BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS FÍSICO MATEMÁTICAS

VALIDACIÓN Y SELECCIÓN DE ALGORITMOS NUMÉRICOS PARA MODELOS DE SIMULACIÓN DE PROCESOS DINÁMICOS

T E S I S

presentada para obtener el título de:

LICENCIATURA EN MATEMÁTICAS APLICADAS

PRESENTA:

JESSICA ZAQUEROS MARTÍNEZ

Asesores de tesis:

DR. GUSTAVO RODRÍGUEZ GÓMEZ (CCC,INAOE)
DR. CARLOS GUILLÉN GALVÁN (FCFM, BUAP)

Marzo, 2018

A ti lector

Agradecimientos

Estoy totalmente agradecida con las personas que de alguna u otra forma me ayudaron a llevar a buen término este trabajo. Primero, quiero agradecer al doctor Gustavo Rodríguez por ser mi guía en este proyecto, por su paciencia y todo el tiempo dedicado, por sus consejos y anécdotas. También, agradezco al doctor Carlos Guillén por todo su apoyo y sus consejos, y a mis sinodales por sus consejos y correcciones.

A quién también agradezco infinitamente es a Bru, por estar ahí siempre para mí, por toda la fortaleza y apoyo incondicional que siempre me otorga.

A mis padres y hermanos les doy las gracias por apoyarme siempre.

A mis amigos: Jerox, Andrés, Mary, Gabo y Dani por creer en mí y alentarme a seguir hasta el final.

Y finalmente, agradezco a Dios por permitirme concluir este trabajo.

Introducción

La simulación es una herramienta de trabajo que nació lentamente, en paralelo con la aparición de la computadora, y que poco a poco se ha impuesto gracias a la mayor velocidad y capacidad que las computadoras han ido ofreciendo a la industria. Con el surgimiento, en la década de los 50s, de los lenguajes orientados a procedimientos como Fortran (ver Metcalt [25]) se da un avance significativo a este nuevo instrumento. Un antecesor de los actuales simuladores de procesos es el programa PACER que realizó su primera prueba práctica en 1964. PACER es un acrónimo para Process Assembly Case Evaluator Routine, el cual fue ideado por el profesor Paul T. Shannon mientras trabaja para la Humble Oil and Refining Company, de acuerdo con Martin [23].

En 1967 se fundó el **Winter Simulation Conference (WSC)**, lugar donde desde entonces y hasta ahora se archivan los lenguajes de simulación y aplicaciones derivadas, siendo en la actualidad el referente en lo que a avances en el campo de los sistemas de simulación se refiere, ver Montevechi et al. [26].

Los últimos años han sido testigos de un crecimiento significativo del interés por el modelado y simulación de sistemas físicos. Un factor clave en este crecimiento ha sido el desarrollo de los lenguajes de simulación orientados a objetos basados en ecuaciones que permiten modelar sistemas dinámicos complejos, por ejemplo, sistemas mecánicos, eléctricos, entre otros.

Los modelos de simulación están en uso todos los días y así simulación es un concepto que no es ajeno a nosotros. Por ejemplo, los meteorólogos a diario nos muestran simulaciones del sistema meteorológico, donde vemos el movimiento de los frentes atmosféricos para los días venideros. Muchos tenemos consolas de juegos que simulan una gran variedad de actividades, lo que nos permite poner a prueba nuestras habilidades como pilotos de carreras, aventureros, etc. Las simulaciones no necesitan basarse en una computadora. Modelos de ferrocarriles y barcos de control remoto son ejemplos familiares de simulaciones físicas.

Existen dos tipos de simulación: la simulación estática, que imita un sistema en un punto en el tiempo, y la simulación dinámica, que imita un sistema a medida que avanza a través del tiempo. En esta investigación nos centraremos en la simulación

dinámica.

De forma intuitiva se define a la simulación como:

Amplia colección de métodos y aplicaciones que imitan el comportamiento real de los sistemas por medio de una computadora con el software apropiado.

Dentro de este contexto, la simulación dinámica es una herramienta para construir los simuladores de procesos, los cuales se pueden enfocar hacia el análisis del proceso o para entrenar operadores de equipos complejos. Los primeros simuladores para entrenamiento de operadores aparecieron en la industria aeronáutica, después de la segunda guerra mundial (Rodríguez-Gómez [32]). En la actualidad se producen simuladores de equipos tan diversos como helicópteros, sistemas de transporte colectivo, vehículos militares, plantas generadoras de energía eléctrica, etc.

