots \\ \dot{x}_{a_n} &= f_n(x_{a_1}, x_{a_2}, \cdots, x_{a_n}, u_1, \cdots, u_m) \end{aligned} \tag{3.6}$$

con trayectorias de entrada seccionalmente constantes $u_j(t)$, puede transformarse en:

$$\begin{aligned} \dot{x}_1 &= f_1(q_1, q_2, \cdots, q_n, u_1, \cdots, u_m) \\ \dot{x}_2 &= f_2(q_1, q_2, \cdots, q_n, u_1, \cdots, u_m) \\ &\vdots \\ \dot{x}_n &= f_n(q_1, q_2, \cdots, q_n, u_1, \cdots, u_m) \end{aligned} \tag{3.7}$$

donde $q_i(t)$ se relaciona con $x_i(t)$ mediante alguna función de cuantificación.

Las variables q_i se llaman *variables cuantificadas*. Este sistema de ecuaciones puede representarse mediante el diagrama de bloques de la Fig.3.8, donde **q** y **u** son los vectores formados por las variables cuantificadas y por las variables de entrada respectivamente.

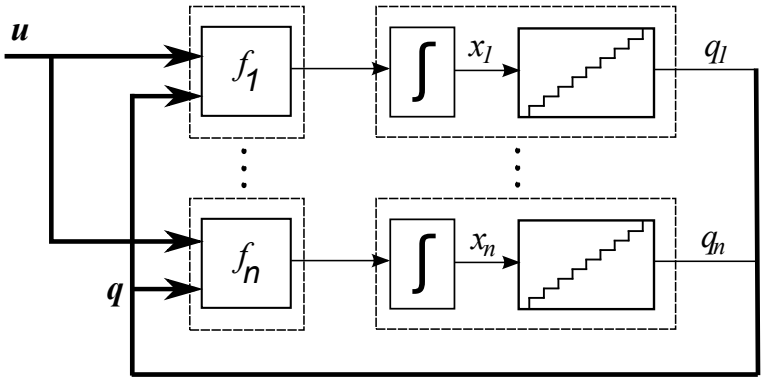

Figura 3.8: Diagrama en Bloques de la Ec.(3.7).

Cada subsistema en la Fig.3.8 puede representarse exactamente por un modelo DEVS, ya que todos los subsistemas son funciones estáticas o integradores cuantificados. Estos modelos DEVS pueden acoplarse, y, debido a la clausura bajo acoplamiento, el sistema completo formará también un modelo DEVS.

Entonces, cuando un sistema se modifica agregando cuantificadores en la salida de todos los integradores, el sistema resultante es equivalente a un modelo DEVS acoplado que puede simularse, siempre y cuando todas las entradas sean seccionalmente constantes.

Esta idea constituye la primer aproximación a un método para simular sistemas continuos mediante eventos discretos.

Desafortunadamente, esta idea no funciona ....

Este método conduce, en la mayor parte de los casos, a un modelo DEVS *ilegítimo*, es decir, un modelo que realiza un número infinito de transiciones en un intervalo acotado de tiempo. De tal forma, la simulación se trabará luego de cierto tiempo. Por ejemplo, dada la siguiente ecuación diferencial de primer orden:

$$\dot{x}(t) = -x(t) - 0{,}5 \tag{3.8}$$

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 (3.8) se obtiene

$$\dot{x}(t) = -q(t) - 0{,}5 \tag{3.9}$$

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.

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

Si se analizan las oscilaciones infinitamente rápidas en el sistema de la ecuación (3.8) 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.

3.3.1. 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:

$$q(t) = \begin{cases} \text{floor}[x(t_0)/\Delta Q] \cdot \Delta Q & \text{si } t = t_0 \\ x(t) & \text{si } |q(t^-) - x(t)| \geq \Delta Q \\ q(t^-) & \text{en otro caso} \end{cases} \tag{3.10}$$

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

Las Figuras 3.9–3.10 muestran una función de cuantificación con histéresis de orden cero y dos variables relacionadas mediante dicha función

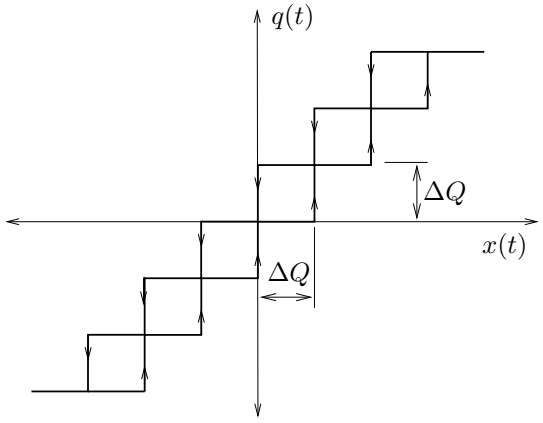

Figura 3.9: Función de Cuantificación con Histéresis de orden cero.

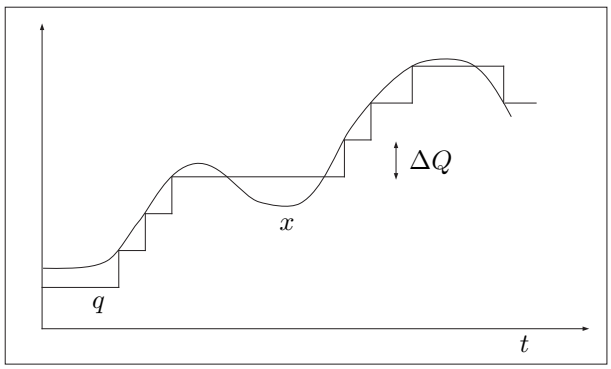

Figura 3.10: Variables Relacionadas con una Función de Cuantificación con Histéresis de orden cero.

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 como el de la Ec.(3.6), el método de QSS1 lo transforma en un *sistema de estados cuantificados* de la forma de la Ec.(3.7), donde cada par de variables $x_i(t)$ y $q_i(t)$ se relaciona mediante funciones de cuantificación con histéresis de orden cero.

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

En la figura 3.11 se ve la representación en diagrama de bloques de un sistema de estados cuantificados usando esta idea. Tal como se hizo anteriormente, el sistema puede ser separado en integradores cuantificados con histéresis y funciones estáticas.

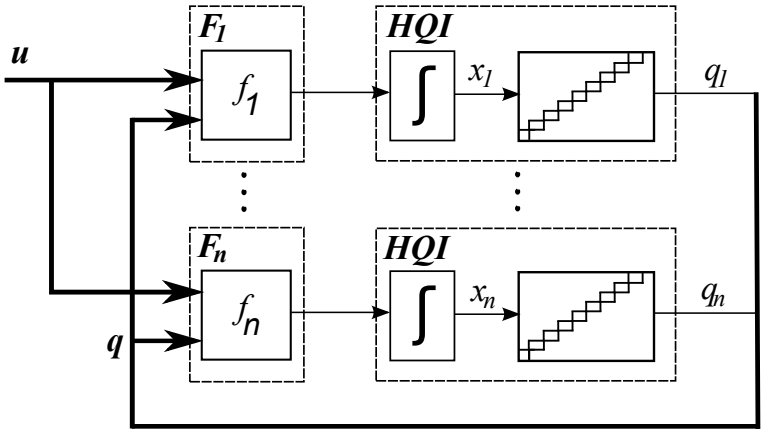

Figura 3.11: Representación en diagrama en bloques de un sistema de estados cuantificados.

Se puede demostrar que las variables cuantificadas y de estado son siempre seccionalmente constantes y seccionalmente lineales respectivamente. Por esto, la simulación con QSS no puede trabarse como ocurría con la cuantificación sin histéresis.

Para llevar a cabo la simulación, al igual que antes, podemos construir un modelo DEVS acoplando los subsistemas correspondientes a las funciones estáticas y a los *integradores cuantificados con histéresis*.

Modelo DEVS del integrador cuantificado de primer orden

Un integrador cuantificado con histéresis es un integrador cuantificado donde la función de cuantificación sin memoria es reemplazada por una con histéresis. Esto es equivalente a cambiar la Ec.(3.4c) por la Ec.(3.10) en el sistema de la Ec.(3.4).

Con esta modificación, el integrador cuantificado con histéresis puede representarse por el modelo DEVS:

$$
\begin{aligned}
&M_{HQI} = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta), \text{ donde} \\
&X = Y = \Re \times \{0\} \\
&S = \Re^3 \times \Re_0^+ \\
&\delta_{\text{int}}(s) = \delta_{\text{int}}(x, d_x, q, \sigma) = (x + \sigma \cdot d_x, d_x, x + \sigma \cdot d_x, \sigma_1) \\
&\delta_{\text{ext}}(s, e, x_u) = \delta_{\text{ext}}(x, d_x, q, \sigma, e, x_v, p) = (x + e \cdot d_x, x_v, q, \sigma_2) \\
&\lambda(s) = \lambda(x, d_x, q, \sigma) = (x + \sigma \cdot d_x, 0) \\
&ta(s) = ta(x, d_x, q, \sigma) = \sigma
\end{aligned}
$$

con:

$$\sigma_1 = \frac{\Delta Q}{|d_x|}$$

y:

$$\sigma_2 = \begin{cases} \frac{q + \Delta Q - (x + e \cdot d_x)}{x_v} & \text{si } x_v > 0 \\ \frac{(x + e \cdot d_x) - (q - \Delta Q)}{|x_v|} & \text{si } x_v < 0 \\ \infty & \text{si } x_v = 0 \end{cases}$$

3.3.2. 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 3.12.

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, mq a la pendiente de la misma y $x(t)$ a la entrada, el comportamiento descripto se puede escribir formalmente como:

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

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

$$|x(t) - q(t)| \leq \Delta q \quad \forall t \geq 0 \tag{3.11}$$

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).

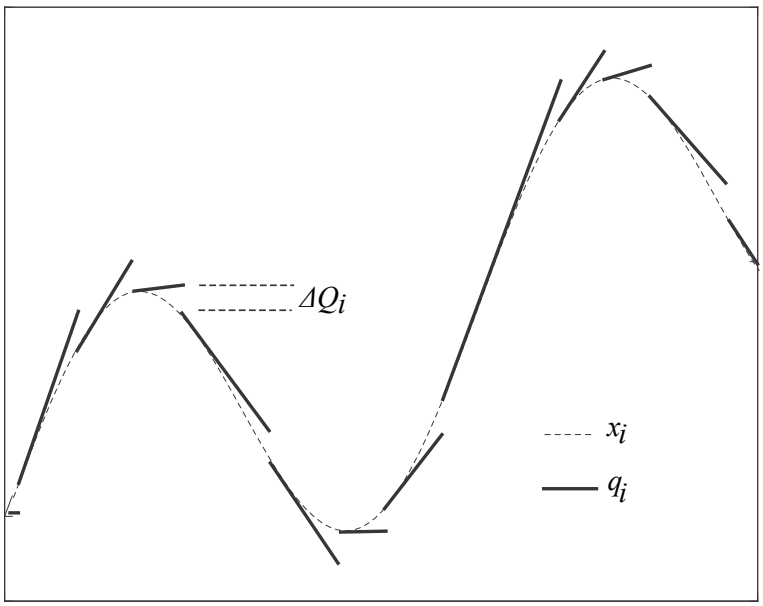

Figura 3.12: Función de cuantificación de primer orden.

Utilizando este método, se obtienen sistemas de estados cuantificados de segundo orden. Estos nuevos sistemas también pueden ser representados por las ecuaciones (3.7):

$$\begin{array}{rcl} \dot{x}_1 & = & f_1(q_1, q_2, \cdots, q_n, u_1, \cdots, u_m) \\ \dot{x}_2 & = & f_2(q_1, q_2, \cdots, q_n, u_1, \cdots, u_m) \\ & \vdots & \\ \dot{x}_n & = & f_n(q_1, q_2, \cdots, q_n, u_1, \cdots, u_m) \end{array} \tag{3.12}$$

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 [2] 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.

En figura 3.13 puede verse el diagrama en bloques de un sistema cuantificado de segundo orden representado mediante modelos atómicos DEVS. En el mismo se representan las trayectorias de salida de cada modelo atómico mediante su valor y pendiente.

[2] 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.

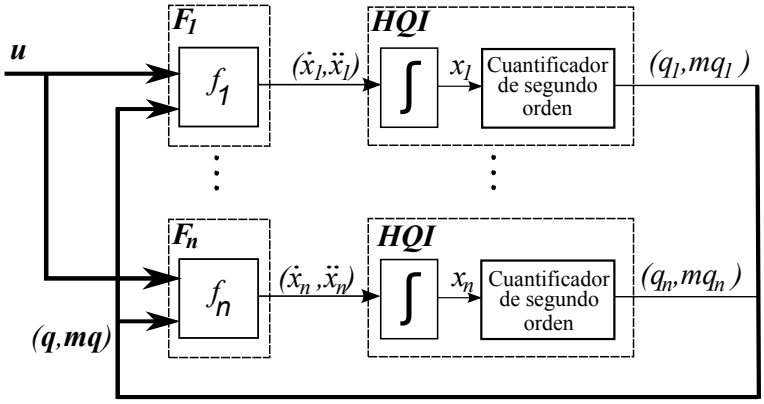

Figura 3.13: Diagrama en Bloques de un QSS2 genérico

Modelo DEVS del integrador cuantificado de segundo orden

El siguiente modelo DEVS representa el comportamiento de un integrador cuantificado de primer orden con trayectorias de entrada y salida seccionalmente lineales.

$$
\begin{aligned}
HQI &= (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta), \text{ donde} \\
X &= (\Re \times \Re) \times \{inports\} = \Re \times Re \times \{1\} \\
Y &= (\Re \times \Re) \times \{outports\} = \Re \times Re \times \{1\} \\
S &= \Re^5 \times \Re_0^+ \\
\delta_{\text{int}}(s) &= \delta_{\text{int}}(x, d_x, dd_x, q, mq, \sigma) \\
&= (\widetilde{x}, dx + ddx \cdot \sigma, dd_x, \widetilde{x}, dx + dd_x \cdot \sigma, \widetilde{\sigma}) \\
\delta_{\text{ext}}(s, e, v) &= \delta_{\text{ext}}(x, d_x, dd_x, q, mq, \sigma, e, v, mv, port) \\
&= (\hat{x}, v, mv, q + mq \cdot e, mq, \hat{\sigma}) \\
\lambda(s) &= \lambda(x, d_x, dd_x, q, mq, \sigma) = (\widetilde{x}, d_x + dd_x \cdot \sigma, 1) \\
ta(s) &= ta(x, d_x, q, \sigma) = \sigma
\end{aligned}
$$

donde:

$$
\begin{aligned}
\widetilde{x} &= x + d_x \cdot \sigma + dd_x \cdot \sigma^2/2 \\
\widetilde{\sigma} &= \begin{cases} \sqrt{\frac{2\Delta Q}{dd_x}} & \text{si } dd_x \neq 0 \\ \infty & \text{c.o.c} \end{cases} \qquad (3.13) \\
\hat{x} &= x + d_x \cdot e + dd_x \cdot e^2/2 \qquad (3.14)
\end{aligned}
$$

y $\hat{\sigma}$ se puede calcular como la menor solución positiva de:

$$\left|\hat{x} + v \cdot \hat{\sigma} + mv \cdot \hat{\sigma}^2 - (q + mq \cdot e + mq \cdot \hat{\sigma})\right| = \Delta Q \qquad (3.15)$$

Los eventos de entrada y salida están constituidos por números que representan el valor, la pendiente y el puerto de la trayectoria a los que están asociado. El estado $s \in S$ guarda el valor y pendiente (v y mv) del último valor a la

entrada, el valor actual de la variable de estado (x), el valor y pendiente (q y mq) del último valor que se sacó y el tiempo (σ) que falta para que se genere el próximo evento de salida.

La función de transición interna recalcula el estado s después de cada evento de salida y la función de transición externa hace lo mismo pero luego de que se recibe un evento de entrada. Las ecuaciones (3.13) y (3.15) calculan cuánto tiempo falta hasta el próximo evento de salida, o sea, hasta que la diferencia entre $x(t)$ y $q(t)$ sea igual a ΔQ. Finalmente, la función de salida (λ) genera el evento de salida con el valor y pendiente correspondientes.

Modelo DEVS de las funciones estáticas

Como se mencionó anteriormente, las funciones estáticas de los QSS2 deben tener en cuenta no solo el valor de las trayectorias de entrada sino también la pendiente de la misma. De igual modo, la salida de las mismas esta compuesta por el valor y pendiente de la trayectoria de salida.

Teniendo en cuenta esto el modelo DEVS de una función estática genérica $f_i(u_1, \ldots, u_n)$ resulta en este caso:

$$
\begin{aligned}
&F_i = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta), \text{ donde} \\
&X = \Re \times \Re \times \{inports\} = \Re \times Re \times 1, \ldots, n \\
&Y = (\Re \times \Re) \times \{outports\} = (\Re \times Re) \times \{1\} \\
&S = (\Re \times \Re \times \Re)^n \times \Re_0^+ \\
&\delta_{\text{int}}\left((u_1, mu_1, c_1), \ldots, (u_n, mu_n, c_n), \sigma\right) = \\
&\qquad = ((u_1, mu_1, c_1), \ldots, (u_n, mu_n, c_n), \infty) \\
&\delta_{\text{ext}}\left((u_1, mu_1, c_1), \ldots, (u_n, mu_n, c_n), \sigma, e, v, mv, port\right) = \\
&\qquad = ((\hat{u}_1, \hat{mu}_1, \hat{c}_1), \ldots, (\hat{u}_n, \hat{mu}_n, \hat{c}_n), 0) \\
&\lambda\left((u_1, mu_1, c_1), \ldots, (u_n, mu_n, c_n), \sigma\right) = \\
&\qquad = (f_i(\hat{u}_1, \ldots, \hat{u}_n), c_1 \cdot mu_1 + \ldots + c_n \cdot mu_n, 1) \\
&ta\left((u_1, mu_1, c_1), \ldots, (u_n, mu_n, c_n), \sigma\right) = \sigma
\end{aligned}
$$

con

$$
\begin{aligned}
\hat{u}_j &= \begin{cases} v & \text{si } j = port \\ u_j + mu_j \cdot e & \text{c.o.c} \end{cases} \\
\hat{mu}_j &= \begin{cases} mv & \text{si } j = port \\ mu_j & \text{c.o.c} \end{cases} \\
\hat{c}_j &= \begin{cases} \frac{f_i(u + mu \cdot e) - f_i(\hat{u})}{u_j + mu_j \cdot e - \hat{u}_j} & \text{si } j = port \wedge u_j + mu_j \cdot e - \hat{u}_j \neq 0 \\ c_j & \text{c.o.c} \end{cases}
\end{aligned}
\tag{3.16}
$$

Si la función f_i es lineal, este modelo DEVS representa exactamente su comportamiento. La ecuación (3.16) calcula los coeficientes que multiplican a la pendiente de las trayectorias de entrada. En el caso lineal, los mismos coincidirán con los coeficientes A_{ij} y B_{ij} correspondientes a las matrices de evolución y de entrada respectivamente.

En caso de ser f_i no lineal, la trayectoria de salida de la misma no sería seccionalmente lineal. Sin embargo, las trayectorias generadas por el modelo DEVS serán seccionalmente lineales y serán una buena aproximación a la salida real.

3.3.3. Método de QSS3

Para poder obtener una aproximación de tercer orden, se debe tener en cuenta no solo la primera derivada de las trayectorias del sistema (como en QSS2) sino que además se debe tener en cuenta la segunda derivada. En este sentido, se debe redefinir la función de cuantificador mostrado en la figura 3.12 de manera que las trayectorias de salida sean seccionalmente parabólicas. Luego, dado un sistema de ecuaciones de estado como el de la Ec.(3.6):

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

el método LIQSS3 lo aproxima por las Ec.(3.7):

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

donde x y q se relacionan componente a componente mediante funciones de cuantificación de segundo orden.

Una función de cuantificación de segundo orden genera una trayectoria de salida seccionalmente parabólica cuyo valor, pendiente y segunda derivada cambian cuando la diferencia entre su salida y su entrada difieren en más de un valor predeterminado (Fig.3.14).

Formalmente, se dice que las trayectorias $x_i(t)$ y $q_i(t)$ están relacionadas mediante una función de cuantificación de segundo orden si $q_i(t_0) = x_i(t_0)$ y

$$q_i(t) = \begin{cases} x_i(t) & \text{si } |q_i(t^-) - x_i(t^-)| = \Delta Q_i \\ q_i(t_j) + mq_{i_j} \cdot (t - t_j) + pq_{i_j} \cdot (t - t_j)^2 & \text{c.o.c} \end{cases} \tag{3.17}$$

con $t_j \leq t < t_{j+1}$, la secuencia $t_0, \ldots, t_j, \ldots$ tal que t_{j+1} es el mínimo $t > t_j$ en el cual

$$\left| x_i(t_j) + dx_i(t - t_j + ddx_i(t - t_j)^2 - q_i(t)) \right| = \Delta Q_i$$

y pendientes

$$\begin{array}{ll} mq_{i_0} = 0; & mq_{i_j} = \dot{x}_i(t_j^-), \quad j = 1, 2, \ldots \\ pq_{i_0} = 0; & pq_{i_j} = \ddot{x}_i(t_j^-), \quad j = 1, 2, \ldots \end{array}$$

Luego, los integradores cuantificados en LIQSS3 y las funciones estáticas tienen trayectorias de entrada y salida seccionalmente parabólicas. Una vez diferenciado el integrador cuantificado QSS3 y las correspondientes funciones

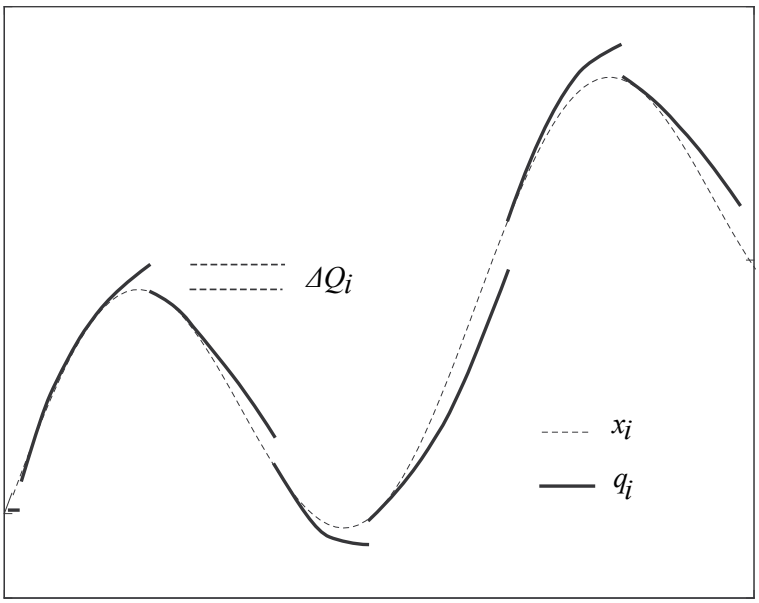

Figura 3.14: Función de cuantificación de Segundo orden.

estáticas, se procede de forma similar a lo hecho en QSS1 y QSS2 para construir los modelos DEVS de los mismos. Una explicación y desarrollo más profundo de los mismos puede encontrarse en [19].

3.3.4. 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:

$$\Delta Q_i(t) = \max\left(E_{rel_i} \cdot |x_i(t_k)|, \Delta Q_{i_m in}\right) \tag{3.18}$$

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:

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

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

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

Haciendo ahora la diferencia entre las Ecs.(3.19) y (3.20), se obtiene la ecuación para el error $\mathbf{e}(t) = \mathbf{x}(t) - \mathbf{x_a}(t)$:

$$\dot{\mathbf{e}}(t) = A \cdot (\mathbf{e}(t) + \mathbf{\Delta x}(t)) \tag{3.21}$$

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:

$$|\mathbf{e}(t)| \leq (I - RE_{rel})^{-1} R \max(E_{rel} \cdot |x_{max}|, \Delta Q_{min}) \tag{3.22}$$

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 (3.22) por:

$$|\mathbf{e}(t)| \leq R \max(E_{rel} \cdot |x_{max}|, \Delta Q_{min}) \tag{3.23}$$

3.3.5. Entorno de Simulación PowerDEVS

Toda la familia de métodos QSS (incluidos los métodos que se presentaran en esta Tesis) fueron implementados en PowerDEVS. Este software, a pesar de ser un simulador de DEVS de propósito general, fue concebido especialmente para facilitar la simulación de sistemas híbridos basados en los métodos QSS [8]. PowerDEVS tiene una interfase gráfica similar a Simulink que permite editar diagrama de bloques. A un nivel inferior, cada bloque tiene una descripción en lenguaje C++ del atómico DEVS que puede ser editada (usando la herramienta para edición de atómicos de PowerDEVS). Además, la distribución del mismo cuenta con librerías que tienen todos los bloques necesarios para modelar sistemas continuos e híbridos (integradores, funciones matemáticas, bloques para manejo de discontinuidades, fuentes,, etc.). En consecuencia, un usuario puede modelar y simular usando los métodos QSS sin necesidad de saber nada de DEVS, simplemente como si modelara o simulara en Simulink.

En las figuras 3.15 y 3.16 pueden verse dos capturas de pantalla en las que se ve la librería de bloques continuos de PowerDEVS y el modelo de un motor de corriente continua con un control de velocidad implementado a través de PWM con la ventana de simulación.

PowerDEVS puede intercambiar parámetros con Scilab durante la simulación del mismo modo que lo hacen Simulink y Matlab. Esta interacción permite explotar toda herramientas de procesamiento de datos, visualización, matemáticas y de manipulación de matrices de Scilab, dándole a su vez al usuario un espacio de trabajo interactivo.

A pesar de que PowerDEVS fue diseñado originalmente para funcionar en plataformas Windows, se desarrollo una versión ad–hoc de Kubuntu 8.04 que incluye PowerDEVS y módulos de Linux de tiempo real (RTAI)

El motor de PowerDEVS incluye módulos que usan funciones RTAI para sincronización en tiempo real, manejar interrupciones, medir el tiempo de CPU,

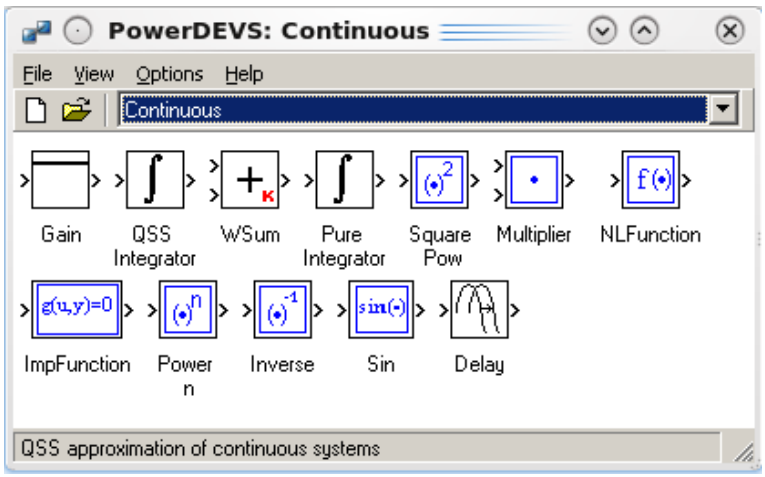

Figura 3.15: Pantalla principal de PowerDEVS.

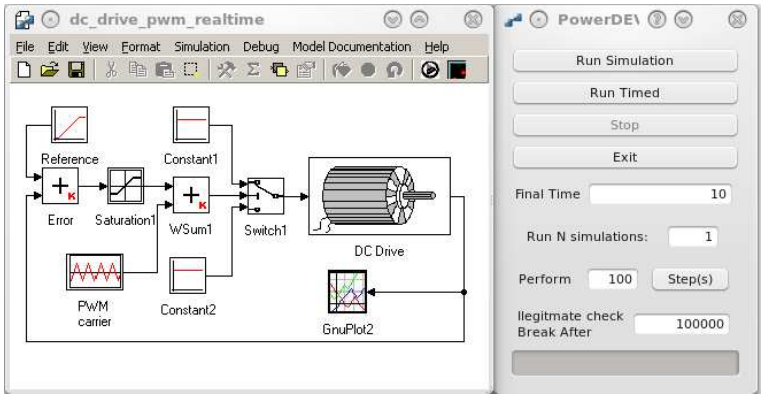

Figura 3.16: Modelo de un motor con control de Velocidad en PowerDEVS

etc. Estos módulos de tiempo real también fueron incluidos en bloques atómicos DEVS que permiten sincronizar la simulación con el reloj de tiempo real asegurando una precisión de aproximadamente 2 μsec, detectar interrupciones de hardware, acceder a los puertos de entrada/salida y medir el tiempo físico con una precisión del orden de los nanosegundos [4].

De esta manera, el usuario puede realizar simulaciones de tiempo real usando los métodos QSS de una manera muy conveniente. De hecho, resulta más fácil que utilizando el Real Time Workshop de Matlab/Simulink ya que PowerDEVS permite modelar y simular en el mismo SO.

Además de las implementaciones de los métodos QSS1,2 y 3 en PowerDEVS, los mismos se implementaron en Modelica [2], *Open Source Physics* (una herramienta de simulación basada en Java) [11] y el método de QSS1 se implementó en CD++ [9] y VLE [38].

3.4. 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 3.17 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

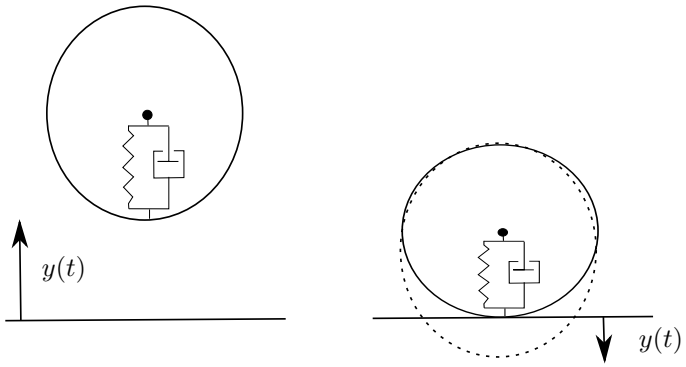

Figura 3.17: Pelota Rebotando

Este sistema puede ser modelado mediante las siguientes ecuaciones:

$$\begin{aligned} \dot{y}(t) &= v(t) \\ \dot{v}(t) &= \begin{cases} -g - \frac{b_a}{m} \cdot v(t) \cdot |v(t)| & \text{si } y(t) > 0 \\ -g - \frac{k}{m} \cdot y(t) - \frac{b}{m} \cdot v(t) & \text{c.o.c} \end{cases} \end{aligned} \tag{3.24}$$

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 [18].

Analicemos entonces la implementación PowerDEVS del ejemplo de la pelota rebotando mostrado en la figura Fig.3.18.

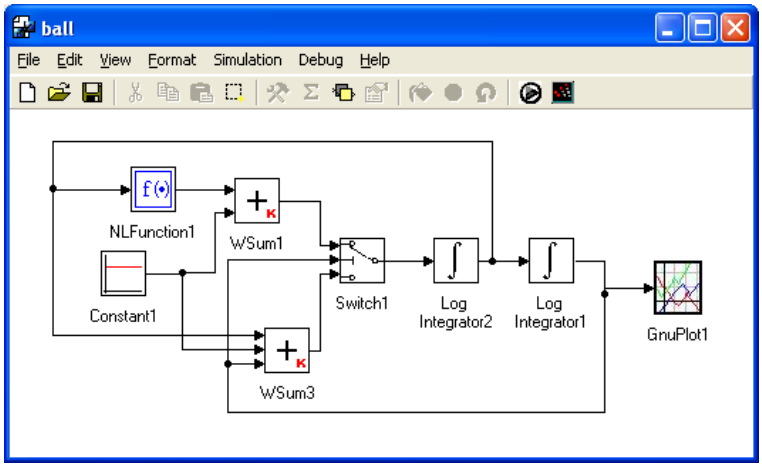

Figura 3.18: Modelo PowerDEVS de una pelota rebotando

El bloque 'Switch1' es el encargado de modelar y manejar la discontinuidad del modelo. Cada vez que recibe los sucesivos valores y pendientes de $y(t)$ por su segundo puerto de entrada, resuelve una ecuación cuadrática (se utilizaron integradores QSS3 en los cuales las trayectorias son seccionalmente parabólicas) para calcular el instante del próximo cruce y agenda para ese instante su próxima transición interna. Mientras tanto, el resto del sistema continua con la simulación. Cuando llega un nuevo valor de $y(t)$, el switch reagenda el tiempo para su próximo evento de salida. Cuando finalmente se llega a ese tiempo, el switch genera un evento con el nuevo valor de la derivada $\dot{v}(t)$. El integrador que calcula $v(t)$ recibe ese evento como un cambio normal en $\dot{v}(t)$ y lo trata como un evento normal.

En otras palabras, el bloque 'Switch1' maneja localmente las discontinuidades mientras que el resto de los bloques no saben de su presencia.

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 [18].

3.5. 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:

$$\dot{\mathbf{x}}_{\mathbf{a}}(t) = \mathbf{f}(\mathbf{x}_{\mathbf{a}}(t), \mathbf{u}(t)) \tag{3.25}$$

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

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{q}(t), \mathbf{u}(t)) \tag{3.26}$$

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

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t) + \mathbf{\Delta x}(t), \mathbf{u}(t)) \tag{3.27}$$

Esta última ecuación solo difiere del sistema original (3.25) 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* [27]. El análisis muestra que la solución de la Ec.(3.26) aproxima a la solución de la Ec.(3.25) 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 (3.26) converge a las trayectorias de las Ecs.(3.25) 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 [27] buscando una función de Lyapunov para el sistema perturbado. Este análisis muestra que cuando el sistema de la Ec.(3.25) 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.(3.26) 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.(3.25) 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 [26] 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:

$$\dot{\mathbf{x}}_\mathbf{a}(t) = \mathbf{A} \cdot \mathbf{x}_\mathbf{a}(t) + \mathbf{B} \cdot \mathbf{u}(t) \tag{3.28}$$

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:

$$\dot{\mathbf{x}}(t) = \mathbf{A} \cdot \mathbf{q}(t) + \mathbf{B} \cdot \mathbf{u}(t) \tag{3.29}$$

Sea $\mathbf{x}_\mathbf{a}(t)$ la solución de la Ec.(3.28) y $\mathbf{x}(t)$ la solución de la aproximación QSS de la Ec.(3.29) 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:

$$|\mathbf{x}(t) - \mathbf{x}_\mathbf{a}(t)| \preceq |\mathbf{V}| \cdot |\mathbb{R}\mathrm{e}\{\mathbf{\Lambda}\}^{-1} \cdot \mathbf{\Lambda}| \cdot |\mathbf{V}^{-1}| \cdot \mathbf{\Delta Q} \tag{3.30}$$

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.(3.30) 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 [34] 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.

Capítulo 4

Backward QSS

En el capítulo 3 se presentó una familia de métodos de integración de ecuaciones diferenciales basados en un enfoque esencialmente distinto al de los métodos de integración clásicos. A la familia de métodos de integración que surgió de este nuevo enfoque se la denominó métodos de integración por cuantificación.

Al comienzo de este capítulo se mostrará que, si bien los métodos de QSS poseen propiedades teóricas muy fuertes y presentan grandes ventajas para simular sistemas discontinuos tal como se mostró en el capítulo anterior, los mismos no son apropiados para la simulación de sistemas stiff debido a la aparición de oscilaciones de alta frecuencia.