Los simuladores de entrenamiento trabajan en **Tiempo Real (TR)** puesto que tienen como objetivo reproducir fielmente el ambiente de trabajo de un operador, con el fin de familiarizarlo con el funcionamiento (normal y anormal) del equipo y adiestrarlo en su operación con base en ejercicios repetidos, lo cual mejora el desempeño del operador en el equipo en cuestión. En este contexto, se entiende por **TR** que: *El tiempo de la* ***Unidad Central de Procesamiento (UCP)*** *empleado para resolver numéricamente el modelo no debe exceder el intervalo de tiempo simulado.*

Entrenar a un operador en un simulador tiene beneficios humanos, ecológicos y económicos, por ejemplo, un operador de una **Central Termoeléctrica (CT)** entrenado en un simulador puede evitar evoluciones de estados anormales de una **CT** que dieran origen a fallas en la planta que ocasionarían explosiones, por lo que se salvarían vidas humanas, no se dañaría al medio ambiente y se evitarían pérdidas materiales.

Definición del problema

Para construir un simulador de un proceso (en este trabajo le llamamos también planta), se sustituye la planta física por un conjunto de modelos matemáticos. Estos modelos en general están representados por Sistemas de **Ecuaciones Algebraico Diferenciales (EADs).** Dependiendo de la orientación del simulador las ecuaciones diferenciales pueden ser ordinarias o parciales. En este caso, el trabajo se centra en simuladores de procesos en los cuales las diferenciales son ordinarias con valores iniciales.

Los modelos que representan al proceso físico simulado son construidos por los ingenieros de procesos a partir de los datos y diagramas de la planta. Estos introducen las hipótesis y simplificaciones necesarias para que los modelos cumplan con el propósito

del simulador y después lo validan.

La validación es el proceso de asegurar que los modelos sean suficientemente exactos para el propósito que fueron realizados (Roobinson [28]). La forma en que los ingenieros de procesos validan los modelos matemáticos de la planta es por medio de la simulación digital. Seleccionan diferentes condiciones iniciales para los modelos y realizan las simulaciones empleando métodos numéricos. Finalmente, comparan los datos obtenidos de las simulaciones con los datos reales de la planta, si los errores se encuentran dentro de márgenes aceptables los modelos son aprobados.

De acuerdo con la ingeniería de software no sólo es necesario validar el modelo, también es importante verificarlo. La verificación es el proceso de garantizar que el diseño del modelo (modelo conceptual) se ha transformado en un modelo de computadora con suficiente precisión (Robinson [28]).

Cuando los modelos, en el proceso de validación, no son aceptados es porque aparecen resultados que no corresponden a la realidad, por lo que es imprescindible ubicar las fuentes que dieron origen a estos errores.

La mayoría de las veces pensamos que las fuentes de error sólo provienen de las estrategias numéricas. Sin embargo, debido a que el proceso de elaboración del modelo matemático que representa a la planta es un proceso iterativo, desde la concepción del modelo hasta su implementación, hay diferentes fuentes de error. Por ejemplo, unas de estas pueden ser las hipótesis y simplificaciones hechas para construir el modelo matemático, porque si estas no fueron las adecuadas, el modelo no representa apropiadamente al proceso físico. También, la gran cantidad de **EADs** puede ocasionar que su implementación computacional sea errónea puesto que el orden de ejecución podría no ser el correcto, error muy generalizado en práctica, tal como lo menciona Gallardo [13]. Otra fuente de error es la selección inadecuada de los métodos numéricos, así como la alteración de variables de estado por parte del usuario, debido a los requerimientos del control de procesos, y que no se deberían alterar dado que impactan de forma negativa a las técnicas numéricas. Otras causas de error son la discretización del modelo o error de truncamiento, la incertidumbre en los datos del problema, los errores de redondeo, problemas mal condicionados y algoritmos inestables, sólo por mencionar algunos.

Debido al tamaño del modelo matemático, es decir el gran número de **EADs**, localizar este tipo de errores no es una tarea de envergadura simple para los ingenieros de procesos.