Luego, se desarrollará un nuevo método de cuantificación de primer orden, similar a QSS1, pero utilizando principios de integración implícita, realizando la cuantificación sobre valores futuros de los estados. Esta idea ya había sido planteada en [17] y [8], pero nunca formalizada ni implementada.

Este nuevo método –que denominaremos BQSS por Backward Quantized State Systems– veremos que permite la simulación de sistemas stiff sin necesidad de reducir el paso de integración.

Además de ser el primer algoritmo implícito de QSS, BQSS tiene la particularidad de no requerir ningún tipo de iteraciones para su implementación. Gracias a la cuantificación, un estado cuantificado cuyo valor es q_i puede en el siguiente paso tomar solamente dos valores: $q_i + \Delta Q_i$ o $q_i - \Delta Q_i$ lo que simplifica muchísimo el problema. Más aún, veremos que la determinación del siguiente estado puede realizarse aún probando sólo uno de los dos valores posibles, lo que conlleva una solución explícita.

4.1. QSS y Sistemas Stiff

Para mostrar los inconvenientes que presentan los métodos de QSS para simular sistemas stiff consideremos el siguiente ejemplo.

Sea el sistema

$$\begin{aligned} \dot{x}_1(t) &= 0{,}01\, x_2(t) \\ \dot{x}_2(t) &= -100\, x_1(t) - 100\, x_2(t) + 2020 \end{aligned} \tag{4.1}$$

que tiene autovalores $\lambda_1 \approx -0{,}01$ y $\lambda_2 \approx -99{,}99$, por lo que puede considerarse stiff.

El método de QSS aproximará este sistema según

$$\begin{aligned} \dot{x}_1(t) &= 0{,}01\, q_2(t) \\ \dot{x}_2(t) &= -100\, q_1(t) - 100\, q_2(t) + 2020 \end{aligned} \tag{4.2}$$

Considerando condiciones iniciales $x_1(0) = 0$, $x_2(0) = 20$, y cuantificación $\Delta Q_1 = \Delta Q_2 = 1$, la integración por QSS hará los siguientes pasos:

En $t = 0$ tenemos $q_1(0) = 0$ y $q_2(0) = 20$. Por lo tanto $\dot{x}_1(0) = 0{,}2$ y $\dot{x}_2(0) = 20$. Esta situación se mantiene hasta que $|q_i - x_i| = \Delta Q_i = 1$.

El próximo cambio en q_1 es entonces agendado para $t = 1/0{,}2 = 5$ mientras que el cambio en q_2 se agenda para $t = 1/20 = 0{,}05$.

Por lo tanto en $t = 0{,}05$ hay un nuevo paso, tras el cual, $q_1(0{,}05) = 0$, $q_2(0{,}05) = 21$, $x_1(0{,}05) = 0{,}01$, $x_2(0{,}05) = 21$. Las derivadas quedan $\dot{x}_1(0{,}05) = 0{,}21$ y $\dot{x}_2(0{,}05) = -80$.

El próximo cambio en q_1 se reagenda para $t = 0{,}05 + (1 - 0{,}01)/0{,}21 = 4{,}764$ mientras el siguiente cambio en q_2 se agenda para $0{,}05 + 1/80 = 0{,}0625$. Por lo tanto, el siguiente paso se hace en $t = 0{,}0625$.

En $t = 0{,}0625$ tenemos $q_1(0{,}0625) = 0$, $q_2(0{,}0625) = x_2(0{,}0625) = 20$, $x_1(0{,}0625) \approx 0{,}0126$ y las derivadas son las mismas que en $t = 0$.

Esto se repite cíclicamente hasta que efectivamente se produce el cambio en q_1 (esto se da en $t \approx 4{,}95$, tras 158 cambios en el valor de q_2, oscilando entre 20 y 21).

La simulación continua de la misma manera. En las Figs.4.1–4.2 puede observarse la evolución de $q_1(t)$ y $q_2(t)$ hasta $t = 500$ seg.

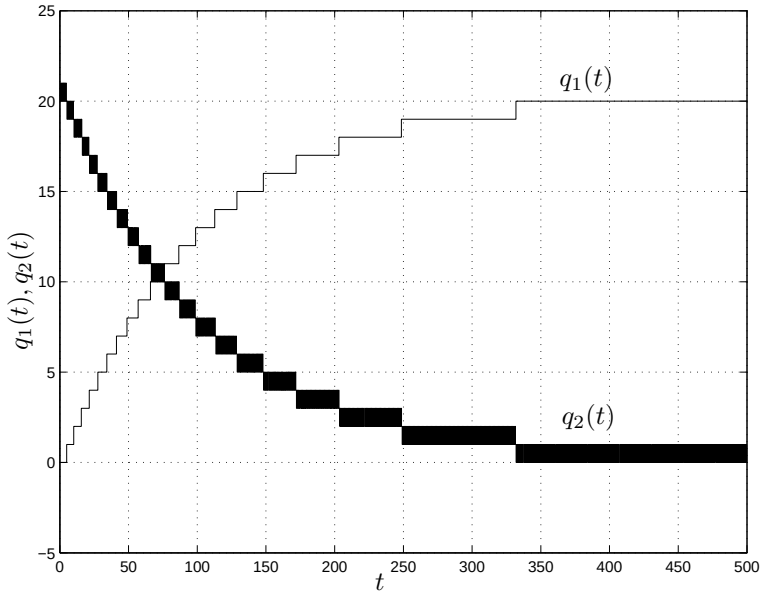

Figura 4.1: Simulación con QSS

Como puede verse, hay oscilaciones rápidas en q_2 que provocan un total de 15995 transiciones en dicha variable, mientras que q_1 realiza sólo 21 cambios. En definitiva hay más de 16000 pasos para completar la simulación (es del orden

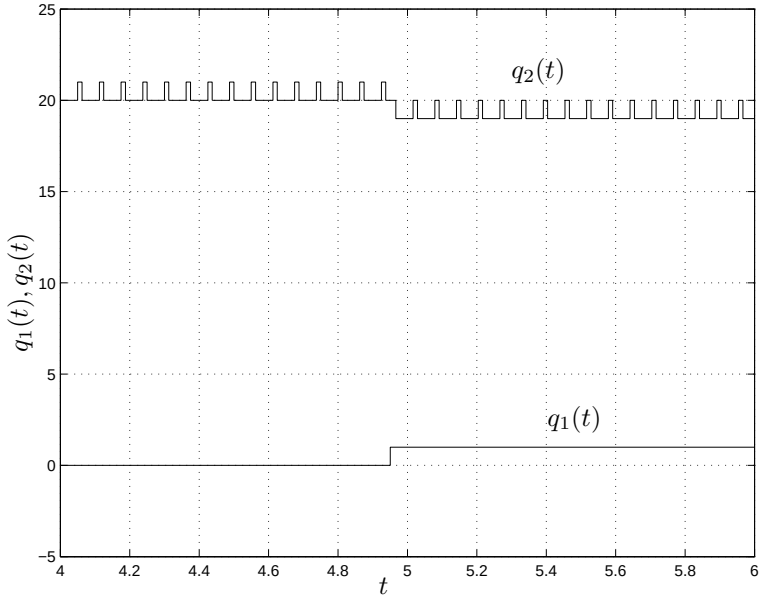

Figura 4.2: Simulación con QSS (detalle)

de los 25000 pasos que realizará como mínimo el método de Euler para obtener un resultado estable).

Evidentemente, el método de QSS no es capaz de integrar eficientemente el sistema (4.1).

4.2. Backward QSS

4.2.1. Idea Básica

Este nuevo método se inspiró en que todos los métodos de integración clásicos que son eficientes para la simulación de sistemas stiff requieren del uso de métodos implícitos, que evalúen la derivada en instantes futuros de tiempo.

Esta idea, aplicada a QSS implicará que las componentes de $\mathbf{q}(t)$ en

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

sean versiones cuantificadas de valores futuros de $x(t)$. En otras palabras, dado $x_i(t)$, $q_i(t)$ será un valor cuantificado cercano a $x_i(t)$ tal que $x_i(t)$ evolucione hacia $q_i(t)$.

Para el ejemplo (4.1), con las condiciones iniciales y cuantificación utilizada anteriormente, esta idea nos llevará a la siguiente simulación:

En $t = 0$, podemos elegir $q_2(0) = 19$ o $q_2(0) = 21$ según $\dot{x}_2(0)$ sea positivo o negativo. En ambos casos resultará $\dot{x}_1(0) > 0$ por lo que el valor cuantificado *futuro* de x_1 será $q_1(0) = 1$.

Si elegimos $q_2(0) = 21$ resulta $\dot{x}_2(0) = -180 < 0$, por lo que x_2 no evoluciona hacia q_2, sino en sentido contrario.

Por otro lado, si elegimos $q_2(0) = 19$ resulta $\dot{x}_2(0) = 20 > 0$. Luego, no es posible elegir q_2 tal que x_2 evolucione hacia q_2.

Sin embargo, dado que el signo de $\dot{x}_2$ cambia según elijamos $q_2 = 19$ o $q_2 = 21$, necesariamente hay un punto intermedio $\hat{q}_2$ entre estos valores tal que $\dot{x}_2 = 0$.

En este caso, dejaremos $q_2 = 21$ pero fijaremos $\dot{x}_2 = 0$ (como si q_2 hubiera adoptado el valor de la raíz $\hat{q}_2$).

El próximo cambio en q_1 quedará entonces agendado para $t = 1/0{,}21 \approx 4{,}762$ mientras que el próximo cambio en q_2 se agendará para $t = \infty$.

El siguiente paso es entonces en $t = 4{,}762$. Aquí resulta $x_1 = 1$ y $x_2 = 20$. Luego tendremos $q_1(4{,}762) = 2$ (porque $\dot{x}_1 > 0$). Aquí evaluamos nuevamente $\dot{x}_2$ para $q_2 = 19$ y $q_2 = 21$, resultando negativa en ambos casos, por lo tanto el valor correcto es $q_2(4{,}762) = 19$ ya que de esta manera x_2 evoluciona hacia q_2.

Con estos valores de q_1 y q_2 se tiene $\dot{x}_1 = 0{,}19$ y $\dot{x}_2 = -80$. El próximo cambio en q_1 se agenda para $t = 4{,}762 + 1/0{,}19 = 10{,}025$, mientras que el de q_2 se agenda para $t = 4{,}762 + 1/80 = 4{,}774$. Por lo tanto, el siguiente paso se da en $t = 4{,}774$, cuando x_2 alcanza a q_2.

Los cálculos continúan de la misma manera. El algoritmo es similar a QSS, sólo que siempre intentamos utilizar el valor q_i que esté en el sentido de la evolución de x_i. Cuando esto no es posible, es porque existe un punto intermedio en el cual $\dot{x}_i = 0$ y entonces forzamos esta condición, pero conservando el valor de q_i.

La Figura 4.3 muestra el resultado de esta simulación, que llevó 21 cambios en q_1 y 22 en q_2. En realidad, luego de $t = 354{,}24$ seg no se realizan más cálculos ya que los cambios en ambas variables se agendan en $t = \infty$ (se encuentra una situación estable).

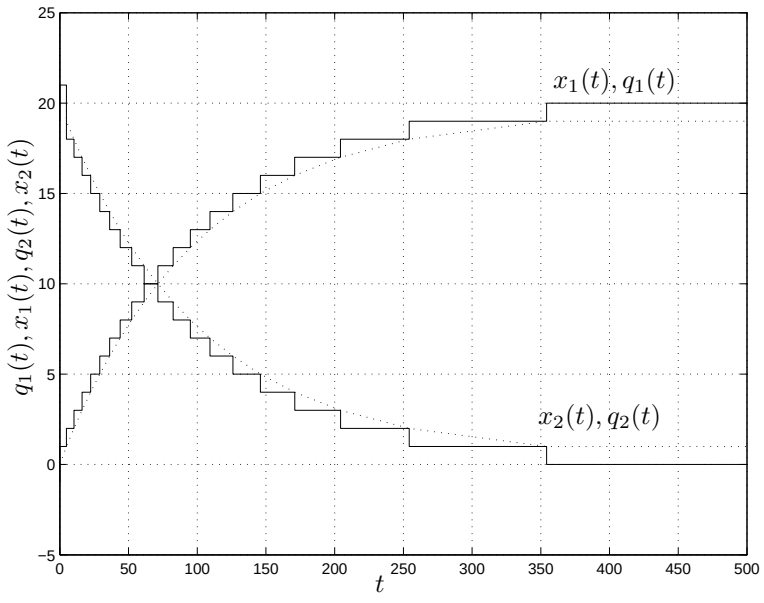

Figura 4.3: Simulación con QSS

Esta es la idea básica del método de BQSS: Para cada variable de estado usamos dos funciones de cuantificación, una por encima y otra por debajo del valor del estado x_i. Luego, el valor de q_i adopta una u otra función de cuantificación

según el sentido en que quede la derivada de x_i.

Esta idea, en el caso analizado, funcionó muy bien. El algoritmo pudo resolver el sistema stiff (4.1) con sólo 43 pasos, lo que iguala la performance de cualquier método implícito (si bien el error es aquí algo grande ya que es una aproximación de primer orden).

Sin embargo, como sabemos del caso de QSS, el uso de cuantificación sin histéresis puede provocar oscilaciones infinitamente rápidas. Por ejemplo, usando la misma idea que antes en el sistema

$$\begin{aligned} \dot{x}_1(t) &= 0{,}5\, x_1 + x_2(t) \\ \dot{x}_2(t) &= -x_1(t) + 0{,}5\, x_2(t) \end{aligned} \tag{4.3}$$

con $\Delta Q_i = 1$ y condiciones iniciales $x_1(0) = 0{,}1$, $x_2(0) = -0{,}01$ se obtiene una solución donde los cambios en q_1 y q_2 ocurren cada vez más rápido, con una frecuencia que tiende a infinito.

La solución a este problema, al igual que en QSS, es el agregado de histéresis en las funciones de cuantificación superior e inferior.

4.2.2. Definición de BQSS

Dado el sistema

$$\dot{\mathbf{x}}(\mathbf{t}) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \tag{4.4}$$

el método BQSS lo aproxima por:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{q}(t), \mathbf{u}(t)) + \boldsymbol{\Delta}\mathbf{f} \tag{4.5}$$

donde las componentes q_j de $\mathbf{q}$ verifican:

$$q_j(t) \in \{\underline{q}_j(t), \overline{q}_j(t)\} \tag{4.6}$$

con:

$$\underline{q}_j(t) = \begin{cases} \underline{q}_j(t^-) - \Delta Q_j \\ \quad \text{si } x_j(t) - \underline{q}_j(t^-) \leq 0 \\ \underline{q}_j(t^-) + \Delta Q_j \\ \quad \text{si } x_j(t) - \underline{q}_j(t^-) \geq \varepsilon_j + \Delta Q_j \\ \underline{q}_j(t^-) \text{ en otro caso} \end{cases} \tag{4.7}$$

$$\overline{q}_j(t) = \begin{cases} \overline{q}_j(t^-) + \Delta Q_j \\ \quad \text{si } \overline{q}_j(t^-) - x_j(t) \leq 0 \\ \overline{q}_j(t^-) - \Delta Q_j \\ \quad \text{si } \overline{q}_j(t^-) - x_j(t) \geq \varepsilon_j + \Delta Q_j \\ \overline{q}_j(t^-) \text{ en otro caso} \end{cases} \tag{4.8}$$

y además debe verificarse:

$$\begin{gathered} f_j(\mathbf{q}(t), \mathbf{u}(t)) \cdot (q_j(t) - x_j(t)) > 0 \\ \vee \\ \exists \hat{\mathbf{q}}^{(j)}(t) / f_j(\hat{\mathbf{q}}^{(j)}(t), \mathbf{u}(t)) = 0 \end{gathered} \tag{4.9}$$

donde cada componente del vector $\hat{\mathbf{q}}^{(j)}(t)$ cumple:

$$|x_i(t) - \hat{q}_i^{(j)}(t)| < \Delta Q_i + \varepsilon_i \tag{4.10}$$

Por otro lado

$$\Delta f_j = \begin{cases} 0, \text{ si } f_j(\mathbf{q}(t), \mathbf{u}(t)) \cdot (q_j - x_j) > 0 \\ -f_j(\mathbf{q}(t), \mathbf{u}(t)), \text{ en otro caso} \end{cases} \tag{4.11}$$

Como en QSS1, a ΔQ_j lo llamaremos quantum y a $\varepsilon_j < \Delta Q_j$ lo denominaremos ancho de histéresis.

Como vemos, la definición del valor de q_j conocido x_j, es implícita. Incluso, puede conllevar más de una solución. Sin embargo, sabemos que $q_j(t)$ puede tomar solamente 2 valores ($\underline{q}_j(t)$ o $\overline{q}_j(t)$).

En principio, podría pensarse que es necesario evaluar todas las combinaciones de valores posibles de los q_j hasta encontrar un vector $\mathbf{q}$ correcto. Sin embargo, como veremos a continuación, esto no es necesario sino que $\mathbf{q}$ puede obtenerse de manera explícita.

Obtención explicita de q

La diferencia principal de BQSS respecto de QSS1 es la forma de obtención de $\mathbf{q}$ a partir de $\mathbf{x}$, ya que (4.9) implica cierta interdependencia entre los valores de las distintas componentes.

En QSS1, los cambios en q_j se dan cuando x_j difiere de q_j en ΔQ_j. Similarmente, en BQSS los cambios se dan cuando x_j alcanza a q_j. Además, un cambio en q_j puede provocar cambios en otras variables cuantificadas debido a (4.9). Asimismo, cambios en alguna componente de $\mathbf{u}$ pueden de la misma forma provocar cambios en algunas variables cuantificadas.

En definitiva, los eventos pueden ser provocados por cambios en las entradas o porque una variable de estado alcanzó la correspondiente variable cuantificada. Tras cualquiera de estos cambios, la principal dificultad aparenta ser encontrar el valor de $\mathbf{q}$ correcto, que cumple con las Ecs.(4.6)–(4.10).

Veremos entonces un algoritmo que encuentra, de una forma muy simple y explícita, un valor de $\mathbf{q}$ que satisface las mencionadas ecuaciones.

Definimos para esto $\mathcal{D}$ como el conjunto de subíndices de las funciones f_i que van siendo evaluadas y $\mathcal{A}$ como el conjunto de subíndices de los q_i que cambian de valor (ambos inicialmente vacios).

Algoritmo 1.

1.a. Si cambia una entrada ($u_j(t) \neq u_j(t^-)$):
> Se agregan a $\mathcal{D}$ los subíndices de las f_i que dependen explícitamente de u_j.
> Para cada $i \in \mathcal{D}$:
- Definimos $\widetilde{\mathbf{q}}^{(i)} \triangleq \mathbf{q}(t^-)$.
- Si $f_i(\widetilde{\mathbf{q}}^{(i)}, \mathbf{u}(t)) \cdot (\widetilde{q}_i^{(i)} - x_i) < 0$
  - Se agrega i a $\mathcal{A}$
  - Definimos

$$q_i(t) \triangleq \begin{cases} \overline{q}_i(t) & \text{si } f_i(\widetilde{\mathbf{q}}^{(i)}, \mathbf{u}(t)) > 0 \\ \underline{q}_i(t) & \text{si } f_i(\widetilde{\mathbf{q}}^{(i)}, \mathbf{u}(t)) < 0 \end{cases} \tag{4.12}$$

- Sino, $q_i(t) \triangleq q_i(t^-)$

1.b. Si x_i alcanza a q_i:
> Se agrega i a $\mathcal{A}$ y $\mathcal{D}$.
> Se define $\widetilde{\mathbf{q}}^{(i)} \triangleq \mathbf{q}(t^-)$.
> Se calcula $q_i(t)$ según la Ec.(4.12)

2. Mientras $\mathcal{A} \neq \phi$
> Sea j el menor elemento de $\mathcal{A}$.
> Definimos $\mathcal{B}$ como el conjunto de subíndices $i \notin \mathcal{D}$ y tales que f_i depende explícitamente de q_j.

> Para cada $i \in \mathcal{B}$:
- Definimos $\widetilde{\mathbf{q}}^{(i)}$ con componentes

$$\widetilde{q}_k^{(i)} = \begin{cases} q_k(t) & \text{si } k \in \mathcal{D} \\ q_k(t^-) & \text{si } k \notin \mathcal{D} \end{cases} \tag{4.13}$$

- Si $f_i(\widetilde{\mathbf{q}}^{(i)}, \mathbf{u}(t)) \cdot (\widetilde{q}_i^{(i)} - x_i) < 0$
  - Se agrega i a $\mathcal{A}$.
  - Calculamos $q_i(t)$ según la Ec.(4.12).
- Sino, $q_i(t) \triangleq q_i(t^-)$.

> Se agregan los elementos de $\mathcal{B}$ al conjunto $\mathcal{D}$ y se saca j de $\mathcal{A}$.

3. Para los $i \notin \mathcal{D}$, se deja $q_i(t) = q_i(t^-)$.

Como puede verse, este algoritmo siempre encuentra un valor de $\mathbf{q}$ (luego demostraremos que cumple con la definición dada en la sección 4.2.2). Notar que ningún subíndice puede ser agregado más de una vez al conjunto $\mathcal{D}$, por lo que cada componente de la función $\mathbf{f}$ es evaluada a lo sumo una vez.

Teorema 4.1. *Supongamos que en la aproximación BQSS* (4.5) *conocemos* $\mathbf{x}(t)$ *y* $\mathbf{q}(t^-)$ *que satisface las Ecs.*(4.6)–(4.10)*. Asumamos además que* $u_i(t) \neq u_i(t^-)$ *o bien que* $x_i(t) = q_i(t^-)$ *para un subíndice* i*. Luego, el Algoritmo 1 siempre encuentra un valor* $\mathbf{q}(t)$ *que cumple con las Ecs.*(4.6)–(4.10)*.*

Demostración: Para los $i \notin \mathcal{D}$, definimos $\widetilde{\mathbf{q}}^{(i)} \triangleq \mathbf{q}(t^-)$.

Demostraremos entonces que una componente $q_j(t)$ arbitraria cumple las Ecs.(4.6)–(4.10).

Observar que los únicos valores que pueden ser asignados a $q_j(t)$ son $q_j(t^-)$, $\underline{q}_j(t)$ o $\overline{q}_j(t)$. En los últimos dos casos, la Ec.(4.6) se cumple automáticamente.

En el primer caso, al asignar $q_j(t) = q_j(t^-)$ puede verse que se cumple $\underline{q}_j(t) = \underline{q}_j(t^-)$ y $\overline{q}_j(t) = \overline{q}_j(t^-)$. De otra forma, $x_j(t)$ hubiera alcanzado a su variable cuantificada $q_j(t^-)$ y no hubiéramos asignado $q_j(t) = q_j(t^-)$ (punto 1.b.). Con esto, se garantiza que siempre se cumple la Ec.(4.6).

Para probar el cumplimiento de (4.9) y (4.10) se debe demostrar que si $f_j(\mathbf{q}(t), \mathbf{u}(t)) \cdot (q_j(t) - x_j(t)) \leq 0$ entonces $\exists \hat{\mathbf{q}}^{(j)}$ tal que $f_j(\hat{\mathbf{q}}^{(j)}, \mathbf{u}(t)) = 0$ y las componentes de $\hat{\mathbf{q}}$ satisfacen (4.10).

Notar que si $q_j(t) \neq q_j(t^-)$, el nuevo valor se calcula según la Ec. (4.12). Por lo tanto, teniendo en cuenta (4.7)–(4.8) se cumplirá que

$$f_j(\widetilde{\mathbf{q}}^{(j)}, \mathbf{u}(t)) \cdot (q_j(t) - x_j(t)) \geq 0 \tag{4.14}$$

Por otro lado, si se asignó $q_j(t) = q_j(t^-)$, era porque la ecuación (4.14) se cumplía. En definitiva, la ecuación (4.14) se cumple siempre.

Teniendo en cuenta esta ecuación, si $f_j(\mathbf{q}(t), \mathbf{u}(t)) \cdot (q_j(t) - x_j(t)) \leq 0$, entonces

$$f_j(\widetilde{\mathbf{q}}^{(j)}(t), \mathbf{u}(t)) \cdot f_j(\mathbf{q}(t), \mathbf{u}(t)) \leq 0$$

y por el teorema del valor medio, existe un $\hat{\mathbf{q}}^{(j)}$ entre $\widetilde{\mathbf{q}}^{(j)}(t)$ y $\mathbf{q}(t)$ tal que $f_j(\hat{\mathbf{q}}^{(j)}, \mathbf{u}) = 0$.

Las Ecs.(4.7) y (4.8) aseguran que

$$|\overline{q}_i - x_i| \leq \Delta Q_i + \varepsilon_i \,; \;\; |\underline{q}_i - x_i| \leq \Delta Q_i + \varepsilon_i \tag{4.15}$$

Por otro lado, puede verse fácilmente que las componentes $\widetilde{q}_i^{(j)}$ pueden adoptar solamente los valores $\overline{q}_i(t)$ o $\underline{q}_i(t)$, o directamente $x_i(t)$ (en el punto 1.b).

Entonces

$$\begin{array}{l} |\widetilde{q}_i^{(j)} - x_i(t)| \leq \Delta Q_i + \varepsilon_i \\ |q_i^{(j)} - x_i(t)| \leq \Delta Q_i + \varepsilon_i \end{array} \tag{4.16}$$

De esta ecuación y por estar $\hat{q}_i^{(j)}$ entre $q_i(t)$ y $\widetilde{q}_i^{(j)}(t)$ resulta

$$|\hat{q}_i^{(j)} - x_i(t)| \leq \Delta Q_i + \varepsilon_i \tag{4.17}$$

lo que concluye la demostración. □

4.2.3. Modelo DEVS del integrador BQSS

A continuación se describirá una forma de implementar un modelo DEVS equivalente a la aproximación BQSS de un sistema. Esta implementación tendrá una estructura igual a la de QSS, es decir, consistirá en el acoplamiento de funciones estáticas e integradores cuantificados como se muestra en la Fig.4.4.

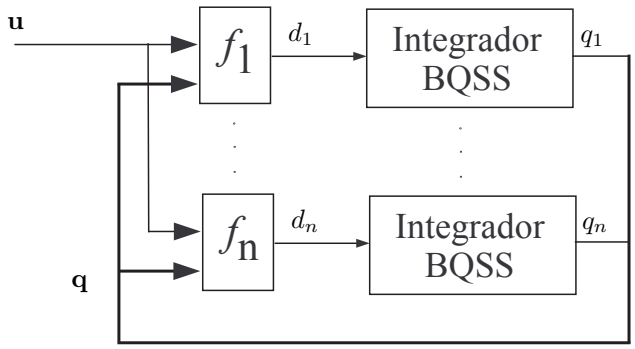

Figura 4.4: Diagrama de Bloques de un BQSS

En el mismo, dado que las trayectorias de $u_i(t)$ y $q_i(t)$ son seccionalmente constantes (esto último lo demostraremos en la siguiente sección), las funciones estáticas son idénticas a las de QSS.

La diferencia entre BQSS y QSS1 radicará en la forma de calcular q_i en los integradores cuantificados y eventualmente, en la forma de evaluar la derivada de x_i cuando $\Delta f_i \neq 0$ en (4.11).

Para calcular q_i utilizaremos el Algoritmo 1, pero implementado de manera local por los integradores cuantificados. Esto nos lleva a lo siguiente:

Supongamos que el integrador que calcula q_i recibe en el tiempo t un evento de entrada con valor d_i y supongamos que no había recibido ni provocado otro evento en dicho instante. Entonces, debe calcular

$$q_i(t) = \begin{cases} \overline{q}_i & \text{si } d_i > 0 \\ \underline{q}_i & \text{si } d_i < 0 \\ q_i(t^-) & \textit{c.o.c.} \end{cases} \tag{4.18}$$

Si el nuevo valor de q_i coincide con $q_i(t^-)$, no se produce ningún evento de salida manteniéndose $q_i(t) = q_i(t^-)$. Por otro lado, si el nuevo valor de q_i no coincide con $q_i(t^-)$, instantáneamente produce un evento de salida con el nuevo

valor de $q_i(t)$. Luego, cualquier nuevo evento de entrada d_i en el tiempo t no cambia el valor fijado en q_i.

Esto, implementado en todos los integradores, asegura que tras un número máximo de 1 evento de salida por cada integrador quede fijado en cada uno de estos el valor correcto de q_i de acuerdo al Algoritmo 1.

Una vez calculado q_i es simple agendar el próximo evento y fijar el valor de Δf_i. Si al recibir un evento de entrada con valor d_i resulta $(q_i - x_i) \cdot d_i > 0$, se calcula el tiempo en el que se producirá el próximo evento como $\sigma = (q_i - x_i)/d_i$ y se mantiene $\Delta f_i = 0$, sino, se fija $\sigma = \infty$ y $\Delta f_i = -d_i$.

Este comportamiento puede traducirse en el siguiente modelo DEVS.

$$M = \langle X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta \rangle \tag{4.19}$$

donde

$$X = Y = \mathbb{R};\ S = \mathbb{R}^5 \times \mathbb{R}^+$$
$$\delta_{\text{int}}(s) = \hat{s};\ \delta_{ext}(s, e, u) = \widetilde{s};\ ta(s) = \sigma$$

siendo

$$s = [x, d_x, q, q_i, q_s, \sigma]$$
$$\hat{s} = [\hat{x}, d_x, \hat{q}, \hat{q}_i, \hat{q}_s, \hat{\sigma}]$$
$$\hat{x} = x + \sigma \cdot d_x$$
$$\hat{q} = \begin{cases} \hat{q}_s & \text{si } d_x > 0 \wedge \sigma > 0 \\ q_s & \text{si } d_x > 0 \wedge \sigma = 0 \\ \hat{q}_i & \text{si } d_x \leq 0 \wedge \sigma > 0 \\ q_i & \text{si } d_x \leq 0 \wedge \sigma = 0 \end{cases}$$
$$\hat{q}_i = \begin{cases} \hat{q}_s - 2 \cdot \Delta Q & \text{si } d_x > 0 \wedge \sigma > 0 \\ q_i - \Delta Q & \text{si } d_x \leq 0 \wedge \sigma > 0 \\ q_i & \textit{c.o.c} \end{cases}$$
$$\hat{q}_s = \begin{cases} q_s + \Delta Q & \text{si } d_x > 0 \wedge \sigma > 0 \\ \hat{q}_i + 2 \cdot \Delta Q & \text{si } d_x \leq 0 \wedge \sigma > 0 \\ q_s & \textit{c.o.c} \end{cases}$$
$$\hat{\sigma} = \begin{cases} (\hat{q} - \hat{x})/d_x & \text{si } d_x \neq 0 \\ \infty & \textit{c.o.c} \end{cases}$$
$$\widetilde{s} = [x + e \cdot d_x, \widetilde{d}_x, q, \widetilde{q}_i, \widetilde{q}_s, \widetilde{\sigma}]$$
$$\widetilde{d}_x = \begin{cases} 0 & \text{si } u \cdot (q - \widetilde{x}) < 0 \wedge e = 0 \\ u & \textit{c.o.c.} \end{cases}$$
$$\widetilde{q}_i = \begin{cases} q_i + \Delta Q & \text{si } \widetilde{x} - q_i \geq \Delta Q + \epsilon \\ q_i & \textit{c.o.c} \end{cases}$$
$$\widetilde{q}_s = \begin{cases} q_s - \Delta Q & \text{si } q_s - \widetilde{x} \geq \Delta Q + \epsilon \\ q_s & \textit{c.o.c} \end{cases}$$
$$\widetilde{\sigma} = \begin{cases} (q - \widetilde{x})/u & \text{si } u \neq 0 \wedge u \cdot (q - \widetilde{x}) \geq 0 \\ 0 & \text{si } u \cdot (q - \widetilde{x}) < 0 \wedge e > 0 \\ \infty & \textit{c.o.c.} \end{cases}$$

y la Función de Salida resulta

$$\lambda(s) = \begin{cases} q_s & \text{si } \sigma = 0 \wedge u > 0 \\ q_i & \text{si } \sigma = 0 \wedge u \leq 0 \\ q + \Delta Q & \text{si } \sigma \neq 0 \wedge u > 0 \\ q - \Delta Q & \textit{c.o.c} \end{cases}$$

4.3. Propiedades Teóricas de BQSS

Trataremos aquí las propiedades fundamentales del método de BQSS. Las mismas se encuentran publicadas en [33]. Primero veremos que en BQSS se realizan un número finito de pasos para cada intervalo finito de tiempo (lo que garantiza que las simulaciones no pueden *trabarse*). Luego, analizaremos las propiedades de estabilidad y cota de error global.

Con el fin de simplificar la lectura y comprensión de las propiedades teóricas recordemos que los métodos de integración por cuantificación se basan en aproximar la ODE

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \tag{4.20}$$

por su representación cuantificada

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{q}(t), \mathbf{u}(t)) \tag{4.21}$$

4.3.1. Legitimidad

Teorema 4.2. *Supongamos que la función* $\mathbf{f}$ *del lado derecho de la ODE* (4.20) *es acotada en cualquier dominio acotado y que la función* $\mathbf{u}(t)$ *es seccionalmente constante. Luego,*

1. *Cualquier solución* $\mathbf{x}(t)$ *de* (4.5) *es continua mientras permanece acotada,*
2. *La trayectoria de* $\mathbf{q}(t)$ *es seccionalmente constante mientras permanece acotada.*

Demostración: La prueba del punto (1) es trivial, ya que al estar acotada la derivada de $\mathbf{x}$ en (4.5), la trayectoria resulta continua.

Para el punto (2) en tanto, es claro que cada componente q_j puede solamente tomar valores de la forma $k \cdot \Delta Q_j$. Sin embargo, para probar que $\mathbf{q}$ es seccionalmente constante es necesario asegurar que solamente cambia un número finito de veces en cualquier intervalo finito de tiempo.

Sea entonces un intervalo arbitrario (t_1, t_2), en el cual una solución $\mathbf{x}(t)$ de (4.5) permanece acotada. Probaremos que en este intervalo $\mathbf{q}(t)$ realizará sólo un número finito de cambios.

Dado que cada componente q_j difiere de x_j en a lo sumo $\Delta Q_j + \epsilon_j$, resulta que $\mathbf{q}(t)$ está también acotado en (t_1, t_2). Dado que $\mathbf{u}(t)$ es seccionalmente constante (y por ende acotado en este intervalo) resulta que $\mathbf{f}(\mathbf{q}, \mathbf{u})$ es también acotado. Teniendo en cuenta (4.5) y (4.11), existen constantes m_j tales que en el intervalo (t_1, t_2)

$$|\dot{x}_j(t)| \leq m_j; \text{ para } j = 1, \ldots, n.$$

Sea $t_c \in (t_1, t_2)$ y supongamos que $\overline{q}_j(t_c^-) \neq \overline{q}_j(t_c^+)$. De acuerdo a (4.8), esta situación no puede repetirse hasta que $|x_j(t) - x_j(t_c)| \geq \epsilon_j$. Por lo tanto, el intervalo de tiempo mínimo entre dos discontinuidades en $\overline{q}_j(t)$ es

$$t_j = \frac{\epsilon_j}{m_j}$$

Luego, llamando $\overline{n}_j$ al número de cambios que realiza $\overline{q}_j(t)$ en el intervalo (t_1, t_2), resulta

$$\overline{n}_j \leq (t_2 - t_1)\frac{m_j}{\epsilon_j}$$

Puede verse fácilmente que $\underline{q}_j$ realizará también un máximo numero de cambios acotado por la ecuación anterior.

Siendo $\mathbf{u}(t)$ seccionalmente constante, realizará un número de cambios finito n_u en el intervalo (t_1, t_2).