Hipótesis de la tesis

La teoría de estabilidad numérica, para llevar a cabo el análisis de estabilidad de los métodos numéricos, y la teoría de grafos, para realizar el análisis estructural de los modelos matemáticos, nos permiten realizar simulaciones eficientes en **TR**.

Objetivo

El objetivo general de este trabajo es proponer una metodología, dentro del contexto de simulación de procesos, que permita ubicar las fuentes de error y seleccionar los métodos numéricos asociados a los modelos, tal que cumplan con las restricciones de ejecución de tiempo y precisión numérica requerida.

Metodología

En esta sección se presenta la metodología que se sigue para alcanzar los objetivos de este proyecto.

1. La selección del algoritmo que proporcione el orden de ejecución correcto del sistema algebraico-diferencial se hará con base en la teoría de grafos.

2. Para seleccionar las estrategias numéricas para resolver sistemas ecuaciones diferenciales ordinarias y sistemas de ecuaciones no lineales de fuentes confiables se revisarán repositorios como Netlib y el proporcionado por la revista **Transactions on Mathematical Software (TOMS)**, que poseen colecciones de software matemático validado.

3. Por medio de la revisión bibliográfica se seleccionarán un conjunto de modelos de procesos físicos que estén constituidos por **Ecuaciones Diferenciales Ordinarias (EDOs)**, como sistemas eléctricos, mecánicos, hidráulicos, para realizar pruebas.

4. Se trabajará con algoritmos de diferencias de 1er y 2do orden para aproximar las matrices jacobianas.

5. Se procederá a diseñar e implementar en Fortran un algoritmo para probar si el ordenamiento de las ecuaciones algebraico-diferenciales del modelo es el correcto.

6. A través de la teoría de estabilidad absoluta de los métodos numéricos para resolver **EDOs** se seleccionarán los algoritmos que cumplan con las restricciones de estabilidad, tiempo y precisión numérica requeridos.

7. A partir de la teoría de **EDOs** se encontrará la formulación matemática apropiada para las **EDOs** que tienen limitadores en sus variables de estado por los requerimientos del control.

8. Se utilizará el lenguaje Fortran para integrar todos los programas en uno sólo y validar dicho programa.

Propuesta de solución

La propuesta de esta tesis es una metodología para ayudar a los ingenieros de procesos a ubicar el origen de los distintos tipos de errores que se presentan durante el diseño, desarrollo y simulación de sus modelos y en consecuencia seleccionar las técnicas numéricas apropiadas para que la simulación cumpla con las restricciones de diseño.

Preguntas de investigación

- ¿Cuáles son las fuentes de errores que surgen cuando se simula un proceso físico?
- ¿Cómo se detectan tales errores?
- ¿Cuáles son los métodos numéricos que permiten cumplir con las restricciones de ejecución de tiempo y precisión numérica requerida al simular un proceso físico?

Contribuciones de la tesis

La principal contribución de esta tesis será brindar una metodología que sea aplicable en la simulación de procesos para validar la simulación y cumplir con los propósitos de ésta. Además, se proporcionarán: un algoritmo que indica si la estructura de las ecuaciones del sistema es la correcta, y un planteamiento teórico de las **EDOs** cuya solución se encuentra limitada debido a las restricciones físicas del proceso real.

Organización de la tesis

En esta sección presentamos brevemente el tema central de cada uno de los capítulos que conforman este trabajo. En el capítulo 1 exponemos de manera sucinta aquellos tópicos en los que se fundamenta esta tesis. En el capítulo 2 presentamos una recopilación de trabajos relevantes relacionados al tema que ocupa a este proyecto. En el capítulo 3 exponemos detalladamente el algoritmo propuesto. En el capítulo 4 presentamos las pruebas realizadas. En el capítulo 5 exponemos las conclusiones generales de la tesis, así como el posible trabajo a futuro.

Por otro lado, en este trabajo utilizamos la siguiente convención para referirnos a las secciones del documento: $x.y.z$, donde x es el número de capítulo, y el número de sección y z el número de subsección.

Índice general

Agradecimientos iii

Introducción iv

Acrónimos xiii

Índice de figuras xv

Índice de tablas xix

1. Preliminares 1

1.1. Métodos numéricos para EDOs con VIs 1

1.2. Métodos Multipasos Lineales 2

1.2.1. Regiones de Estabilidad Absoluta de los MMLs 2

1.3. Cotas de error de los MML 3

1.4. Métodos Runge Kutta 4

1.4.1. Regiones de Estabilidad Absoluta de los métodos R–K 5

1.5. Cotas de error de los métodos R–K 6

1.6. Análisis Estructural 8

1.7. Algoritmos de Ordenamiento 10

1.8. Selección del paso y método de integración 12

1.9. Limitadores 13

2. Trabajo Relacionado **14**

2.1. Determinación del método numérico y paso de integración 14

2.2. Análisis estructural en la simulación de procesos 20

3. Planteamiento del problema y metodología propuesta **24**

3.1. Importancia del problema dentro de la simulación 24

3.2. Metodología propuesta 27

3.2.1. Primera parte de la metodología 27

3.2.2. Segunda parte de la metodología 29

3.2.3. Tercera parte de la metodología 29

3.2.4. Cuarta parte de la metodología 30

3.2.5. Quinta parte de la metodología 31

3.3. Ejemplos académicos 31

3.3.1. Ejemplo de análisis estructural 32

3.3.2. Ejemplificación de limitadores 35

3.4. Procedimiento para hallar la Región de Estabilidad Absoluta (REA) de los MML 38

3.4.1. Estimación de una cota superior para el paso de integración de los métodos R–K 39

3.4.2. Cota superior para el método de Euler 39

4. Experimentos **42**

4.1. Tres Reactores Tipo Tanque con Control PI 42

4.1.1. Resultados de la aplicación de la metodología propuesta 42

4.2. Conjunto de Resortes Acoplados 48

4.2.1. Resultados de la aplicación de la metodología propuesta 51

4.3. Modelo no lineal de un *quadrotor* 54

4.3.1. Resultados de la aplicación de la metodología propuesta 56

4.4. Modelo de una planta termoeléctrica 62

4.4.1. Resultados de la aplicación de la metodología propuesta 64

Conclusiones **72**

5. Conclusiones **72**

A. Cota superior para los métodos R–K **74**

A.1. Cota superior para los métodos R–K de orden dos 74

A.2. Cota superior para los métodos R–K de orden tres 75

A.3. Cota superior para los métodos R–K de orden cuatro 76

B. Condiciones de simulación **78**

C. Funcionamiento del algoritmo y software de ordenamiento **79**

D. Tabla de la prueba de orden del modelo de la planta termoeléctrica **82**

Bibliografía **85**

Acrónimos

A continuación mostramos los acrónimos utilizados en este trabajo, decidimos realizar dos grupos: en español y en inglés, debido a que no quisimos traducir los acrónimos que mayoritariamente en la literatura se encuentran en inglés.

En español:

CT: Central Termoeléctrica

EAs: Ecuaciones Algebraicas

EADs: Ecuaciones Algebraico Diferenciales

EDOs: Ecuaciones Diferenciales Ordinarias

EDPs: Ecuaciones Diferenciales Parciales

MML: Método Multipaso Lineal

PI: Proporcional Integral

REA: Región de Estabilidad Absoluta

TR: Tiempo Real

UCP: Unidad Central de Procesamiento

UT: Unidad de generación Termoélectrica

VIs: Valores Iniciales

En inglés:

ACM: Association for Computing Machinery

BDF: Backward Differentiation Formulas

MDA: Model Driven Architecture

ODE: Ordinary Differential Equations

OMG: Object Management Group

R–K: Runge–Kutta

TOMS: Transactions on Mathematical Software

WSC: Winter Simulation Conference

Índice de figuras

1.1. Gráfica de Euler hacia delante y Euler con retraso. 9

3.1. Diagrama que muestra el desarrollo de un modelo. 27

3.2. Primera parte de la metodología. 28

3.3. Esquema de la prueba estructural mejorada. 28

3.4. Segunda parte de la metodología. 29

3.5. Tercera parte de la metodología. 30

3.6. Cuarta parte de la metodología. 31

3.7. Quinta parte de la metodología. 32

3.8. Gráfica de la solución sin limitar y la solución esperada. 35

3.9. Diagrama que representa la solución a los limitadores de forma pragmática. 36

3.10. Diagrama que representa nuestra propuesta para dar solución a los limitadores. 36

3.11. Gráfica de la solución pragmática y la solución sin limitar. 37

3.12. Gráfica de la solución propuesta y la solución correcta. 37

3.13. Regiones de estabilidad absoluta de los Métodos R–K. 41

4.1. Gráfica de la distribución los valores característicos con estructura correcta del modelo de los tres reactores. 44

4.2. Gráfica de barras con el paso de integración recomendado para cada método integrador con estructura correcta del modelo de los tres reactores. 44