De acuerdo a la definición de $q_j(t)$, ésta puede tomar solamente los valores $\overline{q}_j(t)$ o $\underline{q}_j(t)$. Además, podrá cambiar sólo si cambian éstas, o si cambia alguna otra variable $q_i(t)$ o $u_i(t)$ para que se siga cumpliendo la restricción (4.9). En definitiva, los cambios de $q_j(t)$ estarán ligados a cambios en alguna $\overline{q}_i(t)$, $\underline{q}_i(t)$ o $u_i(t)$. Por lo tanto, el número total de cambios será menor o igual a la suma de cambios en dichas variables, es decir,

$$n_j \leq n_u + 2(t_2 - t_1) \sum_{i=1}^{n} \frac{m_i}{\epsilon_i}$$

lo que es evidentemente un número finito. □

4.3.2. Representación Perturbada

Las propiedades teóricas de los métodos de QSS se basan en el hecho de que la aproximación (4.21) puede verse como una versión perturbada del sistema original (4.20), donde las perturbaciones en la primera están acotadas por la cuantificación utilizada. Veremos entonces que en BQSS ocurre algo similar.

Cada componente de la Ec.(4.5) puede escribirse como

$$\dot{x}_i(t) = f_i(\mathbf{q}(t), \mathbf{u}(t)) + \Delta f_i \tag{4.22}$$

Definiendo

$$\overset{*}{\mathbf{q}}{}^{(i)}(t) = \begin{cases} \mathbf{q}(t) & \text{si } \Delta f_i = 0 \\ \hat{\mathbf{q}}^{(i)}(t) & \text{en otro caso} \end{cases}$$

y utilizando las Ecs. (4.9)–(4.11), la Ec.(4.22) puede reescribirse como

$$\dot{x}_i = f_i(\overset{*}{\mathbf{q}}{}^{(i)}(t), \mathbf{u}(t)) \tag{4.23}$$

Definiendo $\mathbf{\Delta x}^{(i)}(t) \triangleq \overset{*}{\mathbf{q}}{}^{(i)} - \mathbf{x}(t)$ y reemplazando en la Ec.(4.23) resulta

$$\dot{x}_i(t) = f_i(\mathbf{x}(t) + \mathbf{\Delta x}^{(i)}(t), \mathbf{u}(t)) \tag{4.24}$$

donde

$$|\Delta x_j^{(i)}(t)| \leq \Delta Q_j + \varepsilon_j \tag{4.25}$$

ya que, de las Ecs.(4.6), (4.7) y (4.8) resulta

$$|q_j(t) - x_j(t)| \leq \Delta Q_j + \varepsilon_j$$

y de la Ec.(4.10) se tiene que

$$|\hat{q}_j^{(i)}(t) - x_j(t)| \leq \Delta Q_j + \varepsilon_j$$

4.3.3. Estabilidad y Cota de Error Global

El caso Lineal y Estacionario de la Ec.(4.20) toma la forma

$$\dot{\mathbf{x}}(t) = A\,\mathbf{x}(t) + B\mathbf{u}(t) \tag{4.26}$$

La aproximación BQSS, en tanto, será

$$\dot{\mathbf{x}}(t) = A\,\mathbf{q}(t) + B\mathbf{u}(t) + \mathbf{\Delta f}(t) \tag{4.27}$$

Para este caso, el siguiente teorema demuestra la existencia de una cota de error global:

Teorema 4.3. *Supongamos que la matriz A es Hurwitz. Sea $\phi(t)$ la solución de* (4.26) *con condición inicial $\phi(0)$ y sea $\tilde{\phi}(t)$ una solución de* (4.27) *desde la misma condición inicial. Sea $\mathbf{e}(t) \triangleq \phi(t) - \tilde{\phi}(t)$. Luego, para todo $t \geq 0$ resulta*[1]

$$|\mathbf{e}(t)| \leq |V| \cdot |\mathbb{R}e(\Lambda)^{-1}V^{-1}||A|(\mathbf{\Delta Q} + \varepsilon) \tag{4.28}$$

donde $\Lambda = V^{-1}A\,V$ es la descomposición modal de A y $\mathbf{\Delta Q}$ y ε son los vectores de las cuantificaciones y anchos de histéresis en (4.27).

Demostración:

De acuerdo a la Ec.(4.24), la i–ésima componente de (4.27) puede escribirse como

$$\dot{x}_i(t) = A_i(\mathbf{x}(t) + \mathbf{\Delta x}^{(i)}(t)) + B_i\,\mathbf{u}(t))$$

Definiendo $d_i(t) \triangleq A_i\,\mathbf{\Delta x}^{(i)}(t)$, podemos reescribir

$$\dot{x}_i(t) = A_i(\mathbf{x}(t)) + d_i + B_i\,\mathbf{u}(t)) \tag{4.29}$$

Teniendo en cuenta (4.25) y la definición de d_i resulta

$$|d_i(t)| \leq |A_i| \cdot (\mathbf{\Delta Q} + \varepsilon) \tag{4.30}$$

Volviendo a la notación vectorial, (4.29) resulta:

$$\dot{\mathbf{x}}(t) = A\,\mathbf{x}(t) + B\mathbf{u}(t) + \mathbf{d}(t) \tag{4.31}$$

con

$$|\mathbf{d}(t)| \leq |A| \cdot (\mathbf{\Delta Q} + \varepsilon) \tag{4.32}$$

Reemplazando (4.26) con $\phi(t)$ y (4.31) con $\tilde{\phi}(t)$ y restando, resulta el sistema

$$\dot{\mathbf{e}}(t) = A\,\mathbf{e}(t) + \mathbf{d}(t) \tag{4.33}$$

con $\mathbf{e}(0) = 0$.

Luego, el Teorema 3.3 de [22] establece directamente la validez de (4.28) a partir de (4.33) y (4.32). □

Un corolario de este Teorema es que si la matriz A es estable, entonces la aproximación numérica da una solución finalmente acotada (no se puede garantizar estabilidad asintótica, pero siempre se garantiza estabilidad *práctica*).

Otro corolario es que la cota de error depende linealmente de la cuantificación.

[1] El símbolo "$\leq$" representa una desigualdad vectorial componente a componente. De manera similar, "$|\cdot|$" es el módulo por componentes de una matriz o vector.

4.4. Ejemplos

A continuación se presentarán dos ejemplos de simulación de sistemas stiff utilizando el método de BQSS.

4.4.1. Sistema lineal de segundo orden

Consideremos nuevamente el sistema lineal stiff de la Ec.(4.1) con condiciones iniciales $x_1(0) = 0$ y $x_2(0) = 20$.

Se realizó primero una simulación con BQSS usando quantum $\Delta Q_1 = \Delta Q_2 = 1$. Luego se repitió el experimento disminuyendo en 10 y 100 veces la cuantificación. En el primer caso, la simulación se completó con 20 y 22 pasos en x_1 y x_2 respectivamente. Para $\Delta Q_i = 0{,}1$ se realizaron 201 pasos en cada variable y para $\Delta Q_i = 0{,}01$, 2006 y 2024 pasos en cada variable. En todos los casos el tiempo final se seleccionó $t_f = 1000$, pero la simulación alcanzó una situación estable (sin realizar más pasos) antes de $t = 500$.

En la Fig.4.5 se observan las respuestas obtenidas con $\Delta Q_i = 1$ y 0,1 (no se incluyó la correspondiente a 0.01 por no ser apreciable a la vista la diferencia con la de $\Delta Q_i = 0{,}1$).

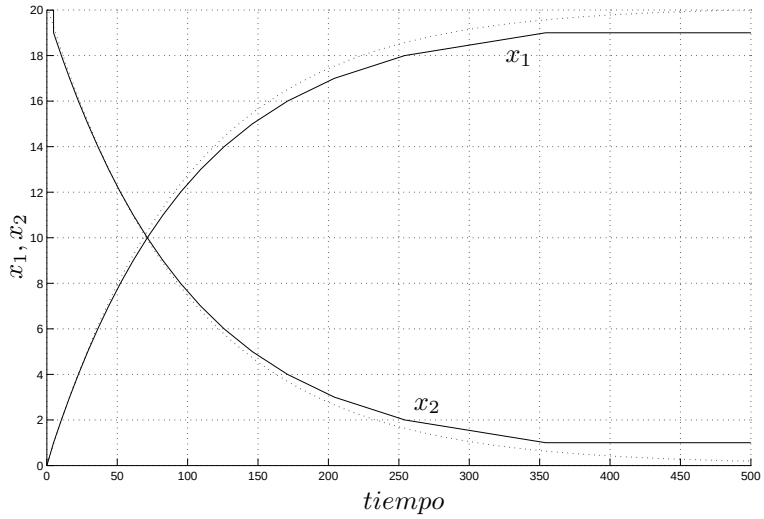

Figura 4.5: Respuesta del sistema lineal (4.1)

La cota de error global teórica, según lo calculado en el Teorema 3 para $\Delta Q_i = 1$ es de

$$e_1 \leq 3{,}004;\ e_2 = 5{,}001 \tag{4.34}$$

mientras que para $\Delta Q_i = 0{,}1$ es 10 veces menor y para $\Delta Q_i = 0{,}01$ es 100 veces menor. Sin embargo, puede verse en la gráfica que la diferencia entre la primera trayectoria y la última es siempre menor que 1 para ambas variables, por lo que el error en la primera simulación no puede ser mayor que 1,03 en x_1 y 1,05 en x_2, lo que muestra que la cota de error es en realidad bastante conservadora.

4.4.2. Sistema No lineal Stiff

El siguiente problema no lineal es un problema stiff estándar de prueba [10].

$$\begin{aligned} \dot{x}_1 &= -0{,}013x_1 - 1000x_1x_3 \\ \dot{x}_2 &= -2500x_2x_3 \\ \dot{x}_3 &= -0{,}013x_1 - 1000x_1x_3 - 2500x_2x_3 \end{aligned} \tag{4.35}$$

con condiciones iniciales $x_1(0) = 1$ $x_2(0) = 1$ $x_3(0) = 0$.

Para la simulación con BQSS se utilizó la cuantificación $\Delta Q_1 = 0{,}01$ $\Delta Q_2 = 0{,}01$ $\Delta Q_3 = 10^{-7}$ y un tiempo final de 1000.

Los resultados de la simulación pueden verse en la Fig.4.6. La misma se realizó con un total de 456 pasos (100 en x_1, 105 en x_2 y 251 en x_3), llegando a una situación estable en t=419.66. El tiempo total de cálculo fue de aproximadamente 6 milisegundos.

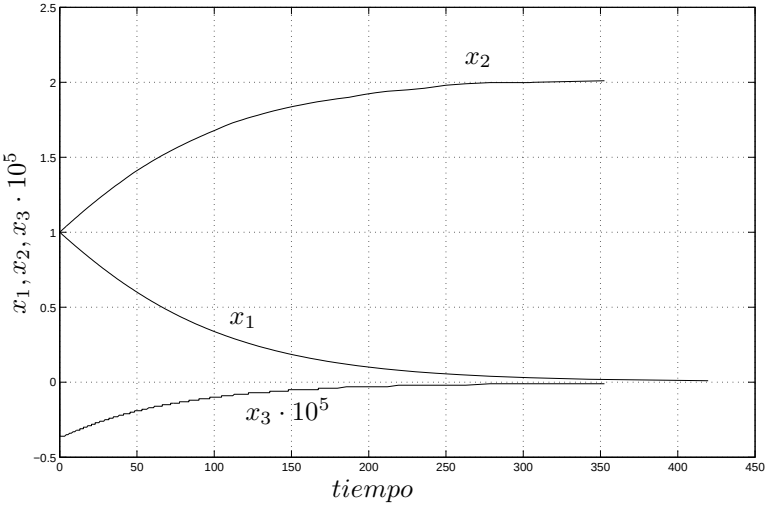

Figura 4.6: Respuesta del sistema no lineal (4.35)

Al simular el mismo sistema con los métodos implícitos de Matlab ode15s, ode23s y ode23tb (utilizando una tolerancia de 10^{-3}) se redujo el numero de pasos a 44, 27 y 27 respectivamente, pero el tiempo de cálculo se incrementó a aproximadamente 17 milisegundos (en modo compilado). La comparación de estos resultados con los de BQSS muestran que el error es del orden del quantum utilizado.

El incremento del tiempo de simulación en Matlab es atribuible al costo computacional de cada paso, en el cual debe resolverse una ecuación implícita, mientras que en BQSS cada paso sólo involucra cálculos escalares elementales. Si bien tanto PowerDEVS como Matlab ejecutan código C++ compilado, la forma en que lo hacen no es la misma, lo que relativiza esta comparación.

4.5. Sistemas Marginalmente Estables

Es sabido que una clase de sistemas que precisan un tratamiento especial en lo que a métodos de integración respecta son los denominados sistemas margi-

nalmente estables. Los mismos presentan la característica general de no ganar ni perder energía al transcurrir el tiempo, lo que se refleja en las curvas de las variables de estado en trayectorias con oscilaciones mantenidas (no se amortiguan ni se inestabilizan).

En esta sección, se presentara un método de integración por cuantificación que permite simular sistemas marginalmente estables obtenido combinando las ideas de QSS y BQSS. Una característica destacable de este método es que es el primer método adecuado para la simulación de este tipo de sistemas cuya implemetación es explicita, es decir, no precisa ni invertir matrices ni realizar iteraciones.

4.5.1. Método de CQSS

El siguiente sistema continuo de segundo orden representa un oscilador armónico:

$$\begin{aligned} \dot{x}_{a1} &= x_{a2} \\ \dot{x}_{a2} &= -x_{a1} \end{aligned} \tag{4.36}$$

Si se conocen las condiciones iniciales $x_{a1}(0)$ y $x_{a2}(0)$, se puede deducir fácilmente que la solución analítica es $x_{ai}(t) = c_i \sin(t) + d_1 \cos(t)$ con c_i y d_i constantes.

El oscilador armónico de la (4.36) pertenece a la categoría de los sistemas marginalmente estables, no gana ni pierde energía al pasar el tiempo.

Se pretende asegurar que la solución numérica también conserve la energía al simular este sistema, de manera que tengan sentido los resultados obtenidos.

Si uno intenta simular este sistema con cualquier método explicito de tiempo discreto tal como Forward Euler(FE), se obtiene como resultado oscilaciones que van incrementándose en amplitud, es decir, numéricamente inestables. Por otro lado, si se utiliza métodos implícitos tales como Backward Euler (BE), las soluciones obtenidas van decreciendo en amplitud a medida que el tiempo avanza (es decir soluciones ascintoticamente estables). Por otro lado, si se simula utilizando la regla trapezoidal, que avanza medio paso con BE y medio con FE, se obtiene un un resultado marginalmente estable para cualquier paso de integración. Cave aclarar que existen muchos métodos de tiempo discreto que permiten asegurar, al menos en el caso de sistemas lineales, que al simular sistemas marginalmente estables se obtendrán soluciones numéricas del mismo tipo. A esta familia de métodos se los denomina métodos F-estables, donde la característica principal de estos métodos es el borde de estabilidad de los mismos coincide con el eje imaginario del plano complejo $\lambda \cdot h$ [8] y además todos los métodos pertenecientes a esta familia son implícitos.

En el caso de los métodos de integración cuantificados, se obtienen resultados similares (ver fig. 4.7) a los comentados para los métodos clásicos . Si se simula un sistema marginalmente estable utilizando QSS, el mismo falla de manera similar a FE, es decir, las oscilaciones crecen en amplitud con el tiempo. Este fenómeno fue estudiado en [24],donde se muestra que el error crece linealmente con el tiempo. Por otro lado, si se utiliza como método de simulación BQSS, los resultados obtenidos padecen del mismo problema que se obtendría si se utilizase BE, es decir, se obtienen soluciones ascitoticamente estables.

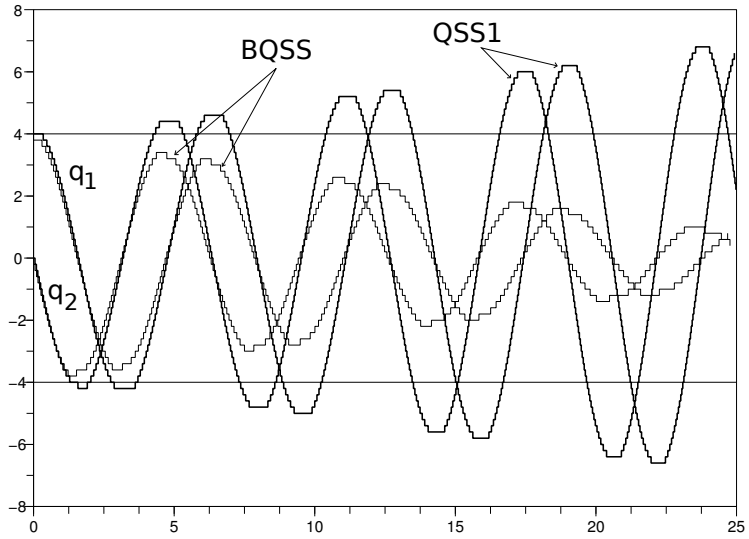

Figura 4.7: Simulación con QSS y BQSS del Oscilador(4.36)

En analogía a lo hecho con la regla trapezoidal al combinar FE con BE, se desarrollo un método denominado CQSS (QSS centrado) que combina QSS y BQSS. Este nuevo método posee la característica de preservar la energía permitiendo así simular sistemas marginalente estables.

La idea básica de este método consiste en utilizar el valor medio entre el q_i que utilizaría QSS y BQSS para el calculo de las derivadas de los estados x_i.

En la figura 4.8 se puede ver el comportamiento de esta nueva función de cuantificación. A su vez, esta figura permite comprender mas claramente el nombre otorgado a este nuevo método. Mientras que en QSS, el estado cuantificado q_i permanece siempre atrás del verdadero estado x_i, en BQSS, siempre esta por delante del mismo. CQSS resulta ser un compromiso de estas dos situaciones extremas.

El método de CQSS tiene en si la misma definición que QSS difiriendo únicamente en que q_i se mantiene centrado en los intervalos.

En la figura 4.9 puede verse observarse los resultados de simulación obtenidos usando CQSS, mostrando que el método puede ser utilizado perfectamente para simular el problema del oscilador armónico del problema (4.36) manteniendo la energía del sistema.

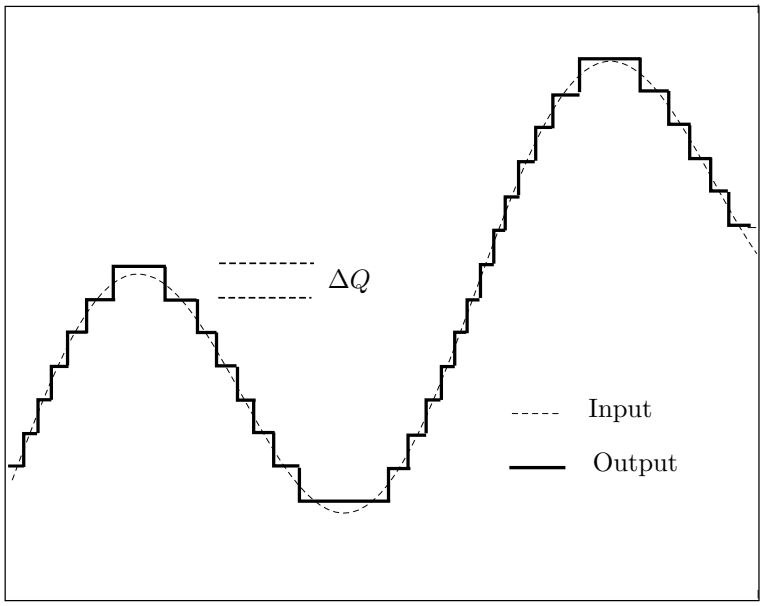

Figura 4.8: Función de cuantificación centrada

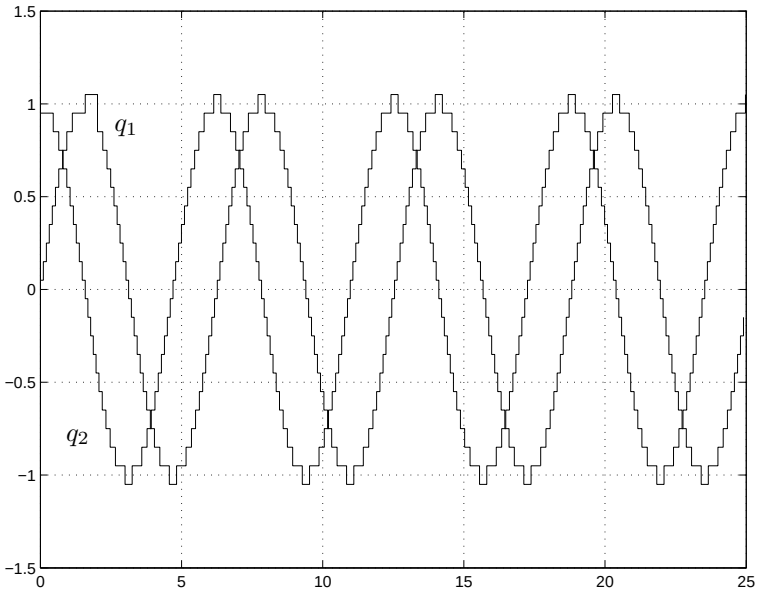

Figura 4.9: Simulación con CQSS del Oscilador(4.36)

4.5.2. Análisis de estabilidad en sistemas de segundo orden

El sistema de segundo orden lineal e invariante en el tiempo

$$\begin{aligned}\dot{x}_1(t) &= \alpha\, x_1(t) + \omega\, x_2(t)\\ \dot{x}_2(t) &= -\omega\, x_1(t) + \alpha\, x_2(t)\end{aligned} \tag{4.37}$$

tiene autovalores complejos conjugados $\lambda_{1,2} = \alpha \pm i\omega$.

Resulta asintoticamente estable si $\alpha < 0$, marginalmente estable se $\alpha = 0$, e inestable si $\alpha > 0$.

QSS y BQSS integran un sistema aproximado que tiene la siguiente forma

$$\begin{aligned}\dot{x}_1(t) &= \alpha\,(x_1(t) + \Delta x_1(t)) + \omega\,(x_2(t) + \Delta x_2(t)) + \Delta f_1\\ \dot{x}_2(t) &= -\omega\,(x_1(t) + \Delta x_1(t)) + \alpha\,(x_2(t) + \Delta x_2(t)) + \Delta f_2\end{aligned} \tag{4.38}$$

El termino de perturbación Δx_j se encuentra acotado según

$$|\Delta x_j| \leq \Delta Q_j + \varepsilon_j < 2\Delta Q_j \tag{4.39}$$

El termino Δf_j es normalmente cero excepto en BQSS, donde puede tomar valores de manera que anular la derivada. De acuerdo con la definición, Δf_1 puede ser distinto de cero solo cuando

$$\alpha\, \hat{q}_1(t) + \omega\, \hat{q}_2(t) = 0 \tag{4.40}$$

con

$$|\hat{q}_i - x_i| \leq \Delta Q_i + \varepsilon_i < 2\Delta Q_i, \tag{4.41}$$

En este caso:

$$-\Delta f_1 = \alpha(x_1 + \Delta x_1) + \omega(x_2 + \Delta x_2)$$

De esta ultima ecuación y de Eq.(4.40), podemos escribir

$$\Delta f_1 = \alpha(\hat{q}_1 - x_1 - \Delta x_1) + \omega(\hat{q}_2 - x_2 - \Delta x_2)$$

Luego, usando la ecuaciones (4.39) y (4.41), obtenemos

$$|\Delta f_1| < 4|\alpha|\Delta Q_1 + 4|\omega|\Delta Q_2 \tag{4.42}$$

Un análisis similar conduce a

$$|\Delta f_2| < 4|\omega|\Delta Q_1 + 4|\alpha|\Delta Q_2 \tag{4.43}$$

Entonces, podemos analizar la estabilidad de la (4.38) usando la candidata a función de Lyapunov

$$V(x) \triangleq \frac{1}{2}(x_1^2 + x_2^2). \tag{4.44}$$

cuya derivada con respecto al tiempo es

$$\dot{V}(x) = \alpha(x_1^2 + x_2^2) + x_1(\alpha\Delta x_1 + \omega\Delta x_2 + \Delta f_1) + x_2(\alpha\Delta x_2 - \omega\Delta x_1 + \Delta f_2)$$

entonces

$$\begin{aligned}\alpha\dot{V}(x) &= \alpha^2(x_1^2 + x_2^2) + x_1\alpha(\alpha\Delta x_1 + \omega\Delta x_2 + \Delta f_1) + x_2\alpha(\alpha\Delta x_2 - \omega\Delta x_1 + \Delta f_2)\\ &\geq \alpha^2\|x\|^2 - 5|\alpha| \cdot \|x\|(|\alpha| + |\omega|)(\Delta Q_1 + \Delta Q_2).\end{aligned}$$

Luego, dado que $\alpha \neq 0$ y

$$\|x\| > 5\left(1 + \left|\frac{w}{\alpha}\right|\right)(\Delta Q_1 + \Delta Q_2) \tag{4.45}$$

resulta que $\alpha \dot{V}(x) > 0$.

Cuando $\alpha < 0$, $\dot{V}(x)$ es negativa en todas las superficie de nivel donde se cumpla la Eq.(4.45). Luego, tanto con BQSS como con QSS se obtiene resultados que presentan *estabilidad* práctica. Es decir, la solución se mantienen en una región acotada en torno al origen.

En forma similar, cuando $\alpha > 0$, $\dot{V}(x)$ resulta positiva en las mismas superficies de nivel, y la norma Euclídea de x crece monótonamente, siempre que las condiciones iniciales satisfagan (4.45).

Entonces, ambos métodos son numéricamente estables para autovalores en el semiplano abierto de la izquierda e inestables si los autovalores están en el semiplano abierto de la derecha.

Veamos que ocurre en el caso en que $\alpha = 0$, es decir, cuando los autovalores se encuentran sobre el eje imaginario. En este caso, se puede ver que $\Delta f_i = 0$ en BQSS siempre que $\|x\| > 2\Delta Q_1 + 2\Delta Q_2$. Esto se debe a que la derivada de x_1 solo depende de q_2 y la derivada de x_2 solo depende del signo de q_1. De este modo, siempre podemos encontrar q_1 y q_2 en la dirección de las derivadas de x_1 y x_2.

Luego, para $\|x\|$ lo suficientemente grande, resulta que

$$\dot{V}(x) = x_1\omega\Delta x_2 - x_2\omega\Delta x_1 \tag{4.46}$$

usando (4.38) para reemplazar el termino $x_1\omega$ y $x_2\omega$ se tiene

$$\begin{aligned}\dot{V}(x) &= (-\dot{x}_2 - \omega\Delta x_1)\Delta x_2 - (\dot{x}_1 + \omega\Delta x_1)\Delta x_1 \\ &= -\dot{x}_1\Delta x_1 - \dot{x}_2\Delta x_2\end{aligned}$$

La definición de BQSS asegura que $\dot{x}_i\Delta x_i \geq 0$. Mas aún, la situación $\dot{x}_i\Delta x_i = 0$ sólo es posible cuando $\dot{x}_i = 0$. Entonces, la simulación de sistemas marginalmente estables utilizando BQSS producirá resultados que serán *estables prácticamente* ya que V (y por lo tanto $\|x\|$) decrecerá con el tiempo.

Para el caso de QSS, se precisa evaluar V entre dos instante de tiempo

$$V(t_b) - V(t_a) = \int_{t_a}^{t_b} \dot{V}(x)dt = \int_{t_a}^{t_b} (-\dot{x}_1\Delta x_1 - \dot{x}_2\Delta x_2)dt \triangleq -\Delta V_1 - \Delta V_2$$

Llamando $t_1, t_2, \cdots, t_m$ a los instantes de tiempo en el intervalo (t_a, t_b) donde x_1 alcanza su nivel de cuantificación. El primer termino ΔV_1 se puede expresar según

$$\Delta V_1 = \int_{t_a}^{t_1} \dot{x}_1\Delta x_1 dt + \sum_{k=1}^{m-1} \int_{t_k}^{t_{k+1}} \dot{x}_1\Delta x_1 dt + \int_{t_m}^{t_b} \dot{x}_1\Delta x_1 dt$$

Dado que $\Delta x_1 = q_1 - x_1$ y $q_1(t)$ se mantiene constante entre t_k y t_{k+1} se puede calcular

$$\int_{t_k}^{t_{k+1}} \dot{x}_1(q_1(t_k) - x_1(t))dt = \int_{x_1(t_k)}^{x_1(t_{k+1})} (q_1(t_k) - x_1)dx_1$$

Si t_k y t_{k+1} son instantes de tiempo en los que x_1 cruza el mismo nivel de cuantificación, la integral da 0. De lo contrario, $x_1(t_{k+1}) - x_1(t_k) = \pm\Delta Q_1$ y

$$\int_{t_k}^{t_{k+1}} \dot{x}_1 \Delta x_1(t)dt = q_1(t_k)(x_1(t_{k+1}) - x_1(t_k)) - \frac{1}{2}(x_1(t_{k+1})^2 - x_1(t_k)^2)$$
$$= (x_1(t_{k+1}) - x_1(t_k)) \cdot (q_1(t_k) - \frac{x_1(t_{k+1}) + x_1(t_k)}{2})$$

En QSS, se tiene que $q_1(t_k) = x_1(t_k)$, y la integral se evalúa hasta $-\Delta Q_1^2/2$. Por lo tanto, QSS resta este valor de ΔV_1 entre pasos consecutivos, excepto cuando x_1 cambia de dirección. Con un análisis similar se puede ver que ΔV_2 decrece en $-\Delta Q_2^2/2$ entre pasos consecutivos en x_2. en conclusión, V crece con el tiempo, por lo que el resultado que se obtiene es inestable.

Para lograr F-Estabilidad, se precisa la integral en la ultima ecuación sea cero. Esto se puede lograr modificando el método de QSS de manera que $q_i(t_k) = 0{,}5(x_i(t_{k+1}) + x_i(t_k))$, es decir, tomando para q_i el valor medio entre QSS y BQSS.

De este modo, queda probado que el método de CQSS mantiene la energía, al menos en sistemas de segundo orden como el mostrado.

La definición formal de CQSS y demostraciones más generales de sus características basados en la teoría de los métodos geométricos de integración pueden encontrarse en [5]. Estos resultados no se incluyen ya que forman parte del trabajo que está realizando Mario Bortolotto para su Tesis de doctorado.

4.6. Ejemplos

A continuación se presentarán dos ejemplos de simulación de sistemas marginalmenente estables utilizando el método de CQSS.

4.6.1. Péndulo

Uno de los ejemplos mas simples pero aún así ampliamente usados de sistemas marginalmente estables es la ecuación del péndulo sin fricción. Usando la segunda ley de Newton se puede escribir: $ml\ddot{\theta} = -mg\sin\theta - kl\dot{\theta}$ donde m es la masa de la bola,l es la longitud del brazo, θ es el ángulo entre la vertical y el brazo, g es la aceleración de la gravedad y k es el coeficiente de fricción. En nuestro caso simularemos el sistema sin pérdida de energía por tanto sin fricción ($k = 0$). Tomando como variables de estado $x_1 = \theta$ y $x_2 = \dot{\theta}$ podemos escribir las ecuaciones de estado.

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = -\frac{g}{l}\sin(x_1)$$

Tomando $g = 1$, $l = 1$ y partiendo con condiciones iniciales $x_1(0) = 1$ y $x_2(0) = 0$. El resultado de la simulación luego de $200seg$, utilizando el método de CQSS con quantum $\Delta Q = 0,01$ en todos los integradores es el mostrado en la figura 4.10.

Como se puede observar el resultado respeta el comportamiento oscilatorio del sistema conservándose la energía del sistema.

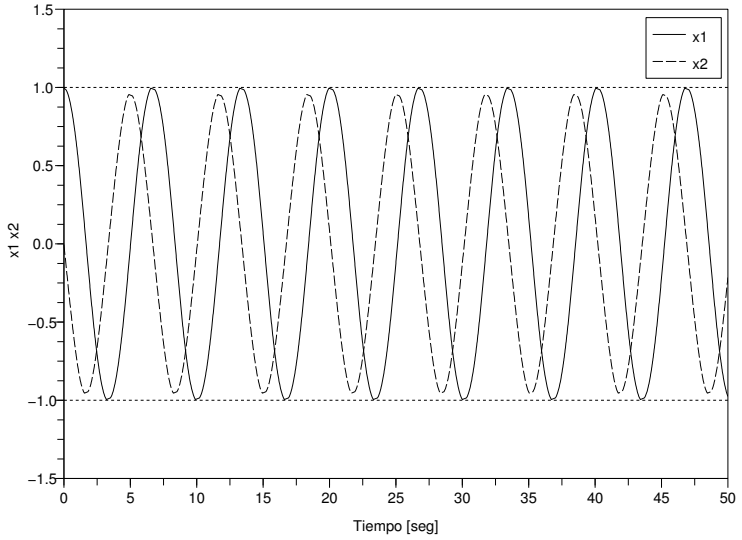

Figura 4.10: Simulación con CQSS del Péndulo

4.6.2. Sistema Stiff y Marginalmente Estable

El siguiente sistema de ecuaciones representa el modelo de una linea de transmisión sin perdida con una carga no lineal al final donde $L = C = 1$:

$$\begin{aligned}
\dot{\phi}_1(t) &= u_0(t) - u_1(t) \\
\dot{u}_1(t) &= \phi_1(t) - \phi_2(t) \\
&\vdots \\
\dot{\phi}_j(t) &= u_{j-1}(t) - u_j(t) \\
\dot{u}_j(t) &= \phi_j(t) - \phi_{j+1}(t) \\
&\vdots \\
\dot{\phi}_n(t) &= u_{n-1}(t) - u_n(t) \\
\dot{u}_n(t) &= \phi_n(t) - g(u_n(t))
\end{aligned} \tag{4.47}$$

Consideremos que la entrada es un pulso descripto por:

$$u_0(t) = \begin{cases} 10 & \text{if } 0 \leq t \leq 10 \\ 0 & \text{c.o.c} \end{cases} \tag{4.48}$$

que la carga no lineal es:

$$g(u_n(t)) = (10000 \cdot u_n)^3 \tag{4.49}$$

y que las condiciones iniciales son $u_i = \phi_i = 0,\ i = 1, \ldots, n$.

El sistema esta compuesto por 40 secciones LC (es decir, n=40), de manera que el orden del sistema es 80.

Si se linealiza en torno al origen ($u_i = \phi_i = 0$), se ve que el sistema es marginalmente estable (el modelo linealizado no presenta ningún termino de amortiguamiento). Además, si se hace un análisis más cuidadoso se puede ver que el sistema es stiff (la carga no lineal agrega un modo rápido cuando u_n crece).

De lo antes analizado, se puede concluir que el método F-estable CQSS es adecuado para simular el modelo dado por las ecuaciones (4.47)–(4.49). Para simular el modelo se utilizó en primer lugar quántum $\Delta Q_i = 0{,}1$ para todas las variables de estado excepto para u_n, donde se utilizó $\Delta Q = 1 \times 10^{-4}$.

El quantum en los métodos de QSS cumple el papel de tolerancia de error absoluta. Para el ultimo estado se utilizo un valor menor que el resto porque se quería ver con mayor resolución la evolución de esta variable dado que los valores que toma son cercanos a cero.

Para obtener los primeros 500 segundos de tiempo de simulación, CQSS realizó aproximadamente 6500 transiciones en cada una de las variables de estado. El resultado de los primeros 100 segundos de simulación pueden verse en la figura 4.11.

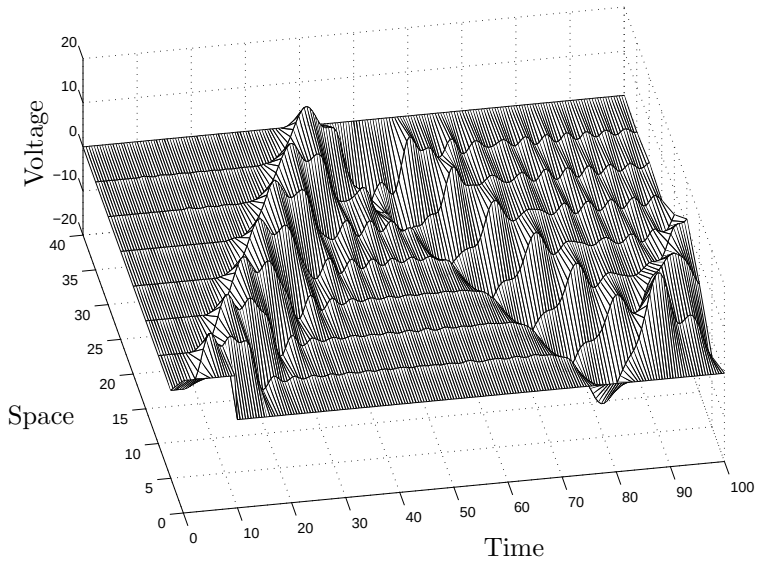

Figura 4.11: Simulación con CQSS del sistema representado por (4.47)–(4.49)

En la figura 4.12 puede verse el voltaje en la sección 35 de la linea de transmisión (es decir, cerca del final donde se encuentra la carga).

Los resultados obtenidos con CQSS son similares a los que se obtienen con el método ode15s de Matlab ® cuando se selecciona una tolerancia de error chica.

Otro punto que es interesante observar es el costo computacional, CQSS (con $\Delta Q_i = 0{,}1$) completa los 500 segundos de tiempo de simulación en 8.66 segundos, realizando aproximadamente 6500 pasos en cada variable de estado. Cada paso cambia el valor de dos componentes del lado derecho de la función de la ecuación (4.47). En consecuencia, cada componente de esta función es evaluada aproximadamente 13000veces. En otras palabras, toda la simulación

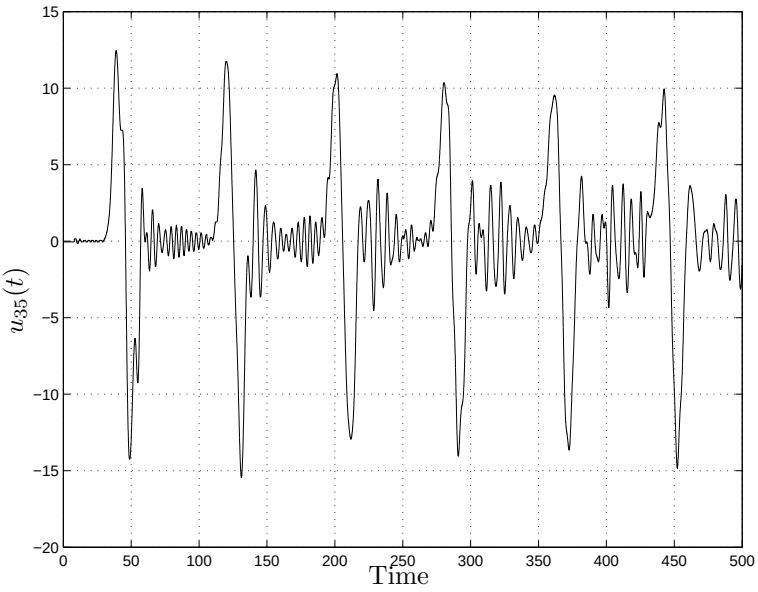

Figura 4.12: Simulación con CQSS del sistema representado por (4.47)–(4.49)

involucra aproximadamente 13000 evaluaciones completas de la función.

Realizando el experimento nuevamente pero con quantum $\Delta Q_i = 0{,}2$, el número de cálculos se reduce aproximadamente a la mitad y la simulación toma en total 4.55segundos. Cabe destacar que aun habiendo aumentado al doble ΔQ_i, la diferencia entre los dos resultados no pueden ser apreciados a simple vista.

Este sistema se simuló también con distintos métodos de integración en Matlab ® obteniéndose el mejor resultado al usar ode15s con una tolerancia relativa de 1×10^{-3} y una absoluta de 1×10^{-7}. En estas condiciones la simulación realizo 9683 pasos y tomó 8,37 segundos.

Los resultados obtenidos con ode15s resultaron más precisos que los obtenidos con CQSS. Este resultado era de esperar si se tiene en cuenta que ode15s es un método de quinto orden de paso variable mientras que CQSS es simplemente un método de primer orden.

4.7. Conclusiones

En este capítulo se presentaron dos métodos de integración basados en cuantificar los estados que surgen de combinar los principios básicos de los métodos implícitos con los de los métodos QSS.

El primero de estos métodos, denominado BQSS, es el primer algoritmo de QSS implícito que permite simular sistemas stiff de manera eficiente. Este método, comparte las propiedades de precisión de los métodos QSS anteriores.

Como característica mas sobresaliente, a pesar de tener una definición implícita, su implementación es en realidad explicita, por lo que se trata de un método explicito que no precisa reducir excesivamente el paso de integración para conservar la estabilidad numérica.

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 por lo cual está muy limitado en precisión. El segundo inconveniente reside en que introduce un término adicional Δf de perturbación en la Ec.(4.5) que en el caso de sistemas no lineales puede producir la aparición de falsos puntos de equilibrio.

La segunda limitación mencionada es imposible de sortear manteniendo la idea del método. En el caso de la primera, se buscó sin suerte obtener métodos BQSS de orden mayor combinando las ideas de BQSS con las de QSS2 y QSS3. A pesar de que no se consiguió obtener métodos BQSS de orden mayor, introduciendo un pequeño cambio se logró desarrollar una familia de métodos de integración que permiten simular sistemas stiff sorteando los problemas encontrados por BQSS. Esta nueva familia de métodos se desarrollará en los capítulos siguientes.

El segundo método presentado en este capítulo (CQSS), es un método basado en la cuantificación de los estados que permite simular sistemas marginalmente estables. Al igual que BQSS comparte la notable característica de poseer una implementación explicita. El método desarrollado es de primer orden por lo que, más allá de las fuertes propiedades teóricas que satisface, no es eficiente cuando se requiere precisión.

Si bien el método de CQSS es un resultado que surgió como consecuencia del trabajo de esta Tesis, su estudio en profundidad y desarrollo posterior escapan a los objetivos de la misma y por lo tanto están siendo abordados actualmente por otro integrante del grupo.

Capítulo 5

Métodos de QSS Linealmente Implícitos

El método de BQSS presentado en el capítulo 4 fue el primer método de integración por cuantificación que permitió la simulación eficiente de sistemas stiff. Sin embargo, dado que este método sólo realiza aproximaciones de primer orden, se encuentra muy limitado en precisión. Otro gran inconveniente de este método es que introduce un termino adicional de perturbación que no sólo incrementa la cota de error sino que además, puede producir la aparición de falsos puntos de equilibrio en algunos sistemas no lineales.

Para solucionar estos inconvenientes en este capítulo se introducirán tres nuevos métodos de integración que combinan la idea de BQSS con la de los métodos de integración linealmente implícitos por lo que se los denominó LIQSS (Linearly Implicit QSS). El primer método de esta familia (LIQSS1) realiza una aproximación de primer orden siguiendo la idea de BQSS pero, busca el valor de los estados para los cuales alguna derivada de los estados cruza por cero evitando así agregar términos adicionales de perturbación y generar la aparición de falsos puntos de equilibrio.

El segundo y tercer método que se presentan (LIQSS2 y LIQSS3) realizan aproximaciones de segundo y tercer orden combinando la idea de LIQSS1 con los principios de los métodos QSS2 y QSS3 respectivamente.

5.1. Método LIQSS1

5.1.1. Idea Básica de LIQSS1

La idea de LIQSS es muy similar a la de BQSS. Sin embargo, cuando no se puede encontrar un valor de q_j tal que x_j vaya hacia él, en lugar de agregar el termino de perturbación Δf_j que fuerza la condición $\dot{x}_j = 0$, LIQSS trata de encontrar el valor $\hat{q}_j$ para el cual $\dot{x}_j = 0$.

Para mostrar la idea, veamos paso a paso como se comporta el algoritmo sobre el ejemplo introductorio de la sección 4.1:

$$\begin{aligned} \dot{x}_1(t) &= 0{,}01\, x_2(t) \\ \dot{x}_2(t) &= -100\, x_1(t) - 100\, x_2(t) + 2020 \end{aligned} \tag{5.1}$$

Recordemos que los autovalores son $\lambda_1 \approx -0{,}01$ y $\lambda_2 \approx -99{,}99$ por lo que es claramente stiff y tomemos las mismas condiciones iniciales y la misma cuantificación de la sección 4.1, es decir $x_1(0) = 0$, $x_2(0) = 20$ y cuantificación $\Delta Q_1 = \Delta Q_2 = 1$.

En $t = 0$, podemos elegir $q_2 = 19$ o $q_2 = 21$ según cual sea el signo de $\dot{x}_2(t)$. En ambos casos , $\dot{x}_1(0) > 0$ por lo que el valor cuantificado futuro de x_1 será $q_1(0) = 1$.

Por otro lado, si elegimos $q_2(0) = 21$ entonces $\dot{x}_2(0) = -180 < 0$ y si elegimos $q_2(0) = 19$ resulta $\dot{x}_2(0) = 20 > 0$ por lo tanto, existe un punto $19 < \hat{q}_2(0) < 21$ en el cual $\dot{x}_2(0) = 0$. Se puede calcular (explotando la dependencia lineal de $\dot{x}_2$ con q_2) el valor de $\hat{q}_2(0)$ según

$$\hat{q}_2(0) = 21 - \frac{-180}{-100} = 19{,}2$$

Finalmente, las derivadas de los estados resultan: $\dot{x}_1(0) = 0{,}192$, $\dot{x}_2(0) = 0$.

El siguiente cambio en q_1 es agendado para $t = 1/0{,}192 \approx 5{,}2083$ mientras que el próximo cambio en q_2 se agendará para $t = \infty$

El siguiente paso es entonces en $t = 1/0{,}192 \approx 5{,}2083$. En ese instante $x_1 = 1$ y $x_2 = 0$. Luego tendremos $q_1(5{,}2083) = 2$ (ya que $\dot{x}_1(5{,}2083) > 0$). Si reevaluamos $\dot{x}_2$ para $q_2 = 19$ y $q_2 = 21$ resulta que en ambos casos es menor que cero, entonces el valor correcto es $q_2(5{,}2083) = 19$ porque de esta manera x_2 evoluciona hacia q_2.

Con estos valores de q_1 y q_2 se tiene $\dot{x}_1(5{,}2083) = 0{,}19$ y $\dot{x}_2(5{,}2083) = -80$. El próximo cambio en q_1 se agenda para $t = 1/0{,}192 + 1/0{,}19 \approx 10{,}47149$, mientras que el siguiente cambio en q_2 se agenda para $t = 1/0{,}192 + 1/80 \approx 5{,}22083$. Por lo tanto, el siguiente paso se da en $t = 5{,}22083$, cuando x_2 alcanza a q_2.

Los cálculos continúan de la misma manera. En la Fig.5.1 se puede ver la evolución de $q_1(t)$ y $q_2(t)$ hasta $t = 500$.

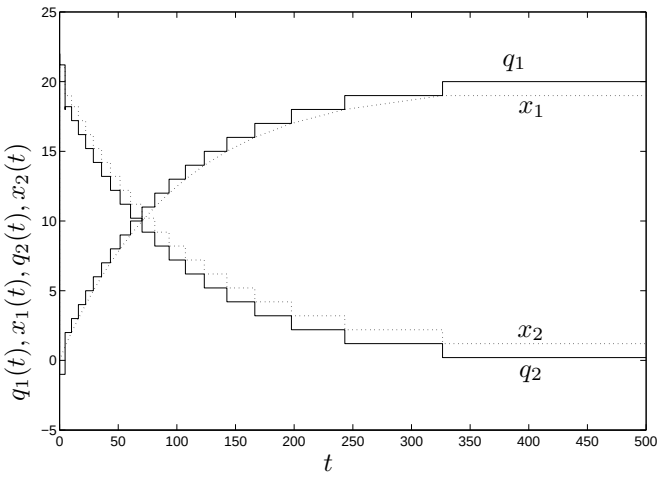

Figura 5.1: Resultado de la simulación con LIQSS1

Como puede verse, las oscilaciones rápidas de q_2 no están presentes al usar este método por lo que la simulación llevó sólo 46 cambios en total (21 cambios

en q_1 y 25 cambios en q_2) lo cual es un resultado más que bueno para un sistema stiff. Cabe recordar que el mismo sistema al simularse con QSS tomó 16016 pasos (ver 4.1)

5.1.2. Definición de LIQSS1

Dado el sistema

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \tag{5.2}$$

el método LIQSS1 lo aproxima por

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{q}(t), \mathbf{u}(t)) \tag{5.3}$$

donde cada q_j queda definido según la siguiente función:

$$q_j = \begin{cases} \overline{q}_j(t) & \text{si } \dot{x}_j(\overline{\mathbf{q}}^j) > 0 \wedge \dot{x}_j(\underline{\mathbf{q}}^j) > 0 \wedge \overline{q}_j(t^-) \neq \overline{q}_j(t) \\ \underline{q}_j(t) & \text{si } \dot{x}_j(\overline{\mathbf{q}}^j) \leq 0 \wedge \dot{x}_j(\underline{\mathbf{q}}^j) \leq 0 \wedge \overline{q}_j(t^-) \neq \overline{q}_j(t) \\ \widetilde{q}_j(t) & \text{si } \dot{x}_j(\overline{\mathbf{q}}^j) \cdot \dot{x}_j(\underline{\mathbf{q}}^j) < 0 \wedge \overline{q}_j(t^-) \neq \overline{q}_j(t) \\ q_j(t^-) & \text{c.o.c} \end{cases} \tag{5.4}$$

con $\overline{\mathbf{q}}^j = \mathbf{q}$ salvo que $q_j = \overline{q}_j$, $\underline{\mathbf{q}}^j = \mathbf{q}$ salvo que $q_j = \underline{q}_j$ y

$$\underline{q}_j(t) = \begin{cases} \underline{q}_j(t^-) - \Delta Q_j \\ \quad \text{si } x_j(t) - \underline{q}_j(t^-) \leq 0 \\ \underline{q}_j(t^-) + \Delta Q_j \\ \quad \text{si } x_j(t) - \underline{q}_j(t^-) \geq 2 \cdot \Delta Q_j \\ \underline{q}_j(t^-) \text{ c.o.c} \end{cases} \tag{5.5}$$

$$\overline{q}_j(t) = \underline{q}_j(t) + 2\Delta Q_j \tag{5.6}$$

$$\widetilde{q}_j(t) = \begin{cases} \overline{q}_j(t) - \frac{1}{A_{jj}} \cdot f_j(\overline{\mathbf{q}}^j(t), \mathbf{u}(t)) & \text{si } A_{jj} \neq 0 \\ q_j(t^-) & \text{c.o.c} \end{cases} \tag{5.7}$$

donde $A_{j,j}$ es la j, j componente de la matriz Jacobiana evaluada en $\overline{\mathbf{q}}^j$, es decir,

$$A_{jj} = \left.\frac{\partial f_j}{\partial x_j}\right|_{\overline{\mathbf{q}}^j, u(t^-)} \tag{5.8}$$

Como puede verse, cuando $A_{j,j} \neq 0$, usando $q_j = \widetilde{q}_j$ se tiene (en el caso lineal) $\dot{x}_j = 0$.

Cálculo de $\widetilde{q}_j$

Notar que se usa $\widetilde{q}_j$ cuando f_j cambia de signo según se use $\underline{q}_j$ o $\overline{q}_j$. En consecuencia, existe un punto intermedio $\hat{q}_j$ en el cual $f_j = 0$

Definiendo $\hat{\mathbf{q}}^j(t)$ igual a $\mathbf{q}^j(t^-)$, excepto que la j–ésima componente es $\hat{q}_j$,

$$A_j = \left.\frac{\partial f_j}{\partial x}\right|_{\mathbf{q}(t^-), u(t^-)}$$

y al residuo

$$g(\mathbf{x}, \mathbf{u}) = f_j(\mathbf{x}, \mathbf{u}) - A_j \mathbf{x},$$

llamando $\hat{q}_j$ al punto donde $f_j = 0$ podemos escribir

$$\begin{aligned} f_j(\mathbf{q}(t^-), \mathbf{u}) &= A_j \mathbf{q}(t^-) - A_{j,j} q_j(t^-) + A_{j,j} q_j(t^-) + g(\mathbf{q}(t^-), \mathbf{u}) \\ f_j(\hat{\mathbf{q}}^j(t), \mathbf{u}) &= A_j \hat{\mathbf{q}}^j(t) - A_{j,j} \hat{q}_j(t) + A_{j,j} \hat{q}_j(t) + g(\hat{\mathbf{q}}^j(t), \mathbf{u}) \end{aligned}$$

teniendo en cuenta que $A_j \hat{\mathbf{q}}^j(t) - A_{j,j} \hat{q}_j(t) = A_j \mathbf{q}(t^-) - A_{j,j} q_j(t^-)$ (ya que $\mathbf{q}(t^-)$ y $\hat{\mathbf{q}}^j$ solo difieren en la j–ésima componente) y considerando que $f_j(\hat{\mathbf{q}}^j, \mathbf{u}) = 0$, podemos resolver la ecuación anterior para $\hat{\mathbf{q}}^j$, obteniéndose

$$\hat{q}_j(t) = q_j(t^-) - \frac{f_j(\mathbf{q}(t^-), \mathbf{u})}{A_{j,j}} + \frac{g(\mathbf{q}(t^-), \mathbf{u}) - g(\hat{\mathbf{q}}^j, \mathbf{u})}{A_{j,j}} \tag{5.9}$$

Si f_j depende linealmente de q_j,entonces el último término de la Eq.(5.9) es cero y $\widetilde{q}_j = \hat{q}_j$, es decir, el valor de $\widetilde{q}_j$ hace que $\dot{x}_j = 0$.

En un caso no lineal, tendremos $\widetilde{q}_j \approx \hat{q}_j$, y $\dot{x}_j \approx 0$. A pesar de que en este caso las oscilaciones no van a desaparecer, las mismas serán de baja frecuencia.

Esto es análogo a lo que hacen los métodos linealmente implícitos de tiempo discreto al resolver la ecuación implícita teniendo en cuenta solo la parte lineal del problema. Debido a esta analogía es que se llamo al método LIQSS.

Estima de A_{jj}

Supongamos que en el instante t_1 cambia el valor de q_j y que ese cambio produce a su vez un cambio en $\dot{x}_j$. Entonces, podemos estimar A_{jj} según:

$$A_{jj} = \frac{f_j(\mathbf{q}(t_1^-), \mathbf{u}) - f_j(\mathbf{q}(t_1), \mathbf{u})}{q_j(t_1^-) - q_j(t_1)} \tag{5.10}$$

Notar que si f_j depende explícitamente de q_j, entonces la Ec.(5.10) da el valor exacto del Jacobiano.

5.1.3. Implementación en DEVS de LIQSS1

Como se dijo anteriormente, la diferencia principal entre LIQSS1 y QSS1 reside en el modo en el cual se obtiene $\mathbf{q}$ a partir de $\mathbf{x}$. La Eq.(5.4) nos muestra que q_j no sólo depende de x_j sino que además depende de $\mathbf{q}$.

Siguiendo la idea de la implementación en DEVS de QSS, es decir, mediante el acoplamiento de *integradores cuantificados* y *funciones estáticas*, pero teniendo en cuenta las diferencias mencionadas, podemos encontrar el modelo DEVS para el algoritmo LIQSS.

La estructura de su implementación se puede ver en la figura 5.2.

Dado que las trayectorias de $u_j(t)$ y $q_j(t)$ son seccionalmente constantes, las funciones estáticas son las mismas que las de QSS.

En consecuencia, sólo se debe definir el integrador LIQSS1 de manera que calcule q_j según la definición de LIQSS1.

Para poder construir el modelo DEVS del integrador cuantificado LIQSS, analizaremos primero su comportamiento.

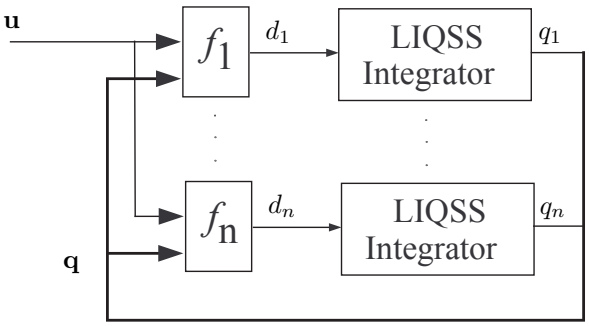

Figura 5.2: Diagrama en bloques de LIQSS1

Supongamos que en el instante de tiempo t el estado x_j alcanza al valor $\overline{q}_j$ con una pendiente positiva ($\dot{x}_j(t^-) > 0$). Entonces, se deben actualizar las funciones de cuantificación superior e inferior ($\overline{q}_j$ y $\underline{q}_j$) incrementándolas en ΔQ_j. Con los valores de $\underline{q}_j(t)$, $\overline{q}_j(t)$ y una estima de A_{jj} podemos prever cuanto resultaría $\dot{x}_j(t)$ evaluado en cada uno de ellos. Esta estima se puede realizar a partir de la siguiente ecuación:

$$\dot{x}_j(\overline{q}_j(t)) = \dot{x}(t^-) - A_{jj}q_j(t^-) + A_{jj}\overline{q}_j(t) \quad (5.11)$$

$$\dot{x}_j(\underline{q}_j(t)) = \dot{x}(t^-) - A_{jj}q_j(t^-) + A_{jj}\underline{q}_j(t) \quad (5.12)$$

donde A_{jj} puede ser fácilmente estimado a partir de los valores de $\dot{x}(t^-)$, $q_j(t^-)$.

Luego, si $\dot{x}_j(\overline{q}_j) > 0$ y $\dot{x}_j(\underline{q}_j) > 0$ tomamos $q_j = \overline{q}_j$ y generamos un evento de salida con el valor q_j. Contrariamente, si resulta $\dot{x}_j(\overline{q}_j) \leq 0$ y $\dot{x}_j(\underline{q}_j) \leq 0$ tomamos $q_j(t) = \underline{q}_j(t)$ y generamos un evento de salida con el valor $q_j(t)$.

Podría darse la situación en la cual los signos de $\dot{x}_j(\overline{q}_j)$ y $\dot{x}_j(\underline{q}_j)$ fuesen distintos. En este caso se puede asegurar por el teorema del valor medio que existe un valor $\widetilde{q}_j$ entre $\underline{q}_j$ y $\overline{q}_j$ en el cual $\dot{x}_j(\widetilde{q}_j) = 0$. Este valor puede ser estimado a partir de la ecuación (5.7) dando el valor exacto en los casos en que $\dot{x}_j$ depende linealmente de q_j. Tomamos entonces $q_j(t) = \widetilde{q}_j$ y generamos un evento de salida con el valor $q_j(t)$.

Además, la única situación en que se genera un evento de salida es cuando $x_j(t)$ alcanza ya sea a $\overline{q}_j(t)$ o a $\underline{q}_j(t)$.

Lo único que falta hacer una vez que se ha calculado q_j es agendar en que instante de tiempo se producirá el próximo evento. En cualquiera de estas situaciones, el tiempo σ_j hasta el próximo evento está determinado por el primer cruce x_j ya sea con $\overline{q}_j$ o $\underline{q}_j$, pudiéndose calcular como:

$$\sigma_j = \begin{cases} (\overline{q}_j(t) - x_j(t))/d_j & \text{si } d_j > 0 \\ (x_j - \underline{q}_j(t))/d_j & \text{si } d_j < 0 \\ \infty & \text{c.o.c} \end{cases}$$

Notar que una vez que se ha generado un evento de salida, no se generará ningún nuevo evento de salida en ese instante aunque se reciba un evento de entrada que cambie el signo de $\dot{x}_j$ El comportamiento descripto para el integrador cuantificado puede ser fácilmente traducido en el siguiente modelo DEVS correspondiente al integrador j-ésimo:

$$M = \langle X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$$

donde

$$X = Y = \mathbb{R} \times \mathbb{N};\ S = \mathbb{R}^4 \times \mathbb{R}^+$$

$$\begin{aligned}
\delta_{\text{int}}(s) &= \delta_{\text{int}}(dX, X, q, dq, \sigma) = \left(dX, \widetilde{X}, \widetilde{X}, dq_1, \sigma_1\right) \\
\delta_{ext}(s, e, u) &= \delta_{ext}(dX, X, q, dq, e, v) = \left(v, \hat{X}, q, dq, \sigma_2\right) \\
\lambda(s) &= \lambda(dX, X, q, dq, \sigma) = X + dX.\sigma + dq_1 \\
ta(s) &= \sigma
\end{aligned}$$

con

$$\begin{aligned}
\widetilde{X} &= X + dX.\sigma \\
\sigma_1 &= \begin{cases} \Delta Q/dX & \text{si } dX \neq 0 \\ \infty & c.o.c \end{cases} \\
dq_1 &= \begin{cases} \Delta Q & \text{si } dX > 0 \wedge A_{jj}.(X + dX.\sigma + \Delta Q + In) \geq 0 \\ -\Delta Q & \text{si } dX \leq 0 \wedge A_{jj}.(X + dX.\sigma - \Delta Q + In) \leq 0 \\ \frac{-In}{A_{jj}} - q & c.o.c \end{cases} \\
\hat{X} &= X + dx.e
\end{aligned}$$

donde $In = dX - A_{jj}.(q + dq)$,y σ_2 se calcula como la mínima solución positiva de

$$|X + dX.e + v.\sigma_2 - q| = \Delta Q$$

5.2. Método de LIQSS2

5.2.1. Idea básica de LIQSS2

Combinando las ideas de QSS2 y LIQSS1 se desarrolló un nuevo método de integración linealmente implícito denominado LIQSS2.

Las trayectorias de las variables cuantificadas de este nuevo método son seccionalmente lineales en lugar de seccionalmente constantes. Como en QSS2 buscaremos que las pendientes de las variables cuantificadas coincidan con las de las variables de estado correspondientes (en los instantes de cambio al menos). Por otro lado, mientras que en LIQSS1 se elegía q_j de manera de evitar que cambie el signo de la derivada $\dot{x}_j$ (buscando la condición $\dot{x}_j = 0$), en LIQSS2 elegiremos q_j de manera de evitar que cambie el signo de la derivada segunda $\ddot{x}_j$. En forma análoga a LIQSS1, buscaremos que se cumpla $\ddot{x}_j \cdot (q_j - x_j) \geq 0$.

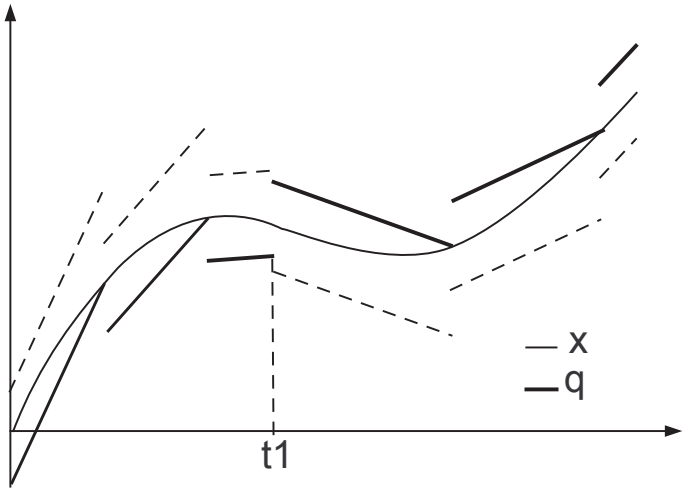

Figura 5.3: Trayectorias de LIQSS2

En la figura 5.3 se muestra un ejemplo general de la forma de las trayectorias siguiendo esta idea.

Para ilustrar el funcionamiento del método, volveremos al ejemplo de la Ec.(5.1).

$$\begin{aligned}\dot{x}_1(t) &= 0{,}01\, x_2(t)\\ \dot{x}_2(t) &= -100\, x_1(t) - 100\, x_2(t) + 2020\end{aligned}$$

cuya aproximación tendrá la forma

$$\begin{aligned}\dot{x}_1(t) &= 0{,}01\, q_2(t)\\ \dot{x}_2(t) &= -100\, q_1(t) - 100\, q_2(t) + 2020\end{aligned} \tag{5.13}$$

Tomando condiciones iniciales $x_1(0) = 0$ y $x_2(0) = 20$ y quántum $\Delta Q_1 = \Delta Q_1 = 0{,}1$ el algoritmo haría los siguientes pasos.

En $t = 0$ podríamos tomar $q_1 = 0{,}1$ o $q_1 = -0{,}1$ y $q_2 = 19{,}9$ o $q_1 = 20{,}1$. Tomaremos $q_1 = 0{,}1$ y $q_2 = 19{,}9$. Con estos valores obtenemos $\dot{x}_1 = 0{,}199$ y $\dot{x}_2 = 20$. Luego, elegimos las pendientes de las variables cuantificadas $\dot{q}_1 = 0{,}199$ y $\dot{q}_2 = 20$. Derivando la ec.(5.13) y utilizando las pendientes mencionadas resulta $\ddot{x}_1 = 0{,}2$ y $\ddot{x}_2 = -2019{,}9$. Por lo tanto los signos de las derivadas segundas son consistentes con la condición $\ddot{x}_j \cdot (q_j - x_j) \geq 0$.

La diferencia entre la primer variable de estado y su versión cuantificada es $q_1(t) - x_1(t) = q_1(0) - x_1(0) - \ddot{x}_1 t^2/2 = 0{,}1 - 0{,}1t^2$. Por lo tanto en $t = 1$ se tendrá la condición $q_1(t) = x_1(t)$ y agendaremos para ese instante el cambio en q_1. De manera similar, el cambio en la segunda variable quedará agendado para $t = 0{,}01$ por lo que el siguiente paso ocurrirá en dicho instante e involucrará a q_2.

En $t = 0{,}01$ tenemos $x_1 = 0{,}002$ y $x_2 = 20{,}1$. Además vale $q_1 = 0{,}10199$ y $\dot{q}_1 = 0{,}199$. En principio podemos elegir $q_2 = 20$ o $q_2 = 20{,}2$. Evaluando las

derivadas primeras y segundas como en el paso anterior, obtenemos para cada caso $\ddot{x}_2 = -1000$ y $\ddot{x}_2 = 1000$ respectivamente. Esto nos indica que existe un valor $\hat{q}_2$ entre 20 y 20,2 para el cual $\ddot{x}_2 = 0$. A calculo realizado resulta $\hat{q}_2 = 20{,}1$. Tomando entonces $q_2 = 20{,}1$ se obtiene de la ec.(5.13) los valores $\dot{x}_1 = 0{,}201$ y $\dot{x}_2 = -0{,}199$. Eligiendo luego $\dot{q}_2 = -0{,}199$ se tiene $\ddot{x}_1 = -0{,}00199$ y $\ddot{x}_2 = 0$.

Calculando como antes las diferencias $q_1(t) - x_1(t)$ y $q_2(t) - x_2(t)$ el siguiente cambio en q_1 se agenda para $t \approx 11{,}1$ mientras que el siguiente cambio en q_2 ocurriría recién en infinito (ya que $q_2(t)$ y $x_2(t)$ son paralelas).

Luego, tendremos el siguiente paso en $t = 11{,}1$ donde resultará $x_1 = 2{,}1071$, $x_2 = 17{,}8948$, y los cálculos continúan de la misma manera. Transcurridos 18 pasos en q_1 y 22 en q_2 la simulación llega al tiempo final $tf = 500$. Este número de pasos demuestra que el algoritmo pudo tratar muy bien la rigidez de este sistema.

En las figuras 5.4 y 5.5 se muestran los primeros pasos en cada variable de estado y en la figura 5.6 se gráfica el resultado completo de la simulación.

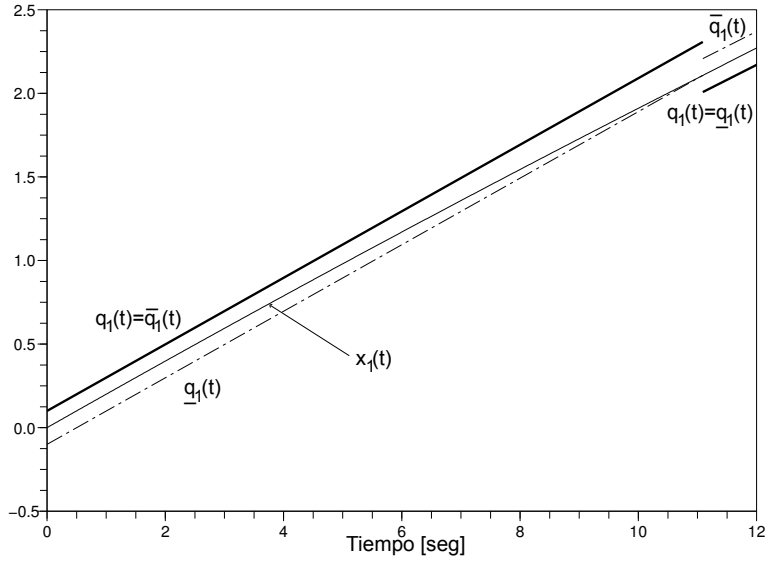

Figura 5.4: Resultado de simulación con LIQSS2 hasta $t = 12$ de x_1

En la figura (5.4) notar que en sólo 2 pasos x_1 avanzó 21 veces el valor del quántum. LIQSS1 hubiera necesitado al menos realizar 21 pasos.