4.3. Gráfica de barras del costo computacional para cada método de integración con estructura correcta del modelo de los tres reactores. . . 45

4.4. Gráfica de la distribución de los valores característicos con estructura incorrecta del modelo de los tres reactores. 47

4.5. Gráfica de barras con el paso de integración recomendado para cada método integrador con estructura incorrecta del modelo de los tres reactores. 47

4.6. Gráfica de barras del costo computacional para cada método de integración con estructura incorrecta del modelo de los tres reactores. . 48

4.7. Gráfica de la variable y_3 y el costo computacional cuando la simulación posee una estructura correcta del modelo de los tres reactores. 49

4.8. Gráfica de la variable y_3 y el costo computacional cuando la simulación posee una estructura incorrecta del modelo de los tres reactores. 49

4.9. Gráfica de la distribución los valores característicos con estructura correcta del modelo de los resortes. 52

4.10. Gráfica de barras con el paso de integración recomendado para cada método integrador con estructura correcta del modelo de los resortes. . 53

4.11. Gráfica de barras del costo computacional para cada método de integración con estructura correcta del modelo de los resortes. 53

4.12. Gráfica de la distribución los valores característicos con estructura correcta del modelo del *quadrotor*. 58

4.13. Gráfica de barras con el paso de integración recomendado para cada método integrador con estructura correcta del modelo del *quadrotor*. . . 59

4.14. Gráfica de barras del costo computacional para cada método de integración con estructura correcta del modelo del *quadrotor*. 59

4.15. Gráfica de la distribución de los valores característicos con estructura incorrecta del modelo del *quadrotor*. 61

4.16. Gráfica de barras con el paso de integración recomendado para cada método integrador con estructura incorrecta del modelo del *quadrotor*. . 61

4.17. Gráfica de barras del costo computacional para cada método de integración con estructura incorrecta del modelo del *quadrotor*. 62

4.18. Gráfica de la variable z y el costo computacional cuando la simulación posee una estructura correcta del modelo del *quadrotor*. 63

4.19. Gráfica de la variable z y el costo computacional cuando la simulación posee una estructura incorrecta del modelo del *quadrotor*. 63

4.20. Módulos de la Unidad Termoeléctrica, imagen tomada de [10] 64

4.21. Gráfica de la distribución los valores característicos con estructura correcta del modelo de la planta termoeléctrica. 66

4.22. Gráfica de barras con el paso de integración recomendado para cada método integrador con estructura correcta del modelo de la planta termoeléctrica. 66

4.23. Gráfica de barras del costo computacional para cada método de integración con estructura correcta del modelo de la planta termoeléctrica. 67

4.24. Gráfica de la distribución de los valores característicos con estructura incorrecta del modelo de la planta termoeléctrica. 69

4.25. Gráfica de barras con el paso de integración recomendado para cada método integrador con estructura incorrecta del modelo de la planta termoeléctrica. 69

4.26. Gráfica de barras del costo computacional para cada método de integración con estructura incorrecta del modelo de la planta termoeléctrica. 70

4.27. Gráfica de la variable $QWWGM$ y el costo computacional cuando la simulación posee una estructura correcta del modelo de la planta termoeléctrica. 71

4.28. Gráfica de la variable $QWWGM$ y el costo computacional cuando la simulación posee una estructura incorrecta del modelo de la planta termoeléctrica. 71

C.1. Matriz de incidencia formada por el algoritmo de ordenamiento de la Unidad Termoeléctrica. 80

C.2. Matriz de incidencia de salida del algoritmo de ordenamiento de la Unidad Termoeléctrica. 81