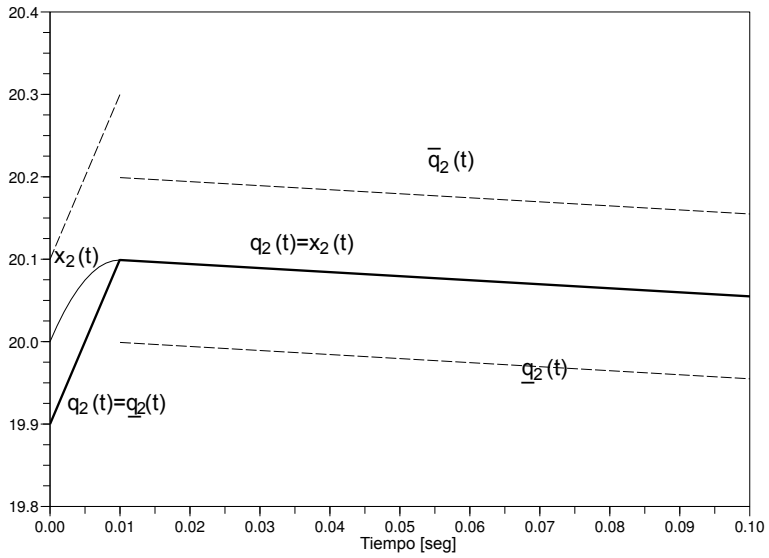

Figura 5.5: Resultado de simulación con LIQSS2 hasta $t = 0{,}1$ de x_2

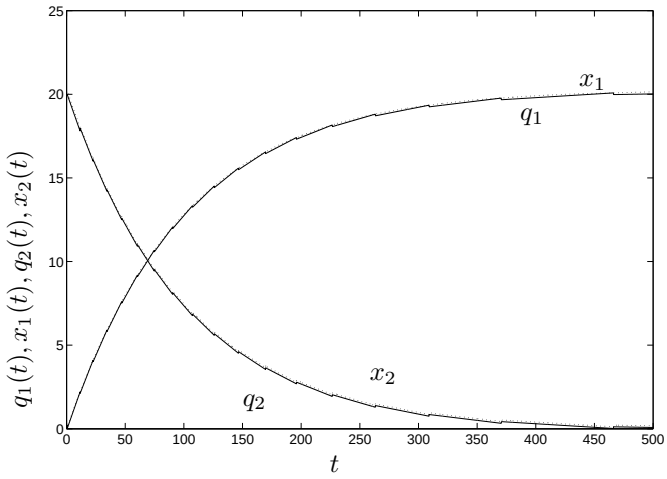

Figura 5.6: Resultado de simulación con LIQSS2

5.2.2. Definición de LIQSS2

Dado el sistema (5.2),

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

el método LIQSS2 lo aproxima por la ec.(5.3)

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

, donde cada componente q_j se define según:

$$q_j = \begin{cases} \overline{q}_j(t) & \text{si } \ddot{x}_j(\overline{\mathbf{q}}^j) > 0 \wedge \ddot{x}_j(\underline{\mathbf{q}}^j) > 0 \wedge \overline{q}_j(t^-) \neq \overline{q}_j(t) \\ \underline{q}_j(t) & \text{si } \ddot{x}_j(\overline{\mathbf{q}}^j) \leq 0 \wedge \ddot{x}_j(\underline{\mathbf{q}}^j) \leq 0 \wedge \overline{q}_j(t^-) \neq \overline{q}_j(t) \\ \widetilde{q}_j(t) & \text{si } \ddot{x}_j(\overline{\mathbf{q}}^j) \cdot \ddot{x}_j(\underline{\mathbf{q}}^j) < 0 \wedge \overline{q}_j(t^-) \neq \overline{q}_j(t) \\ q_j(t^-) & \text{c.o.c} \end{cases} \tag{5.14}$$

con $\overline{\mathbf{q}}^j = \mathbf{q}$ salvo que $q_j = \overline{q}_j$, $\underline{\mathbf{q}}^j = \mathbf{q}$ salvo que $q_j = \underline{q}_j$ y

$$\underline{q}_j(t) = \begin{cases} x_j(t_0) - \Delta Q_j & \text{si } t = t_0 \\ \underline{q}_i(t^-) + \Delta Q_j & \\ \quad \text{si } (x_j(t) = \underline{q}_j(t^-) + 2\Delta Q_j & \\ \underline{q}_i(t^-) - \Delta Q_j & \\ \quad \text{si } (x_j(t) = \underline{q}_j(t^-) & \\ \underline{q}_j(t_j) + mq_j \cdot (t - t_j) & \text{c.o.c} \end{cases} \tag{5.15}$$

$$\overline{q}_j(t) = \underline{q}_j(t) + 2 \cdot \Delta Q_j \tag{5.16}$$

$$\widetilde{q}_j(t) = \begin{cases} \dfrac{mq_j(t) - \ddot{x}(t^-)}{A_{j,j}} + q_j(t^-) & \\ \quad \text{si } A_{j,j} \neq 0 & \\ q_j(t^-) \quad \text{c.o.c} & \end{cases} \tag{5.17}$$

y

$$mq_j = \begin{cases} mq_j(t^-) \text{ si } q_j(t^-) = q_j(t) \\ A_{jj}q_j(t) + \dot{x}_j(t^-) - A_{jj}q_j(t^-) \text{ c.o.c} \end{cases} \tag{5.18}$$

Notar que en sistemas lineales la ecuación (5.18), hace que siempre sea $\dot{x}_j(t^+) = mq_j$. Además, si $A_{jj} = 0$, el valor $\widetilde{q}_j$ esta calculado de manera que $\ddot{x}_j(t^+) = 0$. En caso de tratarse de sistemas no lineales, los puntos así calculados serán valores muy próximos a los reales.

5.2.3. Implementación en DEVS del integrador LIQSS2

El esquema de simulación en el caso de LIQSS2 es el mismo que el mostrado anteriormente (Fig.5.2), pero en este caso las trayectorias de las variables de estado son seccionalmente parabólicas y las correspondientes a las variables cuantificadas son seccionalmente lineales.

Dado que LIQSS2 y QSS2 tienen la misma característica de las trayectorias de las variables cuantificadas (ambas seccionalmente lineales), LIQSS2 usa las mismas funciones estáticas que QSS2.

La principal diferencia entre LIQSS2 y QSS2 radica en el modo en que se calcula la trayectoria de las variables de estado cuantificadas en el integrador.

Cada segmento de trayectoria de variable cuantificada puede ser caracterizado por un punto inicial q_j y una pendiente mq_j. De la misma manera, la derivada de los estados puede ser caracterizada por el par (d_j, md_j). En conclusión, cada evento de entrada y de salida del integrador cuantificado puede ser caracterizado por dos valores.

Analicemos ahora el comportamiento del modelo DEVS resultante.

Supongamos que en el instante t el estado x_j alcanza ya sea a $\overline{q}_j$ o a $\underline{q}_j$ con $\ddot{x}_j(t^-) > 0$. Dado que x_j es seccionalmente parabólica, sabemos tanto el valor

de $\dot{x}_j(t^-)$ como el de $\ddot{x}_j(t^-)$. Entonces calculamos el nuevo valor de $\overline{q}_j$ y $\underline{q}_j$. A partir de los mismos y de una estima de A_{jj}, se puede prever cuanto resultaría $\ddot{x}_j$ si se tomase $q_j = \overline{q}_j$ o $q_j = \underline{q}_j$.

Para estimar cuanto valdría $\ddot{x}_j$ en función de esta elección, tengamos en cuenta que la pendiente mq_j de la variable cuantificada puede ser calculada para ambos valores de q_j a partir de la ecuación (5.18). Si llamamos $\overline{mq}_j$ a la pendiente correspondiente a $\overline{q}_j$ y $\underline{mq}_j$ a la correspondiente a $\underline{q}_j$, resulta

$$\begin{aligned} \ddot{x}_j(\overline{q}_j) &= \ddot{x}_j(t^-) - A_{jj} mq_j(t^-) + A_{jj}\overline{mq}_j \\ \ddot{x}_j(\underline{q}_j) &= \ddot{x}_j(t^-) - A_{jj} mq_j(t^-) + A_{jj}\underline{mq}_j \end{aligned}$$

Luego, si $\ddot{x}_j(\overline{q}_j) > 0$ y $\ddot{x}_j(\underline{q}_j) > 0$ tomamos $q_j = \overline{q}_j$, $mq_j = \overline{mg}_j$ y generamos un evento de salida con el par (q_j, mq_j). Por otro lado, si $\ddot{x}_j(\overline{q}_j) \leq 0$ y $\ddot{x}_j(\underline{q}_j) \leq 0$ entonces tomamos $q_j = \underline{q}_j$, $mq_j = \underline{mq}_j$ y generamos un evento de salida con el par (q_j, mq_j).

En caso de tener $\ddot{x}_j(\overline{q}_j)0$ y $\ddot{x}_j(\underline{q}_j)$ distinto signo, entonces existe un valor $\hat{q}_j$ entre $\overline{q}_j$ y $\underline{q}_j$ que hace que $\ddot{x}_j(\hat{q}_j) = 0$ este valor se puede calcular según

$$\hat{q}_j = -\frac{\ddot{x}_j(t^-) - A_{jj} mg_j(t-)}{A_{jj}^2} - \frac{\dot{x}_j(t^-) - A_{jj}(t^-) q_j(t^-)}{A_{jj}}$$

Tomamos entonces $q_j = \hat{q}_j$ y $mq_j = \hat{mq}_j$ con

$$\hat{mq}_j = A_{jj}\hat{q}_j + \dot{x}_j(t^-) - A_{jj} q_j(t^-)$$

y generamos un evento de salida con el par (q_j, mq_j).

Notar que con estos valores, en el caso de sistemas lineales, $\ddot{x}_j(t^+) = 0$ y $\dot{x}(t^+) = mq_j$

Además, la única situación en la que se genera un evento de salida es cuando un estado x_j alcanza ya sea a $\overline{q}_j$ o a $\underline{q}_j$.

Una vez que se ha calculado q_j, se debe agendar en qué instante se producirá el próximo evento. El tiempo hasta se produzca el próximo evento está dado por el primer cruce que se produzca de x_j ya sea con $\overline{q}_j$ o $\underline{q}_j$. El mismo se puede calcular como la mínima solución positiva σ_j de la siguiente ecuación

$$\begin{aligned} x_j(t) + \dot{x}_j(t)\cdot\sigma_j + \frac{1}{2}\ddot{x}_j(t)\sigma_j^2 &= \overline{q}_j \\ x_j(t) + \dot{x}_j(t)\cdot\sigma_j + \frac{1}{2}\ddot{x}_j(t)\sigma_j^2 &= \underline{q}_j \end{aligned}$$

En forma similar a lo hecho en LIQSS1, $A_{j,j}$ se puede calcular según:

$$A_{j,j}(t) = \frac{md_j(t^-) - md_j(t)}{mq_j(t^-) - mq_j(t)}$$

El comportamiento descrito para el integrador cuantificado puede ser fácilmente traducido en el siguiente modelo DEVS:

$$M = \langle X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$$

donde

$$X = Y = \mathbb{R}^2 \times \mathbb{N};\ S = \mathbb{R}^6 \times \mathbb{R}^+$$

$$\begin{aligned}
\delta_{\text{int}}(s) &= \delta_{\text{int}}(ddx, dx, x, q, mq, dq, \sigma) \\
&= \left(ddx, dx + 2ddx.\sigma, \widetilde{x}, \widetilde{x}, \widetilde{mq}, \widetilde{dq}, \sigma_1\right) \\
\delta_{ext}(s, e, u) &= \delta_{ext}(ddx, dx, x, q, mq, dq, \sigma, e, v, mv) \\
&= (mv/2, v, \hat{x}, \hat{q}, mq, dq, \sigma_2) \\
\lambda(s) &= \lambda(ddx, dx, x, q, mq, dq, \sigma) = (\widetilde{x} + \widetilde{dq}, \widetilde{mq}) \\
ta(s) &= \sigma
\end{aligned}$$

con

$$\begin{aligned}
\widetilde{x} &= x + dx.\sigma + ddx.\sigma^2 \\
\widetilde{mq} &= A_{jj}\left(\widetilde{x} + \widetilde{dq}\right) + In \\
\sigma_1 &= \begin{cases} \sqrt{|\Delta Q/ddx|} & \text{si } ddx \neq 0 \\ \infty & c.o.c \end{cases} \\
\widetilde{dq} &= \begin{cases} \Delta Q & \text{si } ddx_{est} \geq 0 \wedge mIn + A_{jj}.(dx + 2.ddx.\sigma) < 0 \\ -\Delta Q & \text{si } ddx_{est} \leq 0 \wedge mIn + A_{jj}.(dx + 2.ddx.\sigma) \geq 0 \\ \dfrac{-mIn/A_{jj} - In}{A_{jj}} - \widetilde{x} & c.o.c \end{cases} \\
\hat{x} &= x + e.dx + ddx.e^2 \\
\hat{q} &= q + mq.e
\end{aligned}$$

donde

$$\begin{aligned}
In &= dx + 2.ddx.\sigma - A_{jj}\left(q + dq + mq.\sigma\right) \\
mIn &= ddx - A_{jj}mq \\
ddx_{est} &= \begin{cases} -A_{jj}\left(A_{jj}\left(\widetilde{x} + dQ\right) + In\right) - mIn \\ \qquad\qquad \text{si } \ mIn + A_{jj}.(dx + 2.ddx.\sigma) < 0 \\ -A_{jj}\left(A_{jj}\left(\widetilde{x} - dQ\right) + In\right) - mIn \quad \text{c.o.c} \end{cases}
\end{aligned}$$

y σ_2 se calcula como la mínima solución positiva de

$$\left|\hat{q} + mq.\sigma_2 - \hat{x} - v.\sigma_2 - mv.\sigma_2^2/2\right| = \Delta Q$$

5.3. Método LIQSS3

5.3.1. Idea básica de LIQSS3

De manera similar a lo hecho al extender el método LQSS1 para obtener LIQSS2, combinando las ideas de QSS3 y LIQSS2 se desarrolló un nuevo método de integración linealmente implícito denominado LIQSS3.

Las trayectorias de las variables cuantificadas de este nuevo método son seccionalmente parabólicas en lugar de seccionalmente lineales de modo que pueden ser caracterizadas por la terna de valores (q,mq y pq). Como en QSS3 buscaremos que tanto las pendientes mq como las segundas derivadas de las variables cuantificadas (pq) coincidan con las de las variables de estado correspondientes (en los instantes de cambio al menos). Por otro lado, mientras que en LIQSS2 se elegía q_j de manera de evitar que cambie el signo de la derivada segunda $\ddot{x}_j$ (buscando la condición $\ddot{x}_j = 0$), en LIQSS3 elegiremos q_j de manera de evitar que cambie el signo de la derivada tercera $\dddot{x}_j$. En forma análoga a LIQSS1, buscaremos que en todo momento se cumpla $\dddot{x}_j \cdot (q_j - x_j) \geq 0$.

En la figura 5.7 se muestra un ejemplo general de la forma de las trayectorias siguiendo esta idea.

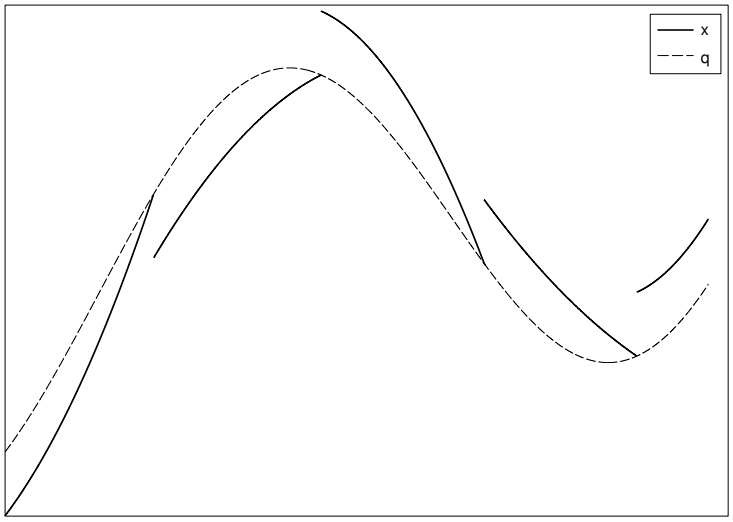

Figura 5.7: Trayectorias de LIQSS3

Al igual que con los demás métodos de esta familia, puede ocurrir que al comenzar una nueva sección de parábola en q_j, supongamos en t_a, cambie el signo de $\dddot{x}$. En tal caso, el teorema del valor medio nos garantiza que existe un valor pq_j en el cual $\dddot{x}(t_a) = 0$. En esta situación, además de tomar el valor pq_j que anula la tercera derivada, podemos tomar el valor inicial de $mq_j(t_a)$ de manera que $\ddot{x} = pq_j$ y el valor inicial de q_j de manera que $\dot{x} = mq_j$. Procediendo de esta manera se logra que la trayectoria del estado se mantenga siempre a la misma distancia de su versión cuantificada y de este modo, que no se generen mas eventos. Los valores de q_j, mq_j y pq_j pueden ser fácilmente obtenidos cuando $\dot{x}_j$ depende linealmente de q_j.

Si simulamos el sistema (5.1) con condiciones iniciales $x_1(0) = 0$ y $x_2(0) = 20$ y quántum $\Delta Q_1 = \Delta Q_1 = 0{,}01$, el método LIQSS3 realiza sólo 40 pasos, la misma cantidad que requirió simular el modelo con LIQSS2 utilizando un quántum 10 veces más grande. Los resultados de simulación pueden verse en la figura 5.8.

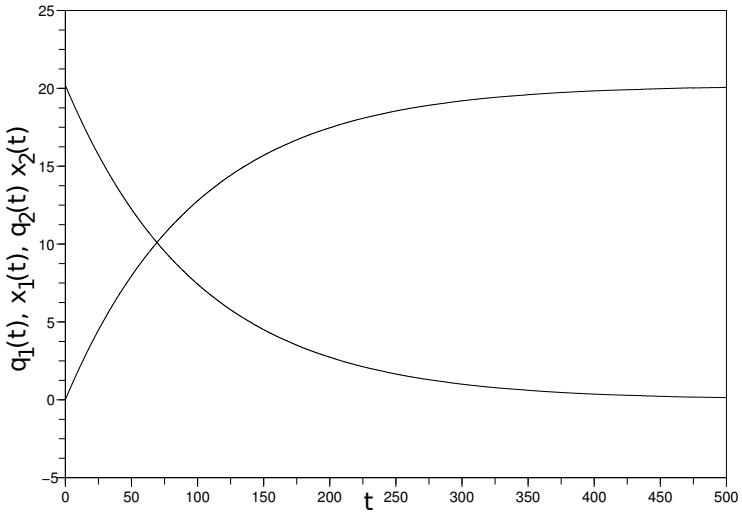

Figura 5.8: Resultado de simulación del sistema (5.1) con LIQSS3

5.3.2. Definición de LIQSS3

Dado el sistema de ecuaciones

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \tag{5.19}$$

el método LIQSS3 lo aproxima por

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{q}(t), \mathbf{u}(t)) \tag{5.20}$$

donde cada componente q_j se define según la siguiente función

$$q_j(t) = \begin{cases} \overline{q}_j(t) & \text{si } \ddot{x}_j(\overline{\mathbf{q}}^j) > 0 \wedge \ddot{x}_j(\underline{\mathbf{q}}^j) > 0 \wedge \overline{q}_j(t^-) \neq \overline{q}_j(t) \\ \underline{q}_j(t) & \text{si } \ddot{x}_j(\overline{\mathbf{q}}^j) \leq 0 \wedge \ddot{x}_j(\underline{\mathbf{q}}^j) \leq 0 \wedge \overline{q}_j(t^-) \neq \overline{q}_j(t) \\ \widetilde{q}_j(t) & \text{si } \ddot{x}_j(\overline{\mathbf{q}}^j) \cdot \ddot{x}_j(\underline{\mathbf{q}}^j) < 0 \wedge \overline{q}_j(t^-) \neq \overline{q}_j(t) \\ q_j(t^-) & \text{c.o.c} \end{cases} \tag{5.21}$$

con $\overline{\mathbf{q}}^j = \mathbf{q}$ salvo que $q_j = \overline{q}_j$, $\underline{\mathbf{q}}^j = \mathbf{q}$ salvo que $q_j = \underline{q}_j$ y

$$\underline{q}_j(t) = \begin{cases} x_j(t_0) - \Delta Q_j & \text{si } t = t_0 \\ \underline{q}_i(t^-) + \Delta Q_j & \text{si } (x_j(t) = \underline{q}_j(t^-) + 2\Delta Q_j \\ \underline{q}_i(t^-) - \Delta Q_j & \text{si } (x_j(t) = \underline{q}_j(t^-) \\ \underline{q}_j(t_j) + mq_j(t_j) \cdot (t - t_j) + pq_j(t_j) \cdot (t - t_j)^2 & \text{c.o.c} \end{cases} \tag{5.22}$$

$$\overline{q}_j(t) = \underline{q}_j(t) + 2 \cdot \Delta Q_j \tag{5.23}$$

$$\widetilde{q}_j(t) = \begin{cases} \dfrac{-pIn_j(t)}{A_{jj}^3} - \dfrac{mIn_j(t)}{A_{jj}^2} - \dfrac{In_j(t)}{A_{jj}} & \text{si } A_{j,j} \neq 0 \\ q_j(t^-) & \text{c.o.c} \end{cases} \tag{5.24}$$

y

$$mq_j(t) = \begin{cases} A_{jj}q_j(t) + In_j(t) & \text{si } (q_j(t^-) \neq q_j(t) \\ mq_j(t_j) + pq_j(t_j) \cdot (t - t_j) & \text{c.o.c} \end{cases} \quad (5.25)$$

$$pq_j(t) = \begin{cases} pq_j(t^-) & \text{si } (q_j(t^-) = q_j(t) \\ \frac{A_{jj} \cdot mq_j(t)}{2} + \frac{mIn_j(t)}{2} & \text{c.o.c} \end{cases} \quad (5.26)$$

Las funciones $In_j(t)$,$mIn_j(t)$ y $pIn_j(t)$ se calculan según

$$In_j(t) = \dot{x}(t^-) - A_{jj} \cdot q_j(t^-) \quad (5.27)$$

$$mIn_j(t) = \ddot{x}(t^-) - A_{jj} \cdot mq_j(t^-) \quad (5.28)$$

$$In_j(t) = \dddot{x}(t^-) - A_{jj} \cdot pq_j(t^-) \quad (5.29)$$

Para aclarar un poco más esta definición, observar que en (5.21), $q_j(t) = \widetilde{q}_j$ sólo se da cuando $\dddot{x}_j(t) = 0$ y $A_{jj} \neq 0$. Además, cuando $q_j(t) = \widetilde{q}_j(t)$, $pq_j(t)$ es tal que la tercera derivada de x_j resulta cero, $\dot{x}_j(t) = mq_j(t)$ y $\ddot{x}_j(t) = pq_j(t)$.

5.3.3. Implementación DEVS del integrador LIQSS3

$$M = \langle X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$$

donde

$$X = Y = \mathbb{R}^2 \times \mathbb{N};\ S = \mathbb{R}^{11} \times \mathbb{R}^+$$

$$\begin{aligned} \delta_{\text{int}}(s) &= \delta_{\text{int}}(dddx, ddx, dx, x, q, mq, pq, dq, In, mIn, pIn, \sigma) \\ &= \left(dddx, \widetilde{ddx}, \widetilde{dx}, \widetilde{x}, \widetilde{x}, \widetilde{mq}, \widetilde{pq}, \widetilde{dq}, \widetilde{In}, \widetilde{mIn}, \widetilde{pIn}, \sigma_1\right) \\ \delta_{ext}(s, e, u) &= \delta_{ext}(dddx, ddx, dx, x, q, mq, pq, dq, In, mIn, pIn, \sigma, e, v, mv, pv) \\ &= \left(pv/3, mv/2, v, \hat{x}, \hat{q}, \hat{mq}, pq, dq, \hat{In}, \hat{mIn}, \hat{pIn}, \sigma_2\right) \\ \lambda(s) &= \lambda(dddx, ddx, dx, x, q, mq, pq, dq, In, mIn, pIn, \sigma) \\ &= (\widetilde{x} + \widetilde{dq}, \widetilde{mq}, \widetilde{pq}) \\ ta(s) &= \sigma \end{aligned} \quad (5.30)$$

con

$$
\begin{aligned}
\widetilde{x} &= x + dx \cdot \sigma + ddx \cdot \sigma^2 + dddx \cdot \sigma^3 \\
\widetilde{dx} &= dx + 2 \cdot ddx \cdot \sigma + 3 \cdot dddx \cdot \sigma^2 \\
\widetilde{ddx} &= ddx + 3 \cdot dddx \cdot \sigma \\
\widetilde{mq} &= A_{jj} \left(\widetilde{x} + \widetilde{dq} \right) + \widetilde{In} \\
\widetilde{pq} &= \frac{A_{jj} \cdot \widetilde{mq} + \widetilde{mIn}}{2} \\
\sigma_1 &= \begin{cases} \sqrt[3]{|\Delta Q / dddx|} & \text{si } dddx \neq 0 \\ \infty & c.o.c \end{cases} \\
\widetilde{dq} &= \begin{cases} \Delta Q & \text{si } dddx_{est} \leq 0 \wedge \widetilde{ddx} \cdot A_{jj} + \widetilde{pIn} > 0 \\ -\Delta Q & \text{si } dddx_{est} \geq 0 \wedge \widetilde{ddx} \cdot A_{jj} + \widetilde{pIn} \leq 0 \\ \frac{-2 \cdot \widetilde{pIn}}{A_{jj}^3} - \frac{\widetilde{mIn}}{A_{jj}^2} - \frac{\widetilde{In}}{A_{jj}} - \widetilde{x} & c.o.c \end{cases} \\
\widetilde{In} &= In + mIn \cdot \sigma + pIn \cdot \sigma^2 \\
\widetilde{mIn} &= mIn + pIn \cdot \sigma \\
\hat{x} &= x + e.dx + ddx.e^2 + dddxe^3 \\
\hat{q} &= q + mq \cdot e + pq \cdot e^2 \\
\hat{mq} &= mq + 2 \cdot pq \cdot e \\
\hat{In} &= v - A_{jj} \cdot (\hat{q} + dq) \\
\hat{mIn} &= mv - A_{jj} \cdot \hat{mq} \\
\hat{pIn} &= pv - A_{jj} \cdot \hat{pq}
\end{aligned}
$$

donde

$$
dddx_{est} = \begin{cases} -A_{jj} \left(A_{jj} \left(A_{jj} (\widetilde{x} + dQ) + \widetilde{In} \right) + \widetilde{mIn} \right) / 2 - \widetilde{pIn} \\ \qquad \text{si } \widetilde{pIn} + A_{jj} \cdot \widetilde{ddx} > 0 \\ -A_{jj} \left(A_{jj} \left(A_{jj} (\widetilde{x} - dQ) + \widetilde{In} \right) + \widetilde{mIn} \right) / 2 - \widetilde{pIn} \\ \qquad \text{si } \widetilde{pIn} + A_{jj} \cdot \widetilde{ddx} \leq 0 \end{cases}
$$

y σ_2 se calcula como la mínima solución positiva de

$$
\left| \hat{q} + \hat{mq} \cdot \sigma_2 + \hat{pq} \cdot \sigma_2^2 - \hat{x} - v \cdot \sigma_2 - \frac{mv \cdot \sigma_2^2}{2} - \frac{pv \cdot \sigma_2^3}{3} \right| = \Delta Q
$$

5.4. Propiedades teóricas

Se tratarán en este punto las propiedades más importantes de los métodos LIQSS. En primer lugar se mostrará que estos métodos realizan un número finito de pasos en cualquier intervalo finito de tiempo, con lo cual se garantiza que la simulación siempre avanzará. Luego, se analizará las propiedades de estabilidad y precisión.

5.4.1. Legitimidad y Trayectorias de LIQSS

Uno de los requerimientos fundamentales de los métodos de QSS es la condición de legitimidad. La misma asegura que en cualquier intervalo finito de

tiempo siempre se realizarán una cantidad finita de pasos. En el siguiente teorema se prueba esta propiedad para el método LIQSS1.

Teorema 5.1. *Supongamos que en* (5.2) *la función* $\mathbf{f}$ *es acotada en el dominio* $D \times D_u$*, donde* $D \subset \Re^n$*,* $D_u \subset \Re^m$ *y asumamos que las trayectorias* $\mathbf{u}(t) \in D_u$ *son seccionalmente constantes. Entonces,*

1. *Cualquier solución* $\mathbf{x}(t)$ *de* (5.3) *es continua mientras* $\mathbf{q}(t)$ *permanezca en* D*.*

2. *La trayectoria* $\mathbf{q}(t)$ *es seccionalmente constante mientras se mantenga en* D*.*

Demostración. La demostración de (1) es trivial ya que, deacuerdo con la ecuación (5.3), la derivada de $\mathbf{x}$ es acotada.

En el caso del punto (2), para probar que $\mathbf{q}$ es seccionalmente constante se debe asegurar que sólo cambia una cantidad finita de veces en cualquier intervalo finito de tiempo.

Sea (t_1, t_2) un intervalo finito de tiempo cualquiera en el cual $\mathbf{q}(t)$ permanece en D. Se debe probar que $\mathbf{q}(t)$ cambia un número finito de veces dentro de ese intervalo de tiempo.

Las hipótesis del teorema aseguran que $\mathbf{f}(\mathbf{q}, \mathbf{u})$ es acotado. Luego, teniendo en cuenta la relación entre x_j y q_j, se tiene que existen constantes positivas $\bar{f}_j$ tales para todo $t \in (t_1, t_2)$:

$$|\dot{x}_j(t)| \leq \bar{f}_j; \text{ para } j = 1, \ldots, n.$$

Sea $t_c \in (t_1, t_2)$ y supongamos que $\overline{q}_j(t_c^-) \neq \overline{q}_j(t_c^+)$. De acuerdo a la ecuación (5.6), esta situación no se puede repetir hasta que $|x_j(t) - x_j(t_c)| \geq \Delta Q_j$. En consecuencia, el mínimo intervalo de tiempo entre dos discontinuidades en $\overline{q}_j(t)$ es

$$t_j = \frac{\Delta Q_j}{\bar{f}_j}$$

Entonces, si llamamos $\overline{n}_j$ al número de cambios de $\overline{q}_j(t)$ en el intervalo (t_1, t_2), el mismo resulta ser:

$$\overline{n}_j \leq (t_2 - t_1)\frac{\bar{f}_j}{\Delta Q_j}$$

Por otro lado, dado que $\mathbf{u}(t)$ es seccionalmente constante, realizará un número constante de cambios n_u en el intervalo (t_1, t_2).

Luego, la definición de q_j asegura que sólo puede cambiar cuando cambia $\overline{q}_j(t)$ o cuando hay algún cambio en alguna otra variable cuantificada o de entrada ($q_i(t)$ o $u_i(t)$) que cambia el signo de $\dot{x}_j$.

En conclusión, los cambios en $q_j(t)$ están ligados a cambios en alguna $\overline{q}_i(t)$ o $u_i(t)$. Entonces, el número total de cambios será menor o igual que la suma de todos los cambios en esas variables, es decir,

$$n_j \leq n_u + (t_2 - t_1)\sum_{i=1}^{n} \frac{\bar{f}_i}{\Delta Q_i}$$

el cual evidentemente es un número finito. □

A pesar de que este teorema es válido para el método LIQSS de primer orden, se puede obtener un resultado equivalente para los casos de segundo y tercer orden combinando esta demostración con la correspondiente de QSS2 [26] y QSS3 [19].

5.4.2. Representación Perturbada

Las propiedades teóricas de los métodos de QSS se basan en una representación perturbada del sistema original (5.2) que es equivalente a la aproximación de la ecuación (5.3).

Definiendo $\Delta\mathbf{x}(t) = \mathbf{q}(t) - \mathbf{x}(t)$ cada fila del sistema (5.3) se puede reescribir como:

$$\dot{x}_i = f_i(\mathbf{x}(t) + \mathbf{\Delta x}(t), \mathbf{u}(t)) \tag{5.31}$$

De las ecuaciones (5.4), (5.6) y (5.7), se puede asegurar que cada componente $\Delta x_i(t)$ de esta acotada por [1]

$$|\Delta x_i(t)| \leq 2 \cdot \Delta Q_i \tag{5.32}$$

donde ΔQ_i es el quántum adoptado para $x_j(t)$. De este modo, Los métodos LIQSS simulan un sistema aproximado que sólo difiere del original ((5.2)) en el término acotado de perturbación de los estados.

5.4.3. Estabilidad y Cota de error Global

Dado el sistema LTI

$$\dot{\mathbf{x}}_a(t) = A\mathbf{x}_a(t) + B\mathbf{u}(t) \tag{5.33}$$

donde A es una matriz Hurwitz con forma canónica de Jordan $\Lambda = V^{-1}AV$, la aproximación por cualquiera de los métodos LIQSS simula el sistema:

$$\dot{\mathbf{x}} = A(\mathbf{x}(t) + \Delta\mathbf{x}(t)) + B\mathbf{u}(t) \tag{5.34}$$

Definiendo ahora el error como $e(t) = x(t) - x_a(t)$, y siguiendo el modo de proceder de [26] y [8], resulta

$$|e(t)| \leq |V||\mathbb{R}\mathrm{e}(\Lambda)^{-1}\Lambda||V^{-1}|2\Delta Q \tag{5.35}$$

donde ΔQ es es el vector de quántum adoptado.

Como puede verse la cota de error obtenida es dos veces mayor que la de QSS, QSS2 y QSS3.

5.5. Ejemplos

5.5.1. Sistema Lineal de Segundo Orden

A lo largo de este capítulo se utilizó varias veces el siguiente sistema:

$$\begin{aligned} \dot{x}_1 &= 0{,}01x_2 \\ \dot{x}_2 &= -100x_1 - 100x_2 + 2020 \end{aligned} \tag{5.36}$$

[1] Los símbolos $|\cdot|$ y "$\leq$" representan respectivamente modulo y desigualdad componente a componente.

con condiciones iniciales $x_1(0) = 0$ y $x_2(0) = 20$ para introducir los métodos de integración desarrollados.

A continuación utilizaremos este mismo sistema pero analizando más profundamente los resultados obtenidos al simularlo utilizando los tres métodos de integración con $\Delta Q_i = 1$. Luego, se repetirán las simulaciones pero disminuyendo 10, 100 y 1000 veces el quántum. En la siguiente tabla se puede apreciar el numero de pasos realizado por cada uno de los métodos con los quántum mencionados:

ΔQ_i	LIQSS1			LIQSS2			LIQSS3		
	N°x_1	N°x_2	Total	N°x_1	N°x_2	Total	N°x_1	N°x_2	Total
1	21	25	46	5	8	13	-	-	-
0.1	201	203	404	18	22	40	12	12	15
0.01	2006	2026	4032	57	65	122	17	23	40
0.001	20064	28174	48238	182	204	386	36	46	82
0.0001	-	-	-	584	648	1232	79	99	178

En la tabla puede verse que usando LIQSS1, el número de pasos varía linealmente con la cuantificación, en LIQSS2 aumenta según la raíz cuadrada de la reducción de cuantificación y en LIQSS3 lo hace según la raíz cúbica.

En la figura 5.9 se ven los resultados obtenidos al simular el sistema usando Simulink, ODE15s y una tolerancia de 10^{-3} superpuestos con los de PowerDEVS usando LIQSS2 con $\Delta Q_i = 0{,}001$. Puede verse que utilizando estos parámetros de simulación no se puede apreciar a simple vista la diferencia entre ambos resultados.

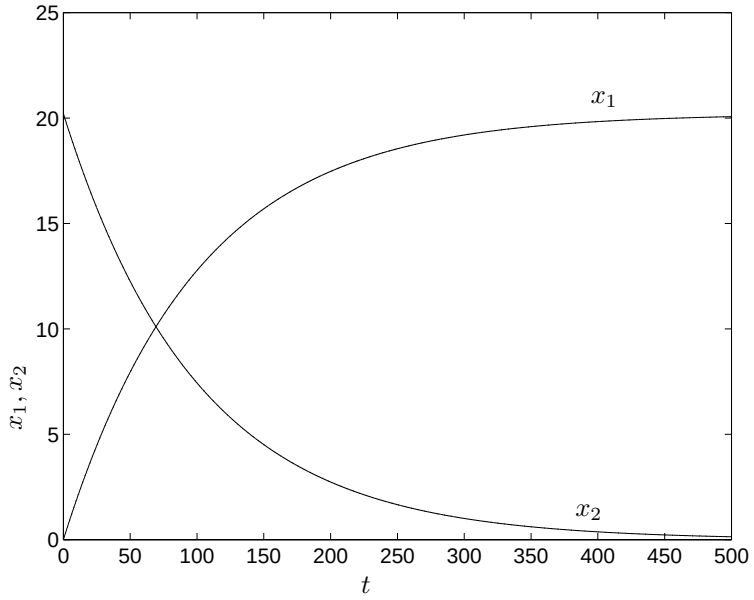

Figura 5.9: Simulación del Sistema Stiff lineal

El tiempo requerido por la simulación no pudo ser evaluado usando PowerDEVS bajo estas condiciones ya que resultó demasiado chico para poder ser medido con exactitud (más aún si se utilizaba LIQSS3). Por lo tanto, se compararon los tiempos de simulación usando LIQSS2 con quántum $\Delta Q_i = 0{,}0001$. El

tiempo requerido por Simulink resultó ser de 0,078 seg (ODE15s, modo acelerado y tolerancia 10^{-3}), mientras que a PowerDEVS le tomó solo 0,015 segundos.