Índice de tablas

3.1. Descripción de parámetros. 33

4.1. Resultados de la *Prueba estructural mejorada* del modelo de los tres reactores. 46

4.2. Tabla de parámetros del modelo de los resortes. 51

4.3. Descripción de variables del modelo del *quadrotor*. 55

4.4. Descripción de parámetros del modelo del *quadrotor*. 55

4.5. Valores que toman las constantes del control PI del *quadrotor*. 56

4.6. Resultados de la *Prueba estructural mejorada* del modelo del *quadrotor*. 56

4.7. Resultados de la *Prueba estructural mejorada* para el tiempo t = 0 s del modelo de la planta termoeléctrica. 68

D.1. Resultados de la *Prueba estructural mejorada* del modelo de la planta termoeléctrica. 84

Capítulo 1

Preliminares

En el presente capítulo proporcionamos las nociones básicas para la comprensión de la metodología propuesta. En la primera parte abordamos la teoría básica de los métodos numéricos para aproximar la solución de **EDOs** con **Valores Iniciales (VIs)**, posteriormente presentamos los conceptos básicos del análisis estructural y elementos de la teoría de grafos necesarios para este tema.

1.1. Métodos numéricos para EDOs con VIs

Consideremos el problema con **VIs** para una única ecuación diferencial ordinaria de primer orden

$$y' = f(t, y), \quad y(a) = y_a. \tag{1.1.1}$$

El siguiente teorema, cuya demostración está dada por Henrici [17], establece las condiciones en $f(t, y)$ que garantizan la existencia de una única solución de (1.1.1).

Teorema 1 *Sea $f(t, y)$ definida y continua para todos los puntos (t, y) en la región D, definida por $a \leq t \leq b$, $-\infty < y < \infty$, a y b finitos, sea también $L > 0$ una constante tal que, para todo t, y, y^* con (t, y) y (t, y^*) en D,*

$$| f(t, y) - f(t, y^*) | \leq L | y - y^* | .$$

Entonces, si y_a es cualquier número dado, existe una única solución $y(t)$ del problema (1.1.1), donde $y(t)$ es continua y diferenciable para todo (t, y) en D.

Una propiedad esencial de la mayoría de los métodos computacionales para resolver (1.1.1) es la discretización, la cual busca una solución aproximada, no en el intervalo continuo $a \leq t \leq b$, sino en el conjunto discreto de puntos $\{t_n \mid n = 0, 1, \ldots, (b-a)/h\}$ donde la sucesión de puntos $\{t_n\}$ está definida por $t_n = a + nh$, $n = 0, 1, 2, \ldots$, con $h > 0$ constante, denominada la *longitud del paso de integración* y llamada brevemente *paso de integración.*

1.2. Métodos Multipasos Lineales

Sea y_n una aproximación para la solución teórica en t_n, esto es, para $y(t_n)$, y sea $f_n \equiv f(t_n, y_n)$. Si el método computacional para determinar las sucesión de puntos $\{y_n\}$ toma la forma de una relación *lineal* entre y_{n+j}, f_{n+j}, $j = 0, 1, \ldots, k$, es llamado un *método multipaso lineal de k-pasos,* o un **Método Multipaso Lineal (MML)** y lo podemos escribir como

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = h \sum_{j=0}^{k} \beta_j f_{n+j}, \tag{1.2.1}$$

donde $\alpha_k = 1$, $\alpha_0^2 + \beta_0^2 \neq 0$. El método es explícito si $\beta_k = 0$ e implícito si $\beta_k \neq 0$, ver Lambert [21].

1.2.1. Regiones de Estabilidad Absoluta de los MMLs

A continuación definimos la **REA** para el **MML** (1.2.1) tal como lo menciona Lambert [21]. Un procedimiento para encontrar el lugar geométrico de esta región se presenta en a sección 3.4.

Definición 1 *Un método multipaso lineal es absolutamente estable (relativamente estable) en una región $\mathcal{R}$ del plano complejo $\mathbb{C}$, si para toda $z = \lambda h \in \mathcal{R}$, todas las raíces del polinomio de estabilidad $\pi(r, z)$ asociado con el método (1.2.1), satisface*

$$| r_s | < 1, \quad s = 1, 2, \ldots, k$$
$$(| r_s | < | r_1 |, \quad s = 2, \ldots, k)$$