Gracias a la cota de error global –Eq.(5.35)– se puede asegurar que el error en la simulación hecha con LIQSS2 es siempre menor que $2 \cdot 10^{-4}$ en x_1 y $6 \cdot 10^{-4}$ en x_2.

En la figura (5.10) se puede ver el error en la simulación usando LIQSS2 en PowerDEVS con quántum $\Delta Q_i = 0{,}0001$ (realizado haciendo la diferencia entre la solución analítica y la obtenida por simulación). Si se observa esta gráfica y se tiene en cuenta la cota teórica calculada, se ve claramente que esta última es bastante conservativa.

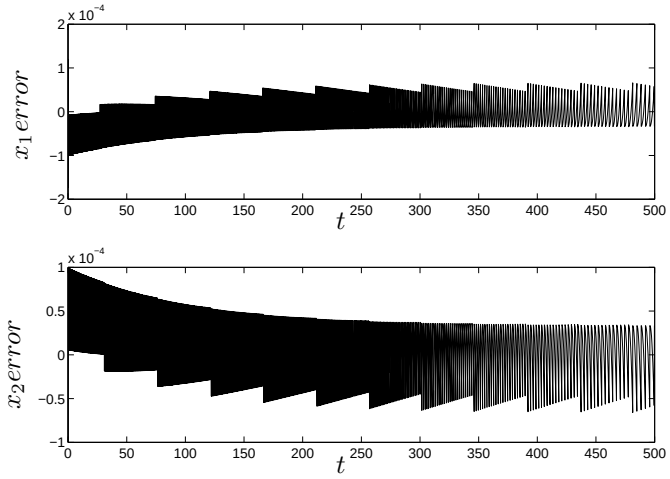

Figura 5.10: Error

5.5.2. Van der Pol Oscillator

El problema propuesto por B.Van der Pol en 1920 consiste en una ecuación diferencial de segundo orden que describe el comportamiento de los circuitos no lineales de un tubo de vacío. La respuesta de este sistema consiste en dos soluciones periódicas, una solución constante $z(t) = 0$ que es inestable y una solución periódica no trivial que corresponde a un ciclo límite.

La ecuación depende de un parámetro que pesa el nivel de influencia que tiene sobre la misma la parte no lineal. Las ecuaciones de estado correspondientes a este sistema son:

$$\begin{aligned} \dot{x}_1(t) &= x_2 \\ \dot{x}_2(t) &= (1 - x_1^2) \cdot x_2 \cdot \mu - x_1 \end{aligned} \tag{5.37}$$

Tomando $\mu = 1000$ es modelo tiene un comportamiento stiff, y es usado frecuentemente como un problema para testear métodos de integración para sistemas stiff [10].

Este modelo de armó en PowerDEVS (Fig.5.11)

Se usó como condiciones iniciales para la simulación $x_1(0) = 2$, $x_2(0) = 0$ y quántum $\Delta Q_1 = 0{,}001$, $\Delta Q_2 = 1$. Utilizando LIQSS2 para simular el sistema hasta $t = 4000$, el tiempo total de simulación requerido es 0,031, involucrando

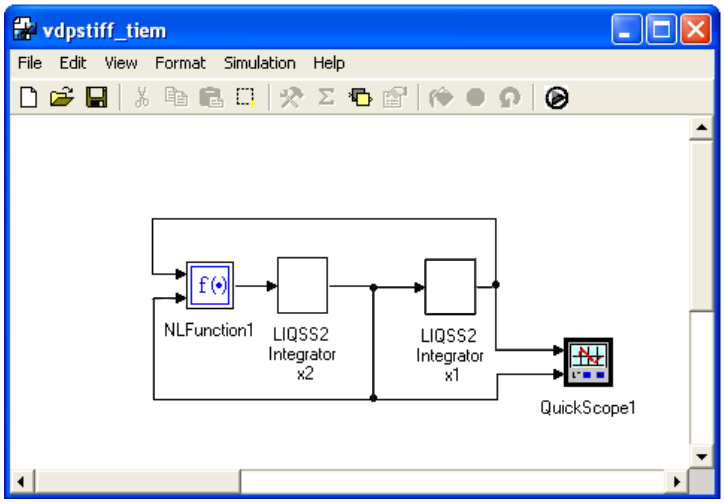

Figura 5.11: Modelo PowerDEVS del oscilador de Van der Pol

un total de 2159 pasos (838 en x_1 y 1321 en x_2). La figura 5.12 muestra el resultado de simulación obtenido.

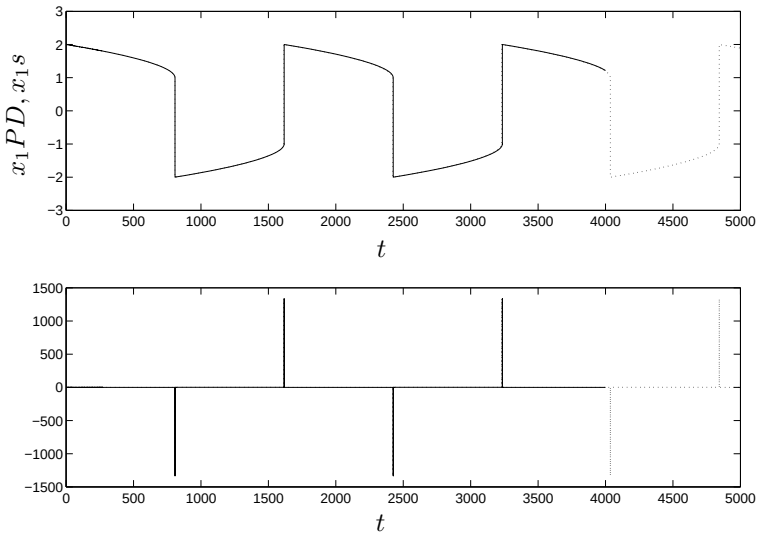

Figura 5.12: Simulación del Oscilador de Van der Pol

Este mismo sistema se simuló con Matlab/Simulink obteniéndose el mejor resultado al usar el método ode15s. Además, para obtener resultados similares a los obtenidos con PowerDEVS, se debe fijar una tolerancia de error de 10^{-14}. En caso de usarse tolerancias de error mayores, los resultados obtenidos son cualitativamente parecidos pero presentan un error significativo de fase. En este caso, la simulación tomó 0,056 requiriéndose 2697 pasos.

El sistema se volvió a simular utilizando LQSS2 pero con quántum 10 veces

más chico ($\Delta Q_1 = 0{,}0001$ y $\Delta Q_2 = 0{,}1$). Bajo estas condiciones la simulación se completó con un total de 4148 pasos (1975 en x_1 y 2173 en x_2) requiriendo 0.047 segundos. Si se compara el número de pasos obtenido bajo estas condiciones con el obtenido anteriormente puede verse que sólo se incrementó 2 veces mientras que los niveles de cuantificación se redujeron en un factor de 10. De esta manera, se pone nuevamente de manifiesto que LIQSS2 es en efecto un método de segundo orden.

5.6. Conclusiones

En este capítulo se presentaron tres métodos de integración basados en cuantificar los estados que, haciendo uso de los principios de los métodos linealmente implícitos, permiten simular sistemas stiff de manera eficiente.

Al igual que BQSS, los métodos LIQSS a pesar de poseer una definición implícita su implementación es explicita. De este modo, gracias a que estos métodos no precisan realizar iteraciones se logra una gran reducción del costo computacional con respecto a los métodos implícitos de tiempo discreto tradicionales.

Se mostró que estos métodos, al igual que los métodos de QSS satisfacen una cota de error global

Los métodos LIQSS mejoraron ampliamente la performance del método BQSS gracias a que estos método han podido ser desarrollados hasta orden tres y gracias a que los mismos, no agregan el término de perturbación Δf que adiciona BQSS al aproximar el sistema continuo por (4.5), eliminando de esta manera la aparición de falsos puntos de equilibrio.

Además de las ventajas obtenidas por los métodos LIQSS en la simulación de sistemas stiff continuos, al igual que los métodos de QSS, los mismos presentan grandes ventajas al simular sistemas discontinuos. El manejo de discontinuidades no requiere ningún trato especial ya que todos los métodos basados en la cuantización de los estados son de por sí sistemas discontinuos. De este modo, el tratamiento de discontinuidades es intrínseco de estos métodos. Esta propiedad, permite lograr mayor eficiencia sobre los métodos tradicionales ya que no precisan iterar para encontrar el punto exacto de ocurrencia de las discontinuidades. Más adelante trataremos en profundidad este aspecto.

Como limitación, estos métodos sólo logran ser efectivos, cuando la estructura del sistema es tal que la rigidez del sistema se manifiesta en algunas variables de estado x_j que evolucionan mucho más rápido que las demás variables del sistema y cuya derivada $\dot{x}_j$ depende explícitamente del propio valor del estado x_j. Esto pude verse en sistemas que presentan componentes A_{jj} en el jacobiano del sistema que son mucho más grandes que las demás componentes A_{ji}. En el capítulo 6 se mostrarán 3 métodos que aún están en etapa de investigación que tratan con esta limitación de los métodos LIQSS.

Uno de los campos de aplicación donde estos métodos presentan grandes ventajas en la simulación de modelos de circuitos de electrónica de conmutación donde son muy comunes las discontinuidades debidas a la operación discontinua de componentes o conmutaciones de llaves. Este tipo de discontinuidades, en principio pueden generar cambios en la estructura del modelo y/o cambios en parámetros del mismo. Por ejemplo, se puede considerar que un diodo al estar 'cortado' desconecta dos puntos de un sistema, provocando de esta manera un

cambio de estructura o se puede considerar que pasa de tener una impedancia muy baja a una muy alta, cambiándose de este modo solo el parámetro correspondiente a la impedancia.

Los entornos de simulación que usan modelado basado en objetos tales como Dymola, evitan siempre que se produzcan cambios en la estructura del sistema ya que eso implica no solo tener que encontrar el punto donde se pasa de un modelo al otro sino además, poder apartir de ecuaciones elementales asociadas a cada componente del modelo, volver a construir un nuevo sistema de ecuaciones diferenciales. Para lograr este cometido, tratan este tipo de problemas utilizando modelos realistas de los componentes en lo que se hace es pasar de un estado de alta impedancia a uno de baja impedancia o viceversa. Así, aunque se logra evitar los cambios de estructura, se traslada el problema a los métodos de integración ya que se obtienen modelos stiff de estructura constante discontinuos en sus parámetros.

Gracias a que, como se comento anteriormente, los métodos LIQSS son adecuados para sistemas stiff y el manejo de discontinuidades no representa ningún inconveniente para ellos, se verá en el capitulo siguiente a través de algunos ejemplos como estos métodos son muy adecuados para la simulación de modelos de electrónica de conmutación.

Capítulo 6

Simulación de Circuitos de Electrónica Conmutada

En este capítulo se presentarán y analizarán tres ejemplos de simulación de modelos de circuitos de electrónica conmutada. El primero de estos ejemplos consiste en un convertidor Buck sin carga, el segundo es un control de velocidad de un motor alimentado por una fuente Buck y el tercero es una cadena de inversores lógicos. En cada uno de estos ejemplos analizará el desempeño de los métodos LIQSS comparándolo con distintos métodos de tiempo discreto.

Además de analizar los ejemplos particulares que se presentan, a través del análisis de los mismo se inferirán conclusiones de carácter más general respecto al tipo de modelos en los cuales los métodos LIQSS presentan ventajas.

6.1. Convertidor Buck

En esta sección se presentará el modelo de un convertidor Buck y se mostrarán resultados de simulación del modelo utilizando distintas herramientas de simulación.

Los convertidores reductores Buck son parte integral de muchos circuitos electrónicos actuales y constituyen, desde el punto de vista electrónico, uno de los circuitos convertidores más simples. El motivo por el cual se denominan convertidores reductores es porque su tensión de salida nunca puede ser mayor que la de entrada. Los mismos permiten reducir un voltaje continuo generalmente no regulado a otro de menor magnitud regulado. En la figura 6.1 se muestra la topología de este convertidor. En la misma se puede ver la presencia del dispositivo de conmutación Sw, un diodo D (componente discontinuo), un inductor L, un capacitor C y una resistencia R.

Los convertidores de alimentación conmutados son circuitos discontinuos y no lineales.

Si se considera que los componentes son ideales, se tiene que, según sea el estado de los dispositivos de conmutación, se llega a distintos modelos. Consideremos por ejemplo el caso de cambios en el estado del diodo D.

Si la llave está abierta y el diodo está conduciendo (idealmente un cortocircuito), el circuito resultante puede ser modelado a partir del siguiente conjunto de ecuaciones de estado donde $x_1 = I_L$ y $x_2 = dI_L/dt$:

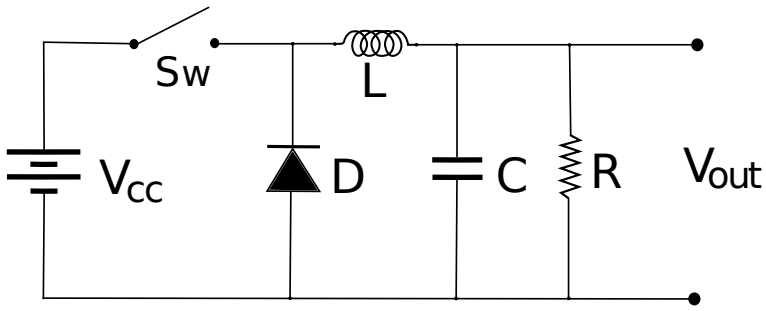

Figura 6.1: Esquema circuital de la fuente Buck

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\frac{1}{LC}x_1 - \frac{1}{RC}x_2 \\ v_{out} &= -L\,x_2 \end{aligned}$$

por otro lado, cuando la llave está abierta y el diodo está cortado (idealmente circuito abierto) resulta la siguiente ecuación de estado donde $x_1 = V_C$

$$\begin{aligned} \dot{x}_1 &= -\frac{1}{RC}x_1 \\ v_{out} &= x_1 \end{aligned}$$

como puede verse, al considerarse los elementos de conmutación como ideales, para cada una de las condiciones de los mismos existen distintas topologías que describen el comportamiento del circuito, obteniéndose así modelos que difieren en estructura (en el caso de mantenerse abierta la llave y conmutarse la condición del diodo pudo verse que cambió el orden del modelo).

Este tipo de cambios de estructura presenta una gran dificultad a los entornos de simulación orientados a objetos (como Dymola y demás herramientas basadas en Modelica). Por este motivo, este tipo de programas agregan realismo a los componentes lográndose así que no se den cambios de estructura. En el caso del ejemplo presentado, si consideramos al diodo como una resistencia cuyo valor es muy alto cuando está cortado (R_{D-off}) y muy bajo cuando está conduciendo (R_{D-on}) y en forma similar consideramos a la llave como una resistencia muy alta al estar abierta (R_{LL-off}) y muy bajo cuando está cerrada (R_{LL-on}), se puede obtener el siguiente modelo DAE (Differencial Agebraic Equation):

$$\begin{aligned} \dot{V}_C &= \tfrac{I_L}{C} - \tfrac{V_C}{RC} \\ \dot{I}_L &= \tfrac{V_D - V_C}{L} \\ V_D &= R_D\left(\tfrac{V_{CC} - V_D}{R_{LL}} - I_L\right) \\ v_{out} &= V_C \end{aligned} \tag{6.1}$$

donde

$$R_{LL} = \begin{cases} R_{LL-on} & \text{si } Sw \text{ está cerrada} \\ R_{LL-off} & \text{si } Sw \text{ está abierta} \end{cases} \quad (6.2)$$

$$R_D = \begin{cases} R_{D-off} & \text{si } V_D > 0 (I_D \cdot R_D > 0) \\ R_{D-on} & \text{si } V_D \leq 0 (I_D \cdot R_D \leq 0) \end{cases} \quad (6.3)$$

Este nuevo sistema, si bien no tiene cambios de estructura (sólo cambian parámetros al haber connmutaciones) resulta muy stiff debido a la presencia de resistencias muy grandes o muy chicas según el caso.

El esquema en PowerDEVS correspondiente al modelo de la fuente Buck representado por las ecuaciones (6.1), (6.2) y (6.3) puede verse en la figura 6.2. Notar que en el mismo se utilizó una fuente de onda cuadrada de 10 kHz para comandar la llave (Switch1). Además puede verse que la tensión del diodo se calculó a través de la función implícita *ImplicitFunction* que resuelve localmente la ecuación implícita del diodo mostrada en el modelo y que el diodo se implementó a través del bloque *Switch2* que selecciona entre los valores de alta y baja impedancia según sea el signo de V_D.

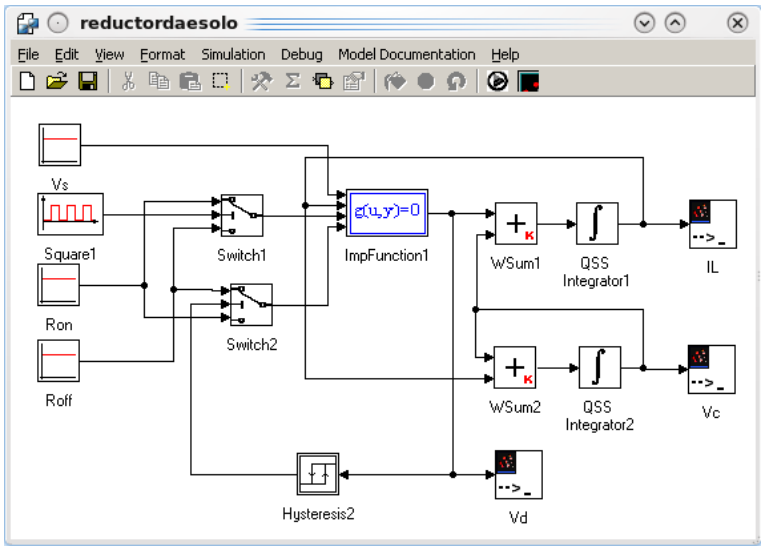

Figura 6.2: Diagrama PowerDEVS del modelo del Convertidor Buck (6.1)

Notar que en los métodos de estados cuantificados, las DAEs se puede tratar muy fácilmente agregando un bloque que resuelve localmente una función implícita [25].

Si se simula el modelo (6.1) con los parámetros de la tabla 6.1 conmutando la llave con una frecuencia de $10kHz$ desde las condiciones iniciales $V_C(0) = 0$ y $I_C(0) = 0$ con quántum $\Delta Q_I = 1 \cdot 10^{-1}$ y $\Delta Q_V = \Delta Q_I = 1 \cdot 10^{-1}$ hasta $t_f =$ 0,01 usando PowerDEVS con LIQSS3 se obtienen los resultados de simulación mostrados en las figuras 6.3.

La simulación hasta $t_f = 0{,}01$ utilizando PowerDEVS con LIQSS3 y quántum $\Delta Q_I = \Delta Q_V = 1 \cdot 10^{-3}$ realizó 2998 pasos en total (1597 en I_L y 1401 en V_C) y requirió sólo 0.129 segundos de CPU.

Parámetro	**Valor**
R	1Ω
C	$1 \cdot 10^{-4}$ F
L	$1 \cdot 10^{-4}$ Hr
Ron Diodo Rof Diodo	$1 \cdot 10^{-6}$ $1 \cdot 10^{6}$
Ron Sw Roff Sw	$1 \cdot 10^{-6}$ $1 \cdot 10^{6}$
Frecuencia de conmutación de la llave	10 kHz
Vcc	24V

Tabla 6.1: Valores de los parámetros de la fuente Buck

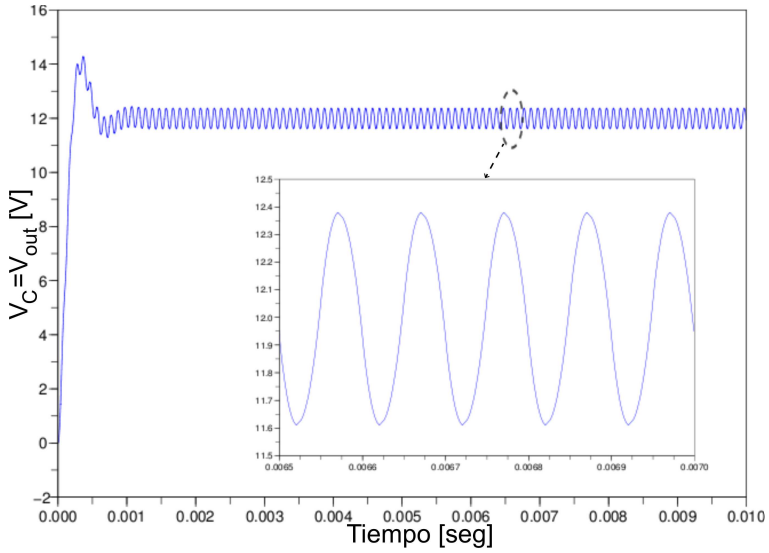

Figura 6.3: Tensión en el Capacitor simulada con PowerDEVS (LIQSS3)

Este mismo modelo fue construido en Dymola utilizando como método de integración DASSL, DAE multi-step solver (dassl/dasslrt of Petzold modified by Dynasim), ya que fue con el que se obtuvieron los mejores resultados. Los resultados gráficos obtenidos en este entorno no diferían de los obtenidos con PowerDEVS pero el tiempo de CPU requerido para terminar la simulación resultó ser de 1.09 segundos (8.5 veces mayor al requerido por PowerDEVS) y el número de evaluación de funciones $\mathbf{f}(\mathbf{x}, t)$ en Dymola fue 12915 (64757 evaluaciones de $f_i(\mathbf{x}, t)$).

Si se compara el número de evaluación de funciones en Dymola con las 14424 de PowerDEVS, se ve que no tiene la misma relación que la encontrada entre los tiempos de simulación. Esta diferencia se debe principalmente a que además de las evaluaciones de funciones $f_i(\mathbf{x}, t)$), en los algoritmos de tiempo discreto implementados en Dymola se deben realizar estimas de Jacobiano, inversiones de matrices y aplicar algoritmos de detección de discontinuidades. En la tabla

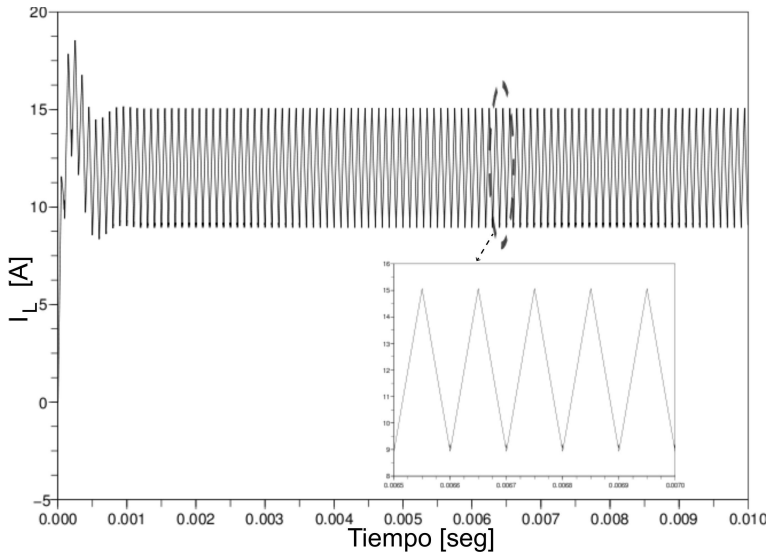

Figura 6.4: Corriente en la bobina simulada con PowerDEVS (LIQSS3)

6.2 puede verse el reporte que dio Dymola al finalizar la simulación. En el mismo se ve un mayor detalle de las tareas que debe realizar el método de integración para poder terminar la simulación.

CPU-time for integration	1.09 seconds
Number of (successful) steps	4805
Number of F-evaluations	12915
Number of H-evaluations	5805
Number of Jacobian-evaluations	2605
Number of (model) time events	200

Tabla 6.2: Reporte de simulación en Dymola del modelo 6.1

Se construyó también este modelo en Matlab/Simulink y se intentó simularlo con las mismas condiciones iniciales que en los dos casos anteriores. Con ninguno de los métodos de integración implementados en Matlab fue posible simular este modelo (ninguno fue capaz de resolver el lazo algebraico que involucra al *Switch* que selecciona el valor de resistencia del diodo (según V_D), y la función implícita que calcula V_D. La única forma de lograr que la simulación se completara en Simulink fue intercalar dinámica (un filtro de primer orden con ganancia estática unitaria) entre la entrada de selección del switch y la señal V_D. Aún siendo que los resultados así obtenidos coincidían con los de los otros simuladores, carecen de validez a la hora de comparar ya que el modelo resultante es otro y además, el tiempo CPU requerido por la simulación resultó ser mucho mayor que el de Dymola.

En la tabla 6.3 se muestran resultados comparativos de cantidad de pasos y tiempo de simulación utilizando distintos ordenes de LIQSS y variando el quántum. Como puede verse, al disminuir 100 veces el quántum, el número de

pasos aumenta aproximadamente 10 veces en LIQSS2, poniéndose de manifiesto que es un método de segundo orden) y en LIQSS3 aumenta aproximadamente $\sqrt[3]{100} = 4{,}64$ veces, acorde con un método de tercer orden.

Método de integración	**Nº de Pasos**	**Evaluaciones de función f_i**	tiempo de CPU
LIQSS2 $1 \cdot 10^{-2}$	3606	14424	0.101
LIQSS2 $1 \cdot 10^{-3}$	10612	42448	0.177
LIQSS2 $1 \cdot 10^{-4}$	33023	132093	0.366
LIQSS3 $1 \cdot 10^{-2}$	1454	8724	0.105
LIQSS3 $1 \cdot 10^{-3}$	2998	17988	0.129
LIQSS3 $1 \cdot 10^{-4}$	5803	34818	0.209
Dymola DASSL DAE multi-step solver	4805	64575	1.09

Tabla 6.3: Comparación de resultados de simulación del modelo 6.1

Además de los datos mostrados en la tabla 6.3, se verificó que en todas las condiciones en que se simuló el sistema, se realizaron 200 conmutaciones de la llave y 201 conmutaciones del diodo.

6.2. Control de velocidad de un Motor

Este ejemplo consiste en un motor de corriente continua (MCC) alimentado por un convertidor Buck cuya tensión de salida está controlada de manera tal que la velocidad de salida del motor ω siga a una señal de referencia ω_{ref}. La figura 6.5 representa un esquema del sistema completo.

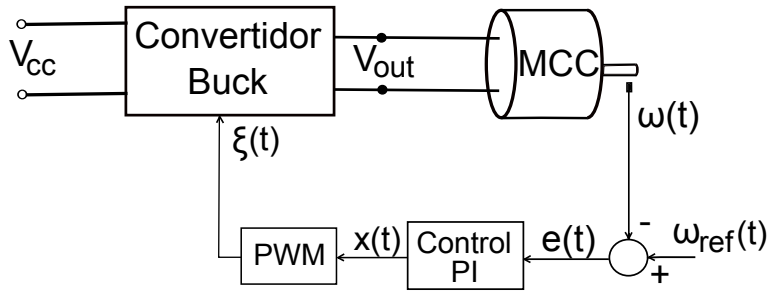

Figura 6.5: Esquema de control de velocidad del MCC con un convertidor Buck

En la misma, puede verse que el control implementado consiste en un control PI del error de velocidad del motor ($e(t)$). Luego, la señal que controla la apertura o cierre de la llave del convertidor se obtiene a partir de un modulador de ancho de pulso (PWM) que genera una onda cuadrada de frecuencia 10 kHz donde el estado alto o bajo de la misma se obtiene comparando la señal $x(t)$ con una onda triangular de frecuencia 10kHz. Si el valor de $x(t)$ es mayor que el de la triangular, $\xi = 1$ en caso contrario $\xi = 0$. En la figura 6.6 se muestra el comportamiento descripto del bloque PWM. La misma fue generada simu-

lando en PowerDEVS el funcionamiento de la misma, permitiendo así mostrar gráficamente la exactitud del mismo en la detección de discontinuidades.

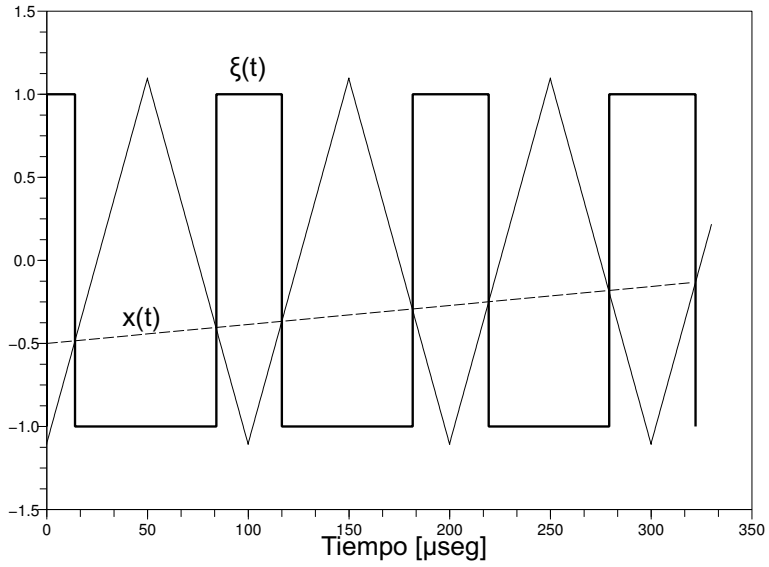

Figura 6.6: Comportamiento del bloque PWM del esquema 6.5

Teniendo en cuenta que la salida del convertidor es igual a la tensión en el capacitor ($V_{out} = V_C$), las ecuaciones que describen el comportamiento del sistema resultan:

$$\dot{I}_a = \frac{1}{L_a}\left(V_C - R_a \cdot I_a - K_m \cdot \omega\right) \tag{6.4}$$

$$\dot{\omega} = \frac{1}{J}\left(I_a \cdot K_m - b \cdot \omega + T_c\right) \tag{6.5}$$

$$\dot{V}_C = \frac{I_L}{C} - \frac{V_C}{RC} - \frac{I_a}{C} \tag{6.6}$$

$$\dot{I}_L = \frac{V_D - V_C}{C} \tag{6.7}$$

$$V_D = R_D\left(\frac{V_{CC} - V_D}{R_{LL}} - I_L\right) \tag{6.8}$$

$$\dot{e}(t) = \omega_{ref}(t) - \omega(t) \tag{6.9}$$

$$x(t) = k_p \cdot e(t) + k_i \int_0^t e(t)dt \tag{6.10}$$

donde

$$R_{LL} = \begin{cases} R_{LL-on} & \text{si } \xi = 1 \ (Sw \text{ está cerrada}) \\ R_{LL-off} & \text{si } \xi = -1 \ (Sw \text{ está abierta}) \end{cases} \tag{6.11}$$

$$R_D = \begin{cases} R_{D-off} & \text{si } V_D > 0 (I_D \cdot R_D > 0) \\ R_{D-on} & \text{si } V_D \leq 0 (I_D \cdot R_D \leq 0) \end{cases} \tag{6.12}$$

Puede verse que las ecuaciones 6.4 y 6.5 corresponden al modelo del motor y las ecuaciones 6.6, 6.7 y 6.8 representan la dinámica del conversor buck teniendo como carga al motor.

Para simular el sistema se utilizaron para el conversor Buck los parámetros mostrados en la tabla 6.1 del primer ejemplo de este capítulo, los parámetros utilizados en el controlador y los correspondientes al motor pueden verse en la tabla 6.4

Parámetro	**Valor**
Resistencia de armadura (R_a)	1,73 Ω
Inductancia de armadura (L_a)	2,54 mH
Momento de Inercia del rotor (J)	$1{,}62 \cdot 10^{-5}$ Nm/s^2
Constante de fuerza electromotriz (K_m)	0,0551
coeficiente de amortiguamiento del sistema mecánico (b)	$1{,}12 \cdot 10^{-5}$
k_p	0,2
k_i	0,2

Tabla 6.4: Tabla de valores de los parámetros del MCC y controlador del ejemplo 2

Además, se utilizó como referencia de velocidad una señal que comienza en $t = 0$ con valor $\omega_{ref}(0) = 0$, evoluciona con pendiente constante 54,1 hasta que en $t = 2$ alcanza el valor $\omega_{ref} = 108{,}2$ y, a partir de ese instante, se mantiene con ese valor.

Para simular este sistema en PowerDEVS, se utilizó LIQSS2 con valores de quántum $\Delta Q_{Ia} = 1 \cdot 10^{-3}$, $\Delta Q_{\omega} = 1 \cdot 10^{-3}$, $\Delta Q_{Vc-min} = 1 \cdot 10^{-6}$, $\Delta Q_{I_{LL}-min} = 1 \cdot 10^{-6}$ y $\Delta Q_{PI-min} = 1 \cdot 10^{-6}$ y en todos los integradores se utilizó $\Delta Q_{rel} = 1e-3$ teniendo además todos los integradores con condiciones iniciales nulas.

Utilizando todos los valores de parámetros y quántum especificados, se simuló el sistema en PowerDEVS hasta $t_f = 3seg$. La simulación requirió en total 10.251 segundos de CPU en los cuales realizaron 586790 pasos en total (233304 en V_C, 229092 en I_L, 24180 en ω, 99875 en I_a y 339 en el integrador del controlador) y se detectaron 60001 discontinuidades en la llave y 68819 en el diodo.

En las figuras 6.7, 6.8 y 6.9 pueden verse la velocidad del motor, la tensión de salida del convertidor y la señal a la salida respectivamente obtenidas al simular el modelo en PowerDEVS utilizando los parámetros y condiciones antes mencionadas.

Se simuló el mismo modelo en Dymola utilizando el método esdirk23a con precisión relativa de $1 \cdot 10^{-4}$, ya que fue el que más eficientemente pudo simular el modelo . Los resultados arrojados por Dymola no difieren de los obtenidos con PowerDEVS, sin embargo, el tiempo de CPU requerido por Dymola para simular 3 segundos el modelo resultó 271.166 segundos (aproximadamente 25 veces más que los 10.251 segundos requeridos por PowerDEVS). La simulación involucró 181649 pasos aceptados y 1450 pasos rechazados; Por otro lado, se detectaron 60121 eventos de estado y 60000 eventos temporales. Dado que el número de eventos detectados usando Dymola y PowerDEVS son similares mientras que el número de pasos de Dymola es menor, se podía pensar que tendría que haber sido menor el costo computacional al simular en Dymola. Sin

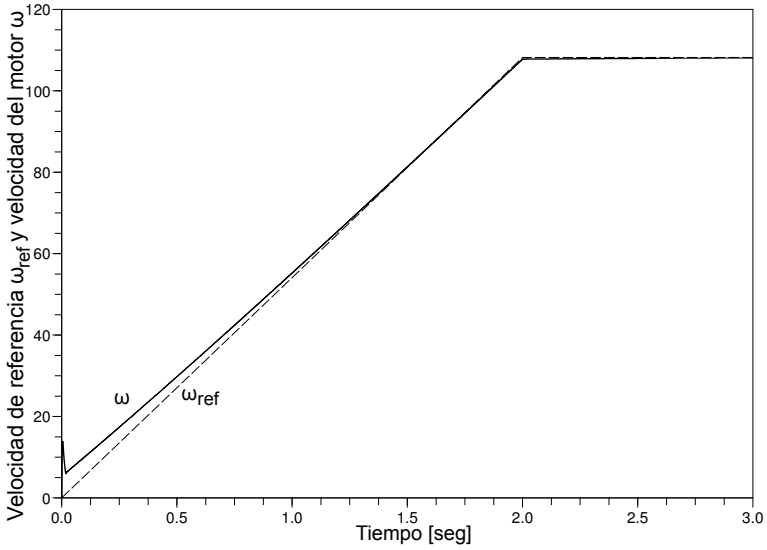

Figura 6.7: Velocidad del motor y señal de referencia del ejemplo 2

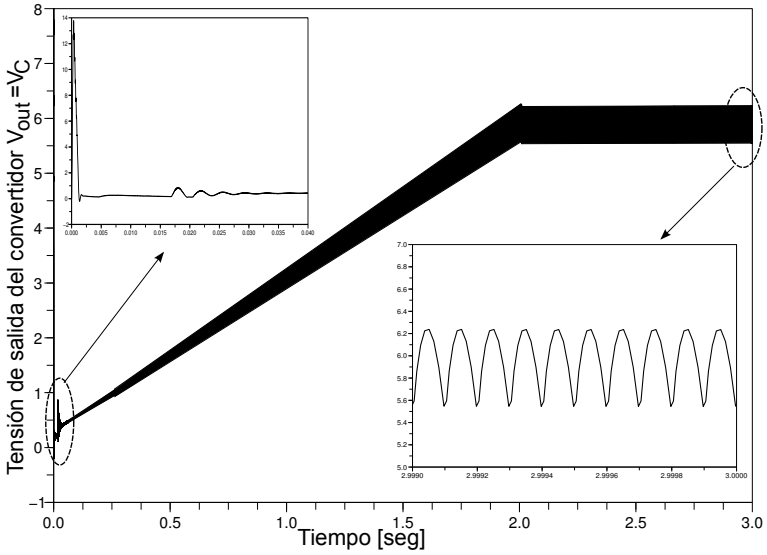

Figura 6.8: Tensión de salida del convertidor buck del ejemplo 2

embargo, cuando se habla de un paso, en PowerDEVS no se evalúan todas las funciones del sistema, sólo aquellas que dependen explícitamente de la variable cuantificada que cambió de valor mientras que en cada paso en Dymola se evalúan todas las funciones del sistema una cantidad de veces que depende del orden del método y de la cantidad de iteraciones requeridas.

A cálculo realizado se puede ver que el número de evaluaciones de las funciones $\dot{x}_i = f_i(\mathbf{q}, \mathbf{u})$ en PowerDEVS resulta $(233304 \cdot 3 + 229092 \cdot 2 + 24180 \cdot 3 + 99875 \cdot 3 + 60001 + 68819 = 1659081$. El número de evaluaciones de funciones $\mathbf{f}(\mathbf{x}, t)$ en Dymola es 1179482, por lo tanto, el número de evaluaciones de $\dot{x}_i = f_i(\mathbf{x}, t)$ re-

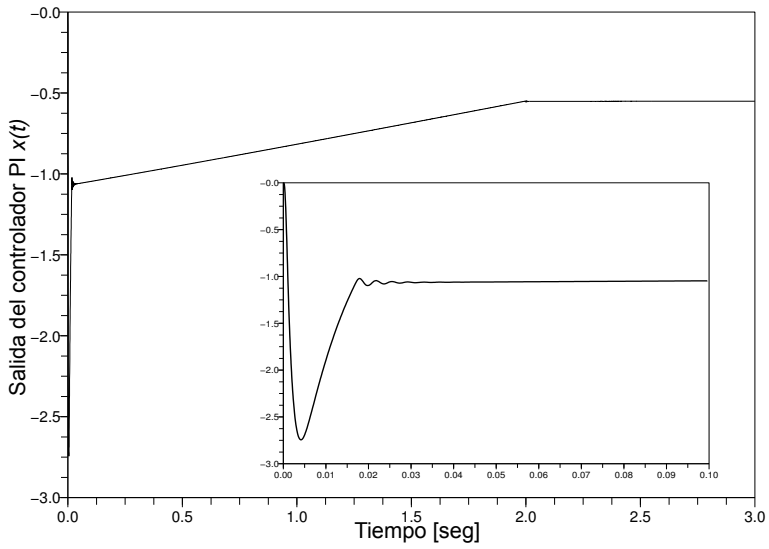

Figura 6.9: Señal $x(t)$ de salida del controlador

sulta igual a la cantidad de evaluaciones de $\mathbf{f}(\mathbf{x}, t)$ multiplicado por el orden del sistema, siendo en este caso $1179482 \cdot 5 = 5897410$. Observando estos números ya se puede comprender por qué PowerDEVS con LIQSS2 logró mejorar los tiempos respecto al método implementado en Dymola. Además, al tiempo requerido por el método en Dymola en evaluar las funciones, se le debe agregar el de estima de Jacobiano e inversiones de matrices de orden cinco en cada paso más el costo del método iterativo implementado para detectar las discontinuidades.

En la tabla 6.5 puede verse un resumen de los tiempos de simulación y cantidad de evaluaciones de función obtenidas al simular con los distintos métodos de integración y plataformas mencionadas.

Método de integración	**Nº de Pasos**	**Evaluaciones de función** f_i	tiempo de CPU
LIQSS2 ($\Delta Q_{ia} = \Delta Q_w = 1 \cdot 10^{-3}$, $\Delta Q_{Vc} = \Delta Q_{I_{LL}} = \Delta Q_{PI} = 1 \cdot 10^{-6}$)	586790	1659081	10.251
Dymola esdirk23a	181649	5897410	271.16

Tabla 6.5: Comparación de performance de simulación del modelo (6.13) usando distintos métodos de integración

6.3. Cadena de inversores lógicos

Un inversor lógico es un circuito eléctrico que transforma una señal lógica de entrada en su versión negada. Una cadena de inversores lógicos consiste en la concatenación de muchos inversores lógicos de manera tal que la salida de un

inversor es la entrada del que lo sigue.

Si la cadena de inversores tiene un número par de inversores, la misma actuará como un retardo para la señal de entrada (obteniéndose a la vez una versión más "suave"de la misma).

Una descripción detallada del modelo matemático de la cadena de inversores puede encontrarse en [1]. El modelo correspondiente a una cadena de m inversores puede ser representado por las siguientes ecuaciones:

$$\begin{cases} \dot{\omega}_1(t) & = U_{op} - \omega_1(t) - \Upsilon g\left(u_{in}(t), \omega_1(t)\right) \\ \dot{\omega}_j(t) & = U_{op} - \omega_j(t) - \Upsilon g\left(\omega_{j-1}(t), \omega_j(t)\right) \quad j = 2, 3, .., m \end{cases} \tag{6.13}$$

donde

$$g(u, v) = \left(max(u - U_{thres}, 0)\right)^2 - \left(max\left(u - v - U_{thres}, 0\right)\right)^2 \tag{6.14}$$

El parámetro Υ define el grado de rigidez del sistema.

El modelo fue simulado en PowerDEVS para una cadena de 500 inversores ($m = 500$) con los parámetros $\Upsilon = 100$ (lo cual redunda en un sistema muy rígido), $U_{thres} = 1$ y $U_{op} = 5$ tomando como condiciones iniciales $\omega_j = 6{,}247 \cdot 10^{-3}$ si j es par y $\omega_j = 5$ si j es impar. Como señal de entrada se tomó:

$$u_{in}(t) = \begin{cases} t - 5 & \text{si } 5 \leq t \leq 10 \\ 5 & \text{si } 10 < t \leq 15 \\ \frac{5}{2}(17 - t) & \text{si } 15 < t \leq 17 \\ 0 & \text{c.o.c} \end{cases} \tag{6.15}$$

En la parte superior de la figura 6.10 puede verse el diagrama en PowerDEVS de un integrador lógico. Cabe aclarar que los saturadores tienen limite superior infinito e inferior cero de manera que la salida de los mismos es el máximo entre su señal de entrada y cero, permitiendo así implementar la ecuación (6.14). En la parte inferior de la misma puede observarse el modelo completo de 500 inversores. En ésta parte de la gráfica, cada bloque *InvA_B* es un modelo acoplado que contiene 100 inversores lógicos como el mostrado en la parte superior de la figura.

En las figuras 6.11 y 6.12 puede verse la evolución de algunas de las variables w_i (diferenciándose las componentes pares de las impares) obtenidas simulando el sistema hasta $tf = 130$seg utilizando LIQSS2 con quántum $\Delta Q_i = 0{,}001$ y $\Delta Q_{i-rel} = 0{,}001$ en todos los integradores ($i = 1, .., 500$).

En primer lugar, se debe mencionar que aunque el tiempo final de simulación se fijó en 500 segundos, la simulación terminó a los 128.47 segundos ya que apartir de ese momento no cambió ninguna variable del sistema. La simulación requirió en total 25105 pasos y 6.199 segundos de CPU. Cada uno de los integradores que calculan ω_j con j impar realizan entre 540 y 567 pasos, mientras que los demás integradores realizan entre 420 y 507 pasos.

Luego, se volvió a simular el modelo del sistema hasta $tf = 500$ pero utilizando en este caso LIQSS3 con quántum $\Delta Q_i = 1 \cdot 10^{-5}$ y $\Delta Q_{i-rel} = 1 \cdot 10^{-5}$. En este caso la simulación terminó en t=139.771 segundos requiriendo 36650 pasos y 13.41 segundos de CPU. Notar que el tiempo de CPU requerido por LIQSS3 es sólo 1.8 veces mayor que el requerido por LIQSS2 mientras que el quántum utilizado es 100 veces menor.

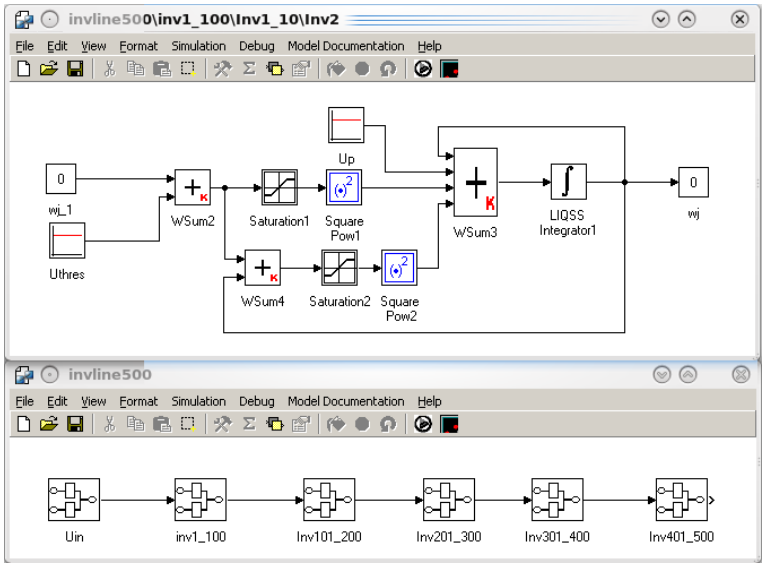

Figura 6.10: Diagrama PowerDEVS de la cadena de inversores lógicos

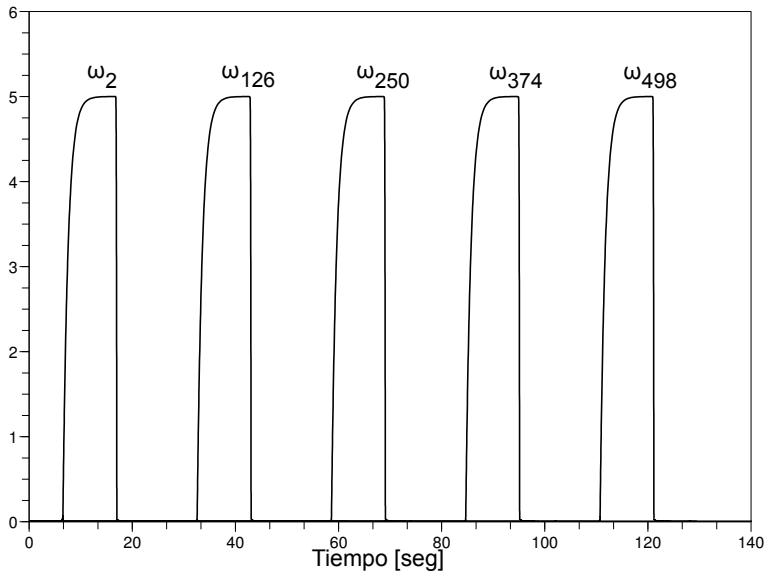

Figura 6.11: Algunas componentes ω_j (con j par) de la solución en PowerDEVS del ejemplo 3

Si se vuelve a simular el modelo con LIQSS3 pero con quántum 10 veces menor ($\Delta Q_i = 1 \cdot 10^{-6}$ y $\Delta Q_{i-rel} = 1 \cdot 10^{-6}$) la simulación requiere 55030 pasos y 18.74 segundos de CPU lo cual es acorde con un método de tercer orden donde al variar 10 veces la relación de quántum, la cantidad de pasos debería variar aproximadamente $\sqrt[3]{10} = 2{,}15$ veces. En este caso la relación de pasos no es tan próxima a 2,15 debido a las discontinuidades.

Se simuló este modelo en Matlab/Simulink en modo acelerado utilizando

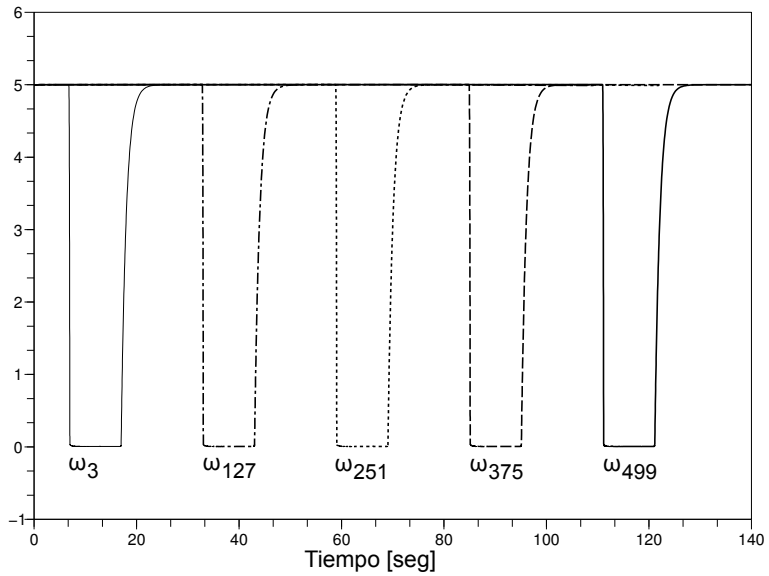

Figura 6.12: Algunas componentes ω_j (con j impar) de la solución en PowerDEVS del ejemplo 3

el método de integración ode15s con tolerancia relativa de error de $1 \cdot 10^{-2}$. La simulación realizó 13220 pasos y requirió 651.37 segundos de CPU (21 veces más de lo requerido por PowerDEVS con LIQSS3 usando el quántum más chico). Teniendo en cuenta que en este caso también se utilizó un método de integración adecuado para sistemas Stiff, el tiempo requerido por la simulación parece excesivamente grande respecto al obtenido con los métodos LIQSS. En primer lugar, debe recordarse que los métodos de tiempo discreto evalúan todas las componentes de la función $\mathbf{f}$ en cada paso (en este caso, tendremos al menos 500 evaluaciones por paso).

Teniendo también en cuenta que todos los métodos de integración de tiempo discreto para sistemas stiff son implícitos, entonces deben hacer al menos una inversión de una matriz de dimensión $n \times n$ (donde n es el orden del modelo). Además, cuando los modelos tienen discontinuidades, en la búsqueda del instante de ocurrencia de las mismas también se debe realizar al menos una inversión de matrices. Entonces, cuando el orden del modelo es grande, el costo computacional de invertir estas matrices es muy alto y consecuentemente el costo computacional de cada paso dado por el método es también muy alto.

Para poder ver cuantitativamente como incide del orden del sistema en el tiempo de simulación, se simuló la cadena de inversores variando la cantidad de inversores lógicos y tomando el tiempo de CPU que requiere en cada uno de los casos simular los modelos hasta $tf = 130$. En la figura 6.13 puede verse como varía el tiempo requerido por la simulación en función de la longitud de la cadena de inversores lógicos (equivalente al orden del modelo). Como puede verse, en el caso de los método LIQSS, independientemente del orden del método, la relación *tiempo de CPU-orden del sistema* es lineal mientras que en el caso del método ode15s ésta relación es polinomial. Esta no linealidad se debe en gran medida al costo computacional de orden cubico de invertir una matriz de $n \times n$ en cada

paso de integración.

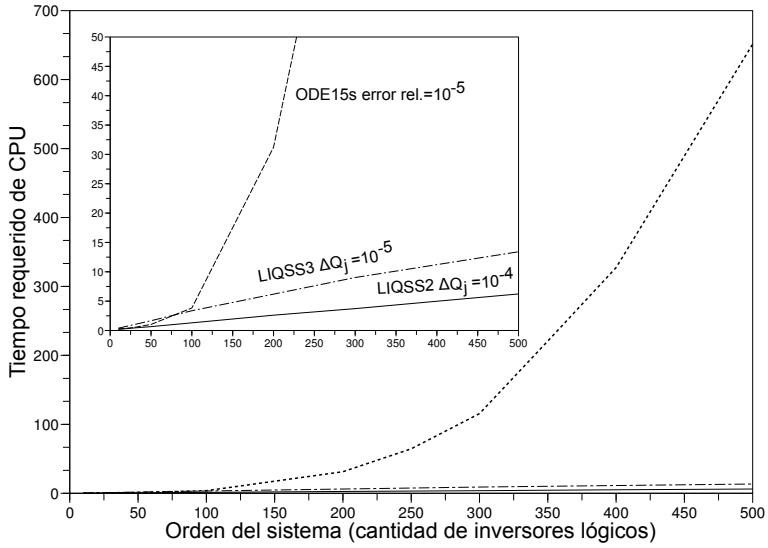

Figura 6.13: Tiempo de CPU en función del orden del sistema simulando una cadena de n inversores lógicos

Observando la figura 6.13 podemos concluir que utilizar métodos implícitos de tiempo discreto en modelos stiff de dimensión 'grande' tiene la ventaja de no tener que utilizar pasos de integración excesivamente pequeños (como ocuriría si se usaran métodos explícitos) pero el costo computacional puede ser excesivamente alto debido a las inversiones de matrices requerido por los métodos implícitos. A su vez, vemos que el problema de dimensión del modelo no es una dificultad para los métodos LIQSS ya que el tiempo requerido por la simulación crece linealmente con el orden.

Este mismo modelo fue simulado en [41] utilizando métodos de integración multirate para sistemas stiff. El método multirate utilizado *Multirate II* utiliza una estrategia multirate diseñada para métodos mono paso. Se uso Rosenbrock ROS2 [14] como método base. Además, se usó para el ejemplo el valor exacto del Jacobiano (esto ahorra el tiempo requerido para la estima del mismo). La estrategia utilizada particiona los componentes entre lentos y rápidos utilizando para esto una función de monitoreo adecuada para este problema particular. Luego, todos los componentes rápidos se resuelven con un paso chico y los lentos con uno grande.

En la tabla 6.6 pueden verse los valores de tiempos de CPU requeridos al utilizar distintos métodos de integración y con distintas configuraciones de parámetros de simulación utilizados para simular el modelo 130 segundos.

Al analizar la tabla 6.6 se debe tener en cuenta que cada paso en los métodos LIQSS implican $2 \times nm$ (nm es el orden del método) evaluaciones de función $\dot{\omega}_i = f_i(\mathbf{q}(t), \mathbf{u}(t))$ mientras que en ode15s en cada paso se evalúa más de una vez toda la función $\dot{\omega} = \mathbf{f}(\omega, t)$.

Si se comparan los tiempos de CPU de la tabla 6.6 teniendo simultáneamente en cuenta la precisión requerida, puede verse que salvo con respecto a los métodos multirate, los métodos LIQSS logran una gran mejora en los tiempos

Método de integración	**Nº de Pasos**	**Evaluaciones de función**	tiempo de CPU
LIQSS2 ($\Delta Q_i = 0{,}001$ y $\Delta Q_{i-rel} = 0{,}001$)	25105	100420	6.199
LIQSS3 ($\Delta Q_i = 1 \cdot 10^{-5}$ y $\Delta Q_{i-rel} = 1 \cdot 10^{-5}$)	36650	219900	13.41
LIQSS3 ($\Delta Q_i = 1 \cdot 10^{-6}$ y $\Delta Q_{i-rel} = 1 \cdot 10^{-6}$)	55030	330180	18.74
Matlab/Simulink Ode15s (tolerancia error relativa $1 \cdot 10^{-2}$)	13220	> 33050000	651.37
Multirate II (Tolerancia de error $1 \cdot 10^{-4}$)	-	4795878	6.36
Multirate II (Tolerancia de error $1 \cdot 10^{-5}$)	-	17358472	21.65

Tabla 6.6: Comparación de performance de simulación del modelo (6.13) usando distintos métodos de integración

de simulación.

Cuando se los compara con los métodos multirate, los tiempos de CPU requeridos son del mismo orden. Sin embargo, no debe perderse de vista que para utilizar los métodos multirate se debe tener cierto conocimiento de las ODEs del modelo a simular y que además, como se mencionó anteriormente, en este caso se le dio al método el Jacobiano exacto mientras que al construir el modelo en PowerDEVS, simplemente se construyó el diagrama en bloques y no se le dio ninguna información a los métodos ni se los adaptó para este modelo en particular.

Además, puede verse que el número de evaluaciones de función realizado por los métodos LIQSS es mucho menor al realizado por los demás métodos y aun así, no se tuvo gran ventaja respecto a los métodos multirate en lo que respecta a tiempos de simulación. El motivo de esto es que PowerDEVS es muy ineficiente para manejar modelos muy grandes. Cada evaluación de función implica en PowerDEVS un gran tráfico de eventos entre los distintos bloques y esto agrega un costo muy grande. Una implementación de los métodos por fuera de PowerDEVS (o incluso una optimización del motor de PowerDEVS para manejar modelos grandes) podría reducir drásticamente estos tiempos.

6.4. Conclusiones

Como pudo verse a través de los ejemplos de este capítulo, al utilizar los métodos LIQSS desarrollados en el capítulo 5, los tiempos de simulación en sistemas stiff pueden ser reducidos significativamente cuando la dinámica rápida está asociada a una variable x_j cuya derivada depende explícitamente de x_j.

En el primer ejemplo se ve claramente que la mejora en eficiencia es realmente significativa en la simulación de sistemas stiff en los que hay muchas discontinuidades, ya que los algoritmos tradicionales pierden mucho tiempo en la detección del instante de ocurrencia de la discontinuidad y el tratamiento simultáneo de la rigidez (resolución de la ecuación implícita del método).

En segundo ejemplo se suma a los problemas anteriores la mayor complejidad del sistema y la presencia de más eventos de estado. En este caso, los métodos de tiempo discreto (aún los más eficientes) resultan extremadamente lentos. Aquí la utilización de los LIQSS brinda resultados 25 veces más rápidos que lo mejor que se puede obtener con Dymola (el único software con el cual se pudo simular el sistema).

Por otro lado, como pudo verse en el tercer ejemplo, los métodos LIQSS también tienen grandes ventajas en sistemas de orden alto. Especialmente cuando no todas las variables del sistema evolucionan al mismo tiempo. La ventaja obtenida por los métodos LIQSS sobre los métodos de integración de tiempo discreto clásicos, se debe en parte a que cuando el orden del sistema es grande, empieza a ser realmente significativo el costo del cálculo de Jacobiano, de inversiones de matrices, etc. observándose un crecimiento cúbico del costo computacional en función de la dimensión del problema. Además, los métodos LIQSS sólo evalúan aquellas componentes del sistema en las que hay cambios significativos, lo que reduce drásticamente los tiempos de simulación y conlleva un aumento sólo lineal del costo computacional con la dimensión del problema.

Una alternativa más eficiente que los métodos clásicos de tiempo discreto para simular este tipo de sistemas la constituyen los métodos multirate y en algunos casos métodos de integración mixtos (utilizando métodos explícitos en la dinámica lenta e implícitos en la rápida) cuando son claramente separables las dinámicas. Sin embargo, éstos son métodos de aplicación ad–hoc (no constituyen métodos de propósito general como los LIQSS) y su utilización requiere de mucho trabajo previo para determinar la partición adecuada del sistema.

De todas maneras, y aún con todo el trabajo previo que implica la utilización de los métodos multirate, los métodos LIQSS (al menos en el ejemplo estudiado, tomado de un artículo sobre métodos multirate) realizan mucho menos cálculos y el tiempo de simulación (aún en una implementación ineficiente) es menor.

Capítulo 7

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 y se presentarán algunas ideas para resolverlos. Finalmente, se presentarán las conclusiones generales de la Tesis.

7.1. Problemas abiertos

Al igual que en BQSS, la idea base en LIQSS1 consiste en tratar que los estados x_j se dirijan siempre hacia sus versiones cuantificadas q_j. Si bien esta condición se verifica siempre en BQSS, en LIQSS sólo se asegura su cumplimiento si, una vez que se ha fijado el valor de q_j no ocurre ningún cambio en el signo de la derivada $\dot{x}_j$ hasta tanto se alcance la condición de un nuevo cambio en q_j. Por ejemplo, si en algún momento cambia una variable q_i (con $i \neq j$), y la función f_j depende de q_i, podría ocurrir que x_j deje de ir en el sentido de q_j. Frente a esta situación, puede generarse nuevamente la aparición de oscilaciones de alta frecuencia en la simulación de sistemas stiff, como veremos en el siguiente ejemplo.

Consideremos el sistema

$$\begin{aligned} \dot{x}_1 &= 0{,}00001 \cdot x_2 \\ \dot{x}_2 &= 0{,}01 \cdot x_3 \\ \dot{x}_3 &= -100 \cdot x_1 - 100 \cdot x_2 - 100 \cdot x_3 + 2020 \end{aligned} \tag{7.1}$$

con condiciones iniciales $x_1(0) = 0$, $x_2(0) = 20$ y $x_3(0) = -20$.

Este sistema tiene autovalores $\lambda_1 \approx -1e-5$; $\lambda_2 \approx -0{,}01$; $\lambda_3 \approx -100$, lo que muestra que es muy stiff.

Si se simula el mismo hasta $t_f = 350000$ utilizando el método LIQSS1 con quántum $\Delta Q_i = 1$ con $i = 1, 2, 3$, se tiene el resultado de la figura 7.1.

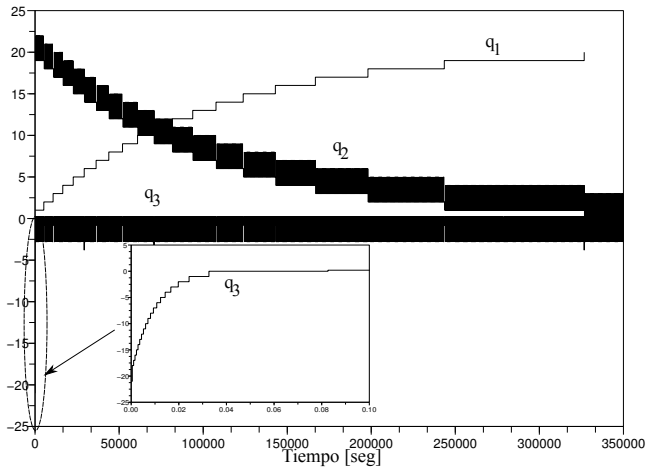

Figura 7.1: Resultado de simulación del sistema (7.1) usando LIQSS1

En la misma puede verse que, a pesar de que la solución es cualitativamente correcta, nuevamente se tienen oscilaciones de alta frecuencia. La simulación requirió en este caso 20 cambios en q_1, 3735 en q_2 y 1858 en q_3. Esto, si bien mejora mucho lo que puede hacer cualquier método explícito, no es un resultado eficiente.

Si se tiene en cuenta que las tres trayectorias tienen una excursión aproximadamente igual a 20, esta simulación debería poderse resolver con aproximadamente 20 pasos en cada variable utilizando el quántum mencionado. El problema se debe justamente a que no siempre se cumple que las variables de estado x_i se dirigen hacia q_i.

Puede verse además que la rigidez del sistema no se debe a términos grandes sobre la diagonal principal de la matriz Jacobiana (que era la condición que mencionamos debía cumplirse para garantizar que LIQSS trate eficientemente el problema stiff).

Actualmente hay programada una versión de LIQSS1 generalizada que garantiza que siempre, al menos en el caso de sistemas lineales, x_j se dirige hacia q_j. Frente a cambios en una variable cuantificada que provocan cambios en el

signo de la derivada de otras, el nuevo algoritmo busca valores correctos de las distintas variables cuantificadas involucradas para que en todas ellas se cumpla la condición mencionada.

Utilizando este nuevo método, al simular nuevamente el sistema (7.1) desde las mismas condiciones de antes y con el mismo quántum pero con un tiempo final de simulación $t_f = 900000$, se obtienen los resultados observados en la figura 7.2.

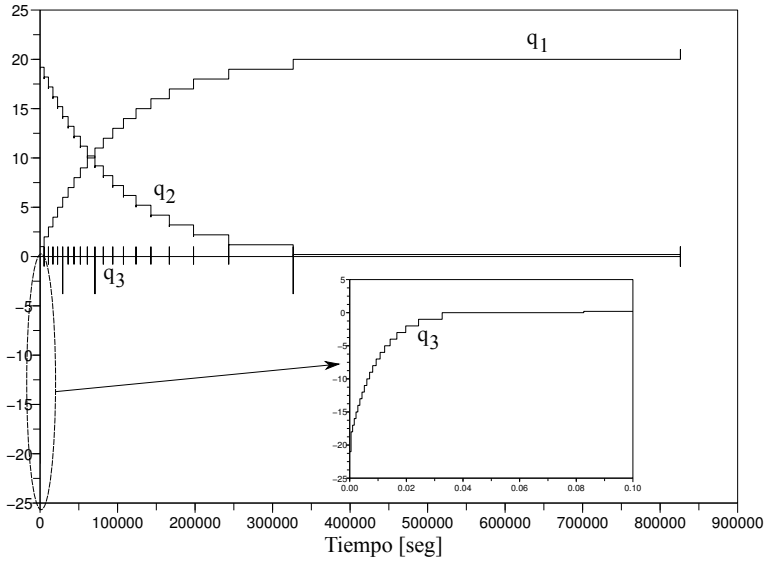

Figura 7.2: Resultado de simulación utilizando la versión generalizada de LIQSS1

La simulación en este caso requirió 22 cambios en q_1, 62 en q_2 y 81 en q_3. Además, a los 826164.32 segundos se alcanzó la situación de equilibrio y no se generaron más eventos.

Veamos a continuación una idea intuitiva de cómo funciona este método

Al igual que en BQSS este método hace que en todo momento x_j se dirija siempre hacia q_j (en el caso de sistemas no lineales esto puede no ser cierto debido al error introducido al linealizar el sistema). Cuando esto no es posible, busca de manera muy similar a LIQSS los valores de q_j que hacen que las derivadas correspondientes sean nulas.

Supongamos que en cierto momento una variable x_j alcanza el valor de q_j. En ese instante, deberá entonces incrementarse o decrementarse q_j en un quántum de acuerdo al valor de $\dot{x}_j$. Sin embargo, con este nuevo valor de q_j pueden cambiar los signos de las derivadas de un conjunto de variables.

El cambio en el signo de cada derivada, implicaría en principio cambiar el valor de la variable cuantificada correspondiente. Y cada cambio de variable cuantificada a su vez, puede eventualmente afectar el signo de otras derivadas.

De esta manera, en el cambio del valor de q_j se pueden ver afectadas un cierto número m de variables cuantificadas en la búsqueda de una situación consistente (es decir, donde todas las variables de estado se dirigen hacia sus versiones cuantificadas).

Es posible además, debido a la influencia mutua entre las distintas variables cuantificadas y las derivadas, que en este proceso no se llegue a una situación consisitente.

En este caso, seleccionamos un subconjunto de r de estas m variables afectadas y resolvemos un sistema de ecuaciones lineales que nos brinda los r valores correspondientes de las q_i que anulan sus r derivadas (utilizando el Jacobiano, o una porción del mismo). Es decir, en el proceso invertimos una matriz de $r \times r$. En el caso no lineal, sólo anularemos la parte lineal de la dinámica correspondiente.

La elección del subconjunto de r componentes se hace buscando cuáles de las m variables afectadas poseen un camino de realimentación que incluya sólo variables afectadas. Esto es, una variable afectada q_i forma parte del subconjunto si f_i depende de q_i o bien si otra componente afectada f_j depende de q_i y f_i depende de q_j, etc.

Una interpretación simple del subconjunto en cuestión puede inferirse a partir del Diagramas de Bloques del sistema: pintamos de un color los m integradores involucrados en el proceso y las m funciones estáticas que calculan sus derivadas. Si partiendo de un integrador pintado existe un camino cerrado que pasa sólo por bloques pintados, entonces la variable correspondiente a este integrador forma parte del subconjunto de r variables.

Si bien este método presenta ventajas en la simulación de algunos sistemas sobre el método LIQSS1, el mismo no debe reemplazarlo siempre. Este tiene una fuerte desventaja sobre su predecesor LIQSS1 al precisar invertir matrices. Esto es necesario para encontrar los valores de q_j que anulan la derivada cuando no se cumple que las variables de estado se dirigen hacia sus versiones cuantificadas ni con $\overline{q}_j$ ni con $\underline{q}_j$. Además, los métodos LIQSS mostraron ser muy apropiados para la simulación de sistemas de electrónica de conmutación.

Este método aún esta en proceso de desarrollo y no ha sido formalizado. Sin embargo, usando las ideas del mismo se podrían obtener métodos LIQSS generales de orden superior.

También se esta trabajando actualmente en dos métodos uno de primer orden y otro de segundo orden que son un caso intermedio entre el método que se presentó recién y los métodos LIQSS presentados en la Tesis. Estos nuevos algoritmos, si bien tampoco podrían resolver adecuadamente modelos generales, están desarrollándose de manera de poder simular eficientemente sistemas stiff cuyos modelos tengan estructura de controlabilidad. En estos casos, las soluciones obtenidas deberían ser las mismas que las de método LIQSS generalizado pero tendrían la ventaja de no requerir inversiones de matrices. Se han planteado dos algoritmos (de orden uno y dos) que implementan estos métodos pero aún no ha sido concluida su programación.

7.2. Trabajos Futuros

La ventaja práctica más grande que se encontró para los métodos desarrollados es en la simulación de circuitos conmutados. Si bien en este trabajo esto se mostró a través de una serie de ejemplos, se deberá realizar un estudio más profundo para poder determinar en forma más precisa en que clase de modelos es realmente conveniente el uso de los mismos y poder dar una justificación adecuada de las razones de estas ventajas.

Además, resta concluir la formalización de los métodos generalizados de LIQSS explicados en la sección anterior. Estos métodos deberán ser implementados adecuadamente y estudiados para determinar su campo de aplicación práctica.