El polinomio de estabilidad asociado con el método (1.2.1), aplicado a la ecuación prueba $y' = \lambda y$, está dado por

$$\pi(r, z) = \rho(r) - z\sigma(r) = 0, \tag{1.2.2}$$

donde

$$\rho(r) = \sum_{j=0}^{k} \alpha_j r^j; \tag{1.2.3a}$$

$$\sigma(r) = \sum_{j=0}^{k} \beta_j r^j. \tag{1.2.3b}$$

En función de las ecuaciones (1.2.2), (1.2.3a) y (1.2.3b), la **REA** $\mathcal{R}$ se puede establecer como aquella parte del plano complejo $z = \lambda h$ tal que

$$\mathcal{R} = \{z \in \mathbb{C} \mid z = \lambda h = \frac{\rho(r)}{\sigma(r)}, \text{y } \pi(r, z) = 0, | r_s | < 1, s = 1, 2, \ldots, k\}. \tag{1.2.4}$$

1.3. Cotas de error de los MML

En los métodos multipaso un resultado clásico es el siguiente

$$|e_n| < [\delta + (t_n - a)h^p GY] \exp [LB(t_n - a)] \tag{1.3.1}$$

donde

$$e_n = y(t_n) - y_n,$$

es el error global, $h = (b - a)/n$, tal que

$$\delta = \max_{\mu=0,1,\ldots,k-1} |e_\mu| ,$$

es el máximo error en los valores de inicio,

$$G = \frac{1}{p!}\int_0^k \left| \sum_{j=0}^{k} [\alpha_j (j-s)_+^p - p\beta_j (j-s)_+^{p-1}] \right| ds,$$

$$Y = \max_{t\in[a,b]} \left| y^{(p+1)}(t) \right| ,$$

$$B = \sum_{j=0}^{k-1} |\beta_j|$$

y L es la contaste de Lipschitz asociada al problema de valor inicial.

La expresión dada por 1.3.1 asegura la convergencia del **MML** ya que $e_n \to 0$, cuando $n \to \infty$, de acuerdo con Lambert [21].

1.4. Métodos Runge Kutta

La filosofía de Runge, después desarrollada por Kutta y Heun, es lograr un método numérico de orden alto sacrificando la linealidad pero preservando sólo un paso a diferencia de los **MMLs** como lo refiere Lambert [21].

Retomando la notación convenida para y_n y f_n, el *método* ***Runge–Kutta (R–K)*** *de R etapas* explícito está definido por

$$\begin{aligned} y_{n+1} - y_n &= h\phi(t_n, y_n, h), \\ \phi(t, y, h) &= \sum_{r=1}^{R} c_r k_r, \\ k_1 &= f(t, y), \\ k_r &= f\left(t + ha_r, y + h\sum_{s=1}^{r-1} b_{rs} k_s \right), \quad r = 2, 3, \ldots, R, \\ a_r &= \sum_{s=1}^{r-1} b_{rs}, \quad r = 2, 3, \ldots, R. \end{aligned} \tag{1.4.1}$$

donde c_r, a_r y b_{rs} son constantes.

Notemos que un *método* ***R–K*** *de R etapas* implica R evaluaciones funcionales por etapa. Cada una de las funciones $k_r(t, y, h)$, $r = 2, 3, \ldots, R$, puede ser interpretada como una aproximación a la derivada $y'(t)$, y la función $\phi(t, y, h)$ como una media ponderada de estas aproximaciones, lo que significa que, en general, son métodos costosos para **TR**, en consecuencia en este trabajo se consideran solo métodos **R–K** de orden menor o igual que cuatro y explícitos.

Además, debemos satisfacer la siguiente condición

$$\sum_{r=1}^{R} c_r = 1$$

1.4.1. Regiones de Estabilidad Absoluta de los métodos R–K

Consideremos el método **R–K** de tres etapas ($R = 3$) y de orden 3, aplicado a la ecuación prueba linealizada $y' = \lambda y$, donde λ es un número real menor que 0, entonces

$$\begin{aligned} k_1 &= \lambda y \\ k_2 &= \lambda y(1 + a_2 h\lambda) \\ k_3 &= h\lambda(a_3 h\lambda + a_2 b_2 h^2\lambda^2), \end{aligned}$$

luego, como

$$\phi(t, y, h) = \sum_{r=1}^{R} c_r k_r,$$

entonces

$$\begin{aligned} y_{n+1} - y_n &= h\phi(t_n, y_n, h) \\ &= h\lambda \left[(c_1 + c_2 + c_3) + (c_2 a_2 + c_3 a_3)h\lambda + c_3 a_2 b_{32} h^2\lambda^2\right] y_n. \end{aligned}$$