Dado que la implementación de los métodos LIQSS es explícita, el tiempo de cálculo de los algoritmos es considerablemente menor que el de los métodos implícitos de tiempo discreto, lo que les otorga una gran ventaja en el área de simulación en tiempo real.

En los últimos años, uno de los integrantes del grupo modificó el código del software PowerDEVS para permitir su funcionamiento en un sistema operativo de tiempo real (LinuxRTAI) y le agregó nuevas funcionalidades al motor del software que le permiten cumplir con diferentes requisitos asociados a simulaciones que deben cumplir restricciones temporales [3]. Esta plataforma brinda entonces una excelente herramienta para estudiar el desempeño de los métodos LIQSS en tiempo real.

Otro tema que se debe investigar es el control asíncrono de sistemas embebidos. Se realizó un plan de trabajo post doctoral con el objetivo de ahondar en esta problemática. El objetivo principal del mismo es estudiar e implementar sistemas asíncronos de control. Se espera que utilizando técnicas de muestreo disparado por cruces de nivel, se reduzca el costo de comunicación entre el controlador y los dispositivos remotos. En este sentido, los métodos desarrollado pueden llegar a ser un gran aporte para la discretización de controladores diseñados en tiempo continuo.

Finalmente, cabe destacar que actualmente hay un proyecto en curso en el ETH Zurich, encabezado por François Cellier, cuyo objetivo es modificar un compilador de Modelica de manera de poder convertir automáticamente modelos descriptos en este lenguaje en modelos PowerDEVS utilizando los métodos de QSS. Este trabajo, permitirá entre otras cosas, utilizar los métodos de estados cuantificados (los LIQSS en particular) para simular modelos stiff conmutados de manera más eficiente.

7.3. 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 particularmente eficientes en la simulación de circuitos de electrónica conmutada.

Se propusieron cuatro métodos de aproximación llamados BQSS, LIQSS1, LIQSS2 y LIQSS3 (los dos primeros de primer orden y los últimos de segundo y tercer orden respectivamente) que, basados en la idea de cuantificación, mapean sistemas continuos en modelos DEVS. 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.

Además, ninguno de estos métodos requiere iteraciones dado que su implementación es explícita. De este modo se logra una importante reducción del costo computacional de los algoritmos en comparación con los métodos tradicionales de tiempo discreto implícitos.

Desde el punto de vista práctico, una de las ventajas reside en la forma en que se explota la ralitud debido a que cada integrador funciona independientemente. A su vez, en la integración de DAEs stiff no se debe iterar sobre las ecuaciones implícitas en cada paso, evitándose de esta manera una gran cantidad de cálculos, convirtiendo a estos métodos en una buena opción para la simulación de sistemas DAEs stiff ralos.

Como podía esperarse de una aproximación por eventos discretos, la mayor reducción de costo computacional se observó en la simulación de sistemas stiff discontinuos. La eficiencia en el manejo de discontinuidades constituye una de las fuertes ventajas de estos métodos con respecto a los algoritmos clásicos.

De este modo, los métodos presentados constituyen una nueva herramientas para la simulación de modelos stiff discontinuos, particularmente modelos de circuitos de electrónica conmutada.

Si bien existen muchas soluciones alternativas para la simulación de esta clase de modelos, la simulación de los mismos sigue siendo un problema para los métodos de integración. El aporte de estos nuevos métodos en este área, si bien no pretende poder dar respuesta a todo los problemas, demostró que si puede solucionar al menos una parte de ellos simulando parte de estos modelos de manera eficiente.

Bibliografía

[1] A. Bartel. Generalised multirate. two row-type versions for circuit simulation. *Natlab Report No. 2000/84, Philips Electronics*, 2000.

[2] T. Beltrame. Design and Development of a Dymola/Modelica Library for Discrete Event-oriented Systems Using DEVS Methodology. Master's thesis, ETH Zurich, Zurich, Switzerland, 2006.

[3] F. Bergero and E. Kofman. Powerdevs. a tool for hybrid system modeling and real time simulation. *Simulation*, 2010. In Press.

[4] F. Bergero, E. Kofman, C. Basabilbaso, and J. Zúccolo. Desarrollo de un simulador de sistemas híbridos en tiempo real. In *Proceedings of AADECA 2008*, Buenos Aires, Argentina, 2008.

[5] M. Bortolotto, E Kofman, and Migoni G. Metodos de integracion por cuantificacion en sistemas hamiltonianos. In *Proceedings of AADECA 2008*, Rio Gallegos, Argentina, 2008.

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

[7] F. Cellier, E Kofman, G Migoni, and M. Bortolotto. Quantized state system simulation. In *Proceedings of SummerSim 08 (2008 Summer Simulation Multiconference)*, Edinburgh, Scotland, 2008.

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

[9] M. D'Abreu and G. Wainer. M/CD++: Modeling Continuous Systems Using Modelica and DEVS. In *Proc. MASCOTS 2005*, pages 229–238, Atlanta, GA, 2005.

[10] W. H. Enright and J. D. Pryce. Two (fortran) packages for assessing initial value methods. *(ACM) Transactions on Mathematical Software*, 13(1):1–27, 1987.

[11] F. Esquembre. Easy Java Simulations: A software tool to create scientific simulations in Java. *Computer Physics Communications*, 156(1):199–204, 2004.

[12] E. Hairer, S. Norsett, and G. Wanner. Solving ordinary dfferential equations i. nonstiff problems. *Springer, 2nd edición*, 1993.

[13] E. Hairer and G Wanner. Solving ordinary differential equations ii. stiff and differential-algebraic problems. *Springer - Verlag*, 1996.

[14] W. Hundsdorfer and J. G. Verwer. Numerical solution of time-dependent advection- diffusion-reaction equations. *Springer Series in Comput. Math. 33*, 4(3), 2003.

[15] P Kaps, S.W.H. Poon, and T.D Bui. Rosenbrock methods for stiff odes: A comparison of richardson extrapolation and embedding technique. *Computing*, (34):17–40, 1985.

[16] P Kaps and G. Wanner. A study of rosenbrock-type methods of high order. *Numerische Mathematik*, (38):279–298, 1981.

[17] E. Kofman. *Simulación y Control de Sistemas Continuos por Eventos Discretos*. PhD thesis, Facultad de Ciencias Exactas, Ingeniería y Agrimensura. Universidad Nacional de Rosario., 2003.

[18] E. Kofman. Discrete event simulation of hybrid systems. *SIAM Journal on Scientific Computing*, 25(5):1771–1797, 2004.

[19] E Kofman. A third order discrete event method for continuous system simulation. part i: Theory. *Proceedings of RPIC'05*, 2005.

[20] E Kofman. A third order discrete event method for continuous system simulation. *Latin American Applied Research*, 2006.

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

[22] E. Kofman, H. Haimovich, and M. Seron. A systematic method to obtain ultimate bounds for perturbed systems. *International Journal of Control*, 80(2):167–178, 2007.

[23] E. Kofman, G. Migoni, and F. Cellier. Integración por cuantificación de sistemas stiff. parte i: Teoría. In *Proc. AADECA 2006*, 2006.

[24] E. Kofman and B. Zeigler. DEVS Simulation of Marginally Stable Systems. In *Proceedings of IMACS'05*, Paris, France, 2005.

[25] Ernesto Kofman. Quantization–Based Simulation of Differential Algebraic Equation Systems. Technical Report LSD0204, LSD, UNR, 2002. Aceptado para publicación en *Simulation*.

[26] Ernesto Kofman. A second order approximation for devs simulation of continuous systems. *Simulation*, 78(2):76–89, 2002.

[27] Ernesto Kofman and Sergio Junco. Quantized State Systems. A DEVS Approach for Continuous System Simulation. *Transactions of SCS*, 18(3):123–132, 2001.

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

[29] G. Migoni and E. Kofman. Linearly implicit discrete event methods for stiff odes. part i: Theory. In *Proc. RPIC 2007*, 2007.

[30] G. Migoni and E. Kofman. Linearly implicit discrete event methods for stiff odes. part ii: Implementation. In *Proc. RPIC 2007*, 2007.

[31] G. Migoni and E. Kofman. Linearly implicit discrete event methods for stiff odes. *Latin American Applied Research*, 39(3):245–254, 2009.

[32] G. Migoni, E. Kofman, and F. Cellier. Integración por cuantificación de sistemas stiff. parte ii: Implementación. In *Proc. AADECA 2006*, 2006.

[33] G. Migoni, E. Kofman, and F.E. Cellier. Integración por Cuantificación de Sistemas Stiff. *Revista Iberoam. de Autom. e Inf. Industrial*, 4(3):97–106, 2007.

[34] J. Nutaro and B. Zeigler. On the Stability and Performance of Discrete Event Methods for Simulating Continuous Systems. *J. Computational Physics*, 227(1):797–819, 2007.

[35] S.P Nørsett and A. Wolfbrandt. Order conditions for rosenbrock type methods. *Numerische Mathematik*, (32):1–15, 1979.

[36] D Petcu. Experiments with an ode solver on a multiprocessor system. *Computers Mathematics with Applications*, (34):17–40, 2001.

[37] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. Numerical recipes. *Cambridge University Press*, 1986.

[38] G. Quesnel, R. Duboz, E. Ramat, and M. Traoré. VLE: a Multi-modeling and Simulation Environment. In *Proc. 2007 Summer Computer Simulation Conference*, pages 367–374, San Diego, California, 2007.

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

[40] A. Sandholm, P Bunus, and P Fritzson. A numeric library for use in modelica simulations with lapack, superlu, interpolation, and matrixio. *The Modelica Association*, (285-292), 2006.

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

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

[43] L. Shampine and M Reichelt. The matlab ode suite. *SIAM Journal on Scientific Computing*, 18(1):1–22, 1997.

[44] G Wainer. *Discrete Event Simulation and Modeling*, chapter Continuous System Simulation and Control. Kofman, E., Cellier, F. and Migoni,G. . CRC Press, 2008.

[45] Bernard Zeigler, Tag Gon Kim, and Herbert Praehofer. *Theory of Modeling and Simulation. Second edition.* Academic Press, New York, 2000.

[46] Bernard Zeigler and J.S. Lee. Theory of quantized systems: formal basis for DEVS/HLA distributed simulation environment. In *SPIE Proceedings*, pages 49–58, 1998.

Apéndice A

Código PowerDEVS de los integradores

En esta sección se presentan los codigos c de la implementasion en PowerDEVS de los integradores presentados en esta Tesis. Dado que todos ellos hacen uso de algunas funciones especificas que simplifican el codigo, las mismas se incluyen al final de la sección.

A.1. Integrador BQSS

```
#include "qss_integrator.h"
void qss_integrator::init(double t,...) {   //INICIALIZACIÓN
    //The 'parameters' variable contains the parameters transferred from the
  editor.
    va_list parameters;
    va_start(parameters,t);
    //To get a parameter: %Name% = va_arg(parameters,%Type%)
    //where:
    //      %Name% is the parameter name
    //      %Type% is the parameter type
    for (int i=0;i<10;i++) {
    y[i]=0;
    X[i]=0;
    q[i]=0;
    };
    dQmin=getScilabVar(fvar );
    fvar= va_arg(parameters,char*);
    dQrel=getScilabVar(fvar );
    fvar= va_arg(parameters,char*);
    X[0]=getScilabVar(fvar);
    dQ=fabs(X[0])*dQrel;
    if (dQ<dQmin){dQ=dQmin;};
    ep=dQmin/100;
    qs=X[0]+dQ;
    qi=X[0]-dQ;
```

```
    q[0]=qi;
    eps=1e-20;
    band=1;
    sigma=0;
}

double qss_integrator::ta(double t) {
    //This function return a double.
    return sigma;
}

void qss_integrator::dint(double t) {   //FUNCIÓN DE TRANSICIÓN INTERNA
    advance_time(X,sigma,1);
    if (sigma>eps) { // X arrives to a new level (q)
    dQ=fabs(X[0])*dQrel;
    if (dQ<dQmin){dQ=dQmin;};
    ep=dQ/100;
    if (X[1]>0) {
    qs=qs+dQ;
    q[0]=qs;
    qi=q[0]-2*dQ;
    }else{
    qi=qi-dQ;
    q[0]=qi;
    qs=q[0]+2*dQ;
    };
    } else { // slope change
    if(X[1]>0) {
    q[0]=qs;
    }else{
    q[0]=qi;
    };
    };
    if (X[1]!=0) {
    sigma=(q[0]-X[0])/X[1]+2*eps;
    } else {
    sigma=INF;
    };
    band=0;
}

void qss_integrator::dext(Event x, double t) { //FUNCIÓN DE TRANSICIÓN EXTERNA
  //The input event is in the 'x' variable.
  //where:
  //     'x.value' is the value
  //     'x.port' is the port number
  double *derx;
  double diffxq[10];
  double dt1;
  derx=(double*)x.value;
```

```
  if (e*X[1]!=0) {
      X[0]=X[0]+e*X[1];
      if (X[1]>0) { // we check the correct value of qi;
      if (X[0]-qi>=dQ+ep) qi=qi+dQ;
      } else { //we check the correct value of qs;
      if (qs-X[0]>=dQ+ep) qs=qs-dQ;
      };
  };
  X[1]=derx[0];
  if (t==0) {// initialization
      if (band==1) { //we need to send the output
      sigma=0;
      } else { //the output was already sent and there was a change in u
  if (X[1]*(q[0]-X[0])>=0) { //q is still ok
      if (X[1]!=0) {
  sigma=(q[0]-X[0])/X[1]+eps;
      } else {
  sigma=INF;
      };
  } else { // q is wrong, but we wait an infinitesimal until changing
it
      sigma=eps;
  };
      };
  } else {
      if ((sigma-e)<eps){
  sigma=2*eps;
      }else{
  if (e>0) { //derivative change
      if (X[1]*(q[0]-X[0])>=0) { // q was ok
  if (X[1]!=0) {
  sigma=(q[0]-X[0])/X[1]+2*eps;
  }else{
  sigma=INF;
  };
      } else { //we need to change q
  sigma=0;
      };
  }else { //we had already sent the value q
      if (X[1]*(q[0]-X[0])>=0) { // q is still ok
  if (X[1]!=0) {
  sigma=(q[0]-X[0])/X[1]+eps;
  } else {
  sigma=INF;
  };
      } else { // q is wrong, but we are close to u=0, so we set u=0.
  X[1]=0;
  sigma=INF;
      };
  };
```

```
      };
  };
}

Event qss_integrator::lambda(double t) {  //FUNCIÓN DE SALIDA
  //This function return an event:
  //     Event(%&Value%, %NroPort%)
  //where:
  //     %&Value% is a direction to the variable that contain the value.
  //     %NroPort% is the port number (from 0 to n-1)
  %%% Local Variables:
  %%% mode: latex
  %%% TeX-master: "thesis"
  %%% End:
  if (sigma<=eps) { //derivative change
  if (X[1]>0){
  y[0]=qs;
  }else{
  y[0]=qi;
  };
  } else { // X arrives to a new level (q)
  double dQnew;
  dQnew=fabs(X[0]+sigma*X[1])*dQrel;
  if (dQnew<dQmin){dQnew=dQmin;};
  if (X[1]>0){
  y[0]=q[0]+dQnew;
  }else{
  y[0]=q[0]-dQnew;
  };
  }
  return Event(&y,0);
```

A.2. Integrador LIQSS1

```
#include "liqss_integrator.h"
void liqss_integrator::init(double t,...) {    //INICIALIZACIÓN
    //The 'parameters' variable contains the parameters transferred from the
  editor.
    va_list parameters;
    va_start(parameters,t);
    //To get a parameter: %Name% = va_arg(parameters,%Type%)
    //where:
    //      %Name% is the parameter name
    //      %Type% is the parameter type
    for (int i=0;i<10;i++) {
    y[i]=0;
    X[i]=0;
    q[i]=0;
    u[i]=0;
```

```
    };
    char *fvar= va_arg(parameters,char*);
    dQmin=getScilabVar(fvar );
    fvar= va_arg(parameters,char*);
    dQrel=getScilabVar(fvar );
    fvar= va_arg(parameters,char*);
    X[0]=getScilabVar(fvar);
    dQ=fabs(X[0])*dQrel;
    q[0]=X[0];
    a=0;
    sigma=0;
}


double liqss_integrator::ta(double t) {
    //This function return a double.
    return sigma;
}


void liqss_integrator::dint(double t) { //FUNCIÓN DE TRANSICIÓN INTERNA
    dQ=dQrel*fabs(X[0]);
    if (dQ<dQmin)dQ=dQmin;
    if (X[1]==0){
sigma=INF;
    } else {
sigma=fabs(dQ/X[1]);
    };
}


void liqss_integrator::dext(Event x, double t) { //FUNCIÓN DE TRANSICIÓN EXTERNA
    //The input event is in the 'x' variable.
    //where:
    //     'x.value' is the value
    //     'x.port' is the port number
    double *derx;
    double diffxq[10];
    double dt1;
    derx=(double*)x.value;
    if ((e==0)&&(t>0)){
    //self feedback
    a=(X[1]-derx[0])/(q_old+dq_old-q[0]-dq);
    };
    u[0]=derx[0]-a*(q[0]+dq);
    X[0]=X[0]+X[1]*e;
    X[1]=derx[0];
    if (sigma>0){
    diffxq[1]=-X[1];
    diffxq[0]=q[0]-X[0]-dQ;
```

```
    sigma=minposroot(diffxq,1);
    diffxq[0]=q[0]-X[0]+dQ;
    dt1=minposroot(diffxq,1);
    if (dt1<sigma) sigma=dt1;
    if (fabs(X[0]-q[0])>dQ) sigma=0;
    };
}


Event liqss_integrator::lambda(double t) { //FUNCIÓN DE SALIDA
    //This function return an event:
    //     Event(%&Value%, %NroPort%)
    //where:
    //     %&Value% is a direction to the variable that contain the value.
    //     %NroPort% is the port number (from 0 to n-1)
    double ddx_est;
    q_old=q[0];
    dq_old=dq;
    advance_time(X,sigma,1);
    q[0]=X[0];
    if (X[1]>0) {
  //we try the upper value
  if (a*(q[0]+dQ)+u[0]>=0) {
  dq=+dQ; //OK
  } else {
    dq=-u[0]/a-q[0];
  };
  if(fabs(dq)>dQ){dq=dQ;}
    } else {
  //we try the lower value
  if (a*(q[0]-dQ)+u[0]<=0) {
  dq=-dQ; //OK
  } else {
  dq=-u[0]/a-q[0];
  };
  if(fabs(dq)>dQ){dq=-dQ;}
    };
    y[0]=q[0]+dq;
    return Event(&y,0);
}
```

A.3. Integrador LIQSS2

```
#include "liqss_integrator.h"
void liqss_integrator::init(double t,...) {
    va_list parameters;
    va_start(parameters,t);
    for (int i=0;i<10;i++) {
    y[i]=0;
```

```
    X[i]=0;
    q[i]=0;
    u[i]=0;
    };
    char *fvar= va_arg(parameters,char*);
    dQmin=getScilabVar(fvar );
    fvar= va_arg(parameters,char*);
    dQrel=getScilabVar(fvar );
    fvar= va_arg(parameters,char*);
    X[0]=getScilabVar(fvar);
    dQ=fabs(X[0])*dQrel;
    if (dQ<dQmin){dQ=dQmin;};
    q[0]=X[0];
    a=0;
    sigma=0;
}


double liqss_integrator::ta(double t) {
    return sigma;
}

void liqss_integrator::dint(double t) {
    dQ=dQrel*fabs(X[0]);
    if (dQ<dQmin)dQ=dQmin;
    if (X[2]==0){
    sigma=INF;
    } else {
    sigma=sqrt(fabs(dQ/X[2]));
    };
}


void liqss_integrator::dext(Event x, double t) {
    //The input event is in the 'x' variable.
    //where:
    //     'x.value' is the value
    //     'x.port' is the port number
    double *derx;
    double diffxq[10];
    double dt1;
    derx=(double*)x.value;
    //linear model estimation
    if ((e==0)&&(t>0)){
//self feedback
a=(X[1]-derx[0])/(q_old+dq_old-q[0]-dq);
    } else {
advance_time(q,e,1);
    };
    u[0]=derx[0]-a*(q[0]+dq);
```

```
    u[1]=derx[1]-a*q[1];

    X[0]=X[0]+X[1]*e+X[2]*e*e;
    X[1]=derx[0];
    X[2]=derx[1]/2;
    if (sigma>0){
diffxq[1]=q[1]-X[1];
diffxq[2]=-X[2];
diffxq[0]=q[0]-X[0]-dQ;
sigma=minposroot(diffxq,2);
diffxq[0]=q[0]-X[0]+dQ;
dt1=minposroot(diffxq,2);
if (dt1<sigma) sigma=dt1;
if(a!=0){
      diffxq[0]=u[1]+a*a*X[0]+a*u[0];
      diffxq[1]=a*a*X[1]+a*u[1];
      diffxq[0]=a*a*X[2]/2;
      dt1=minposroot(diffxq,2);
      if(dt1<sigma)sigma=dt1+1e-10;
}
if (fabs(X[0]-q[0])>dQ) sigma=0;
    };
}


Event liqss_integrator::lambda(double t) { //FUNCIÓN DE SALIDA
    //This function return an event:
    //     Event(%&Value%, %NroPort%)
    //where:
    //     %&Value% is a direction to the variable that contain the value.
    //     %NroPort% is the port number (from 0 to n-1)
    double ddx_est;
    advance_time(q,sigma,1);
    q_old=q[0];
    dq_old=dq;
    advance_time(u,sigma,1);
    advance_time(X,sigma,2);
    q[0]=X[0];
    if (X[1]*fabs(a)-u[1]>0) {
  //we try the upper value
  ddx_est=(a*(q[0]+dQ)+u[0])*fabs(a)-u[1];
  if (ddx_est>=0) {
  dq=-dQ; //OK
  } else {
    dq=-u[1]/a/a-u[0]/a-q[0];
  };
  if(fabs(dq)>dQ){dq=dQ;}
    } else {
  //we try the lower value
  ddx_est=(a*(q[0]-dQ)+u[0])*fabs(a)-u[1];
```

```
  if (ddx_est<=0) {
  dq=+dQ; //OK
  } else {
  dq=-u[1]/a/a-u[0]/a-q[0];
  };
  if(fabs(dq)>dQ){dq=-dQ;}
    };
    y[0]=q[0]+dq;
    y[1]=q[1];
    return Event(&y,0);
}
```

A.4. Integrador LIQSS3

```
#include "liqss_integrator.h"
void liqss_integrator::init(double t,...) {  //INICIALIZACIÓN
    va_list parameters;
    va_start(parameters,t);
    for (int i=0;i<10;i++) {
    y[i]=0;
    X[i]=0;
    q[i]=0;
    u[i]=0;
    };
    char *fvar= va_arg(parameters,char*);
    dQmin=getScilabVar(fvar );
    fvar= va_arg(parameters,char*);
    dQrel=getScilabVar(fvar );
    fvar= va_arg(parameters,char*);
    X[0]=getScilabVar(fvar);
    dQ=fabs(X[0])*dQrel;
    q[0]=X[0];
    a=0;
    sigma=0;
}


double liqss_integrator::ta(double t) {
    return sigma;
}


void liqss_integrator::dint(double t) {  //FUNCIÓN DE TRANSICIÓN INTERNA
    dQ=dQrel*fabs(X[0]);
    if (dQ<dQmin)dQ=dQmin;
    if (X[3]==0) {
    sigma=INF;
    } else {
    sigma=pow(fabs(dQ/X[3]),1.0/3);
```

```
    };
    double diffxq[10];
    diffxq[1]=q[1]-X[1];
    diffxq[2]=q[2]-X[2];
    diffxq[3]=-X[3];
    diffxq[0]=q[0]-X[0]-dQ;
    sigma=minposroot(diffxq,3);
    diffxq[0]=q[0]-X[0]+dQ;
    double dt1;
    dt1=minposroot(diffxq,3);
    if (dt1<sigma) sigma=dt1;
}


void liqss_integrator::dext(Event x, double t) { //FUNCIÓN DE TRANSICIÓN EXTERNA
    //The input event is in the 'x' variable.
    //where:
    //     'x.value' is the value
    //     'x.port' is the port number
    double *derx;
    double diffxq[10];
    double dt1;
    derx=(double*)x.value;
    //linear model estimation
    if (((e==0)&&(sigma>0))&&(t>0)){
//self feedback
a=(X[1]-derx[0])/(q_old+dq_old-q[0]-dq);

    } else {
advance_time(q,e,2);
    };
    u[0]=derx[0]-a*(q[0]+dq);
    u[1]=derx[1]-a*q[1];
    u[2]=derx[2]-a*q[2];
    X[0]=X[0]+X[1]*e+X[2]*e*e+X[3]*e*e*e;
    X[1]=derx[0];
    X[2]=derx[1]/2;
    X[3]=derx[2]/3;
    if (sigma>0){
diffxq[1]=q[1]-X[1];
diffxq[2]=q[2]-X[2];
diffxq[3]=-X[3];
diffxq[0]=q[0]-X[0]-dQ;
sigma=minposroot(diffxq,3);
diffxq[0]=q[0]-X[0]+dQ;
dt1=minposroot(diffxq,3);
if (dt1<sigma) sigma=dt1;
if(a!=0){   //busca el pto donde se da vuelta x[2]
    diffxq[1]=X[1]-(-2*u[2]/a/a-u[1]/a);
    diffxq[2]=X[2]+u[2]/a;
```

```
    diffxq[3]=X[3];
    diffxq[0]=X[0]-(-2*u[2]/a/a/a-u[1]/a/a-u[0]/a);
    dt1=minposroot(diffxq,3);
    if(dt1<sigma)sigma=dt1+1e-10;
}
if (fabs(X[0]-q[0])>dQ) sigma=0;
    };
}


Event liqss_integrator::lambda(double t) {  //FUNCIÓN DE SALIDA
    //This function return an event:
    //     Event(%&Value%, %NroPort%)
    //where:
    //     %&Value% is a direction to the variable that contain the value.
    //     %NroPort% is the port number (from 0 to n-1)
    double ddx_est;
    advance_time(q,sigma,2);
    q_old=q[0];
    dq_old=dq;
    advance_time(u,sigma,2);
    advance_time(X,sigma,3);
    q[0]=X[0];
    if (X[2]*fabs(a)-u[2]<0) {
//we try the upper value
ddx_est=(a* (a* (q[0]+dQ) +u[0] )+u[1])*fabs(a)/2-u[2];
if (ddx_est<=0) {
dq=dQ; //OK
} else {
  dq=-2*u[2]/a/a/a-u[1]/a/a-u[0]/a-q[0];
};
if(fabs(dq)>dQ){dq=dQ;}
    } else {
//we try the lower value\section{Integrador LIQSS3}\label{ap:liqss3code}
ddx_est=(a*(a*(q[0]-dQ)+u[0])+u[1])*fabs(a)/2-u[2];
if (ddx_est>=0) {
dq=-dQ; //OK
} else {
dq=-2*u[2]/a/a/a-u[1]/a/a-u[0]/a-q[0];
};
if(fabs(dq)>dQ){dq=-dQ;}
    };
    q[1]=a*(q[0]+dq)+u[0];
    q[2]=a*q[1]/2+u[1]/2;
    y[0]=q[0]+dq;
    y[1]=q[1];
    y[2]=q[2];
    return Event(&y,0);
}
```

A.5. Integrador CQSS

```
#include "qss_integrator.h"
void qss_integrator::init(double t,...) {  //INICIALIZACIÓN
    va_list parameters;
    va_start(parameters,t);
    for (int i=0;i<10;i++) {
    y[i]=0;
    X[i]=0;
    q[i]=0;
    };
    char *fvar= va_arg(parameters,char*);
    dQmin=getScilabVar(fvar );
    fvar= va_arg(parameters,char*);
    dQrel=getScilabVar(fvar );
    fvar= va_arg(parameters,char*);
    X[0]=getScilabVar(fvar);
    dQ=fabs(X[0])*dQrel;
    if (dQ<dQmin){dQ=dQmin;};
    ep=dQmin/100;
    qs=X[0]+dQ;
    qi=X[0]-dQ;
    q[0]=qi;
    eps=1e-20;
    band=1;
    sigma=0;
}


double qss_integrator::ta(double t) {
    return sigma;
}


void qss_integrator::dint(double t) {   //FUNCIÓN DE TRANSICIÓN INTERNA
    advance_time(X,sigma,1);
    if (sigma>eps) { // X arrives to a new level (q)
    dQ=fabs(X[0])*dQrel;
    if (dQ<dQmin){dQ=dQmin;};
    ep=dQ/100;

    if (X[1]>0) {
    qs=qs+dQ;
    q[0]=qs;
    qi=q[0]-2*dQ;
    }else{
    qi=qi-dQ;
    q[0]=qi;
    qs=q[0]+2*dQ;
    };
```

```
    } else { // slope change
    if(X[1]>0) {
    q[0]=qs;
    }else{
    q[0]=qi;
    };
    };
    if (X[1]!=0) {
    sigma=(q[0]-X[0])/X[1]+2*eps;
    } else {
    sigma=INF;
    };
    band=0;
}


void qss_integrator::dext(Event x, double t) {  //FUNCIÓN DE TRANSICIÓN EXTERNA
    double *derx;
    double diffxq[10];
    double dt1;
    derx=(double*)x.value;
    if (e*X[1]!=0) {
X[0]=X[0]+e*X[1];
if (X[1]>0) { // we check the correct value of qi;
if (X[0]-qi>=dQ+ep) qi=qi+dQ;
} else { //we check the correct value of qs;
if (qs-X[0]>=dQ+ep) qs=qs-dQ;
};
    };
    X[1]=derx[0];
    if (t==0) {// initialization
if (band==1) { //we need to send the output
    sigma=0;
} else { //the output was already sent and there was a change in u
    if (X[1]*(q[0]-X[0])>=0) { //q is still ok
if (X[1]!=0) {
    sigma=(q[0]-X[0])/X[1]+eps;
} else {
    sigma=INF;
};
    } else {//q is wrong,but we wait an infinitesimal until changing it
    sigma=eps;
    };
};
    } else {
if ((sigma-e)<eps){
    sigma=2*eps;
}else{
    if (e>0) { //derivative change
if (X[1]*(q[0]-X[0])>=0) { // q was ok
```

```
    if (X[1]!=0) {
sigma=(q[0]-X[0])/X[1]+2*eps;
    }else{
sigma=INF;
    };
} else { //we need to change q
    if (X[1]>0){
sigma=(qs-X[0])/X[1]+2*eps;
q[0]=qs;
    }else{
sigma=(qi-X[0])/X[1]+2*eps;
q[0]=qi;
    };
};
    }else { //we had already sent the value q
if (X[1]*(q[0]-X[0])>=0) { // q is still ok
    if (X[1]!=0) {
    sigma=(q[0]-X[0])/X[1]+eps;
    } else {
    sigma=INF;
    };
} else { // q is wrong, but we are close to u=0, so we set u=0.
    X[1]=0;
    sigma=INF;
};
    };
};
    };
}

Event qss_integrator::lambda(double t) {  //FUNCIÓN DE SALIDA
    if (sigma<=eps) { //derivative change
if (X[1]>0){
y[0]=q[0];
} else {
y[0]=qi;
};
    } else { // X arrives to a new level (q)
if (X[1]>0){
y[0]=q[0]+dQ;
}else{
y[0]=q[0]-dQ;
};
    };
    y[0]=(y[0]+X[0]+sigma*X[1])/2;
    return Event(&y,0);
}
```

A.6. Funciones comunes a todos los integradores

Función minsroot(coef,order)

Esta función devuelve el menor valor positivo que hace cero el polinomio de orden 'order' cuyos coeficientes son 'coef.

```
double minposroot(double *coeff, int order) {
  double mpr;
  switch (order) {
  case 0:
        mpr=INF;
break;
  case 1:
if (coeff[1]==0) {
  mpr=INF;
} else {
  mpr=-coeff[0]/coeff[1];
};
if (mpr<0) mpr=INF;
break;
  case 2:
if (coeff[2]==0){
  if (coeff[1]==0) {
    mpr=INF;
  } else {
    mpr=-coeff[0]/coeff[1];
  };
  if (mpr<0) mpr=INF;
} else {
  double disc;
  disc=coeff[1]*coeff[1]-4*coeff[2]*coeff[0];
  if (disc<0) {
    //no real roots
    mpr=INF;
  } else {
    double sd,r1;
    sd=sqrt(disc);
    r1=(-coeff[1]+sd)/2/coeff[2];
    if (r1>0) {
      mpr=r1;
    } else {
      mpr=INF;
    };
    r1=(-coeff[1]-sd)/2/coeff[2];
    if ((r1>0)&&(r1<mpr)) mpr=r1;
  };
};
      break;
```

```
  case 3:
    if ((coeff[3]==0)||(1000*fabs(coeff[3])<fabs(coeff[2]))) {
      mpr=minposroot(coeff,2);
    } else {
      double q,r,disc;
      q=(3*coeff[3]*coeff[1]-coeff[2]*coeff[2])/9/coeff[3]/coeff[3];
r=(9*coeff[3]*coeff[2]*coeff[1]-27*coeff[3]*coeff[3]*coeff[0]-2*coeff[2]*coeff[2
]*coeff[2])/54/coeff[3]/coeff[3]/coeff[3];
      disc=q*q*q+r*r;
      mpr=INF;
      if (disc>=0){
  //only one real root
  double sd,s,t,r1;
  sd=sqrt(disc);
  if (r+sd>0) {
    s=pow(r+sd,1.0/3);
  } else {
    s=-pow(fabs(r+sd),1.0/3);
  };
  if (r-sd>0) {
    t=pow(r-sd,1.0/3);
  } else {
    t=-pow(fabs(r-sd),1.0/3);
  };
  r1=s+t-coeff[2]/3/coeff[3];
  if (r1>0) mpr=r1;
      }else{
  //three real roots
  double rho,th,rho13,costh3,sinth3,spt,smti32,r1;
  rho=sqrt(-q*q*q);
  th=acos(r/rho);
  rho13=pow(rho,1.0/3);
  costh3=cos(th/3);
  sinth3=sin(th/3);
  spt=rho13*2*costh3;
  smti32=-rho13*sinth3*sqrt(3);
  r1=spt-coeff[2]/3/coeff[3];
  if (r1>0) mpr=r1;
  r1=-spt/2-coeff[2]/3/coeff[3]+smti32;
  if ((r1>0)&&(r1<mpr)) mpr=r1;
  r1=r1-2*smti32;
  if ((r1>0)&&(r1<mpr)) mpr=r1;
      };
    };
    return mpr;
}
```

Función advance_time(coef,dt,order)

```
void advance_time(double *coeff, double dt, int order) {
```

```
    if (order==-1) {
if (coeff[4]!=0){
order=4;
}
else if (coeff[3]!=0){
order=3;
}
else if (coeff[2]!=0) {
order=2;
}
else {
order=1;
};
    };
switch (order) {
  case 0:
      break;
  case 1:
      coeff[0]=coeff[0]+dt*coeff[1];
      break;
  case 2:
      coeff[0]=coeff[0]+dt*coeff[1]+dt*dt*coeff[2];
      coeff[1]=coeff[1]+2*coeff[2]*dt;
      break;
  case 3:
      coeff[0]=coeff[0]+dt*coeff[1]+dt*dt*coeff[2]+dt*dt*dt*coeff[3];
      coeff[1]=coeff[1]+2*coeff[2]*dt+3*coeff[3]*dt*dt;
      coeff[2]=coeff[2]+3*coeff[3]*dt;
      break;
  };
};
```