Denotemos por $z = \lambda h$, entonces

$$\frac{y_{n+1}}{y_n} = 1 + (c_1 + c_2 + c_3)z + (c_2 a_2 + c_3 a_3)z^2 + c_3 a_2 b_{32} z^3. \tag{1.4.2}$$

Observemos que la ecuación (1.4.2) es una ecuación de diferencias, y su solución (ver Lambert [21]) es de la forma

$$y_n = d_1 r_1^n,$$

donde d_1 es una constante arbitraria. Entonces r_1 está dada por

$$r_1 = \pi(z) = 1 + (c_1 + c_2 + c_3)z + (c_2a_2 + c_3a_3)z^2 + c_3a_2b_{32}z^3. \tag{1.4.3}$$

El análisis anterior lo podemos extender para $\lambda = u + iw \in \mathbb{C}$ y $u < 0$. A partir de (1.4.3) damos la definición de estabilidad absoluta para un métdo **R–K** de tres etapas y orden tres, ver Lambert [21] .

Definición 2 *Un método Runge Kutta de tres etapas y orden tres es absolutamente estable en una región $\mathcal{R} \subset \mathbb{C}$ si $\pi(z)$ dado por (1.4.3) cumple que*

$$|\pi(z)| < 1$$

para toda $z \in \mathcal{R}$.

Luego, la **REA** $\mathcal{R}$ es el conjunto dado por

$$\mathcal{R} = \{z \in \mathbb{C} : |\pi(z)| < 1\}.$$

Generalizando los resultado anteriores para los métodos **R–K** de $R = p$ etapas y de orden p, tenemos que

$$r_1 = \pi(z) = 1 + z + \frac{1}{2!}z^2 + \cdots + \frac{1}{p!}z^p. \tag{1.4.4}$$

Luego, $\pi(z)$ es un polinomio de grado p en z.

Se sigue que, en los métodos de p etapas y orden p ($p \leq 4$) la **REA** está dada por el conjunto

$$\mathcal{R} = \{z \in \mathbb{C} : |\pi(z)| < 1\}, \tag{1.4.5}$$

donde $\pi(z)$ está dado por (1.4.4).

Entonces, para un p dado, $p = 1, 2, 3, 4$, todos los métodos **R–K** de p etapas y orden p tienen la misma **REA**, tal como lo refiere Lambert [21].

1.5. Cotas de error de los métodos R–K

De acuerdo con Lambert [21], definimos el error de truncamiento local en t_{n+1} del método general explícito de un paso (1.4.1) de la siguiente manera

$$T_{n+1} = y(t_{n+1}) - y(t_n) - h\phi(t_n, y(t_n), h),$$

donde $y(t)$ es la solución teórica del problema de valor inicial.

En el error de truncamiento local suponemos que no existen errores de truncamiento anteriores, es decir, $y_n = y(t_n)$. Lo cual implica que

$$T_{n+1} = y(t_{n+1}) - y_{n+1}.$$

El error de truncamiento global lo definimos como

$$e_{n+1} = y(t_{n+1}) - y_{n+1}$$

y su diferencia con el error de truncamiento local es que omitimos la suposición de que no existen errores de truncamiento anteriores, ver Lambert [21].

Podemos demostrar que el error de truncamiento local de los métodos **R–K** es

$$T_{n+1} = \psi(t_n, y(t_n))h^{p+1} + O(h^{p+2}),$$

donde $\psi(t, y)$ se llamará función de error principal, y $\psi(t_n, y(t_n))h^{p+1}$ es el error principal de truncamiento local.

Además si el error de truncamiento local T_{n+1} satisface

$$|T_{n+1}| \leq Kh^{p+1},$$

donde K es una constante, entonces el error global e_{n+1} en el intervalo (a, b) cumple la desigualdad

$$|e_n| \leq \frac{h^p K}{L}[\exp(L(t_n - a)) - 1], \tag{1.5.1}$$

donde $L > 0$ es la constante de Lipschitz de $f(x, y)$ con respecto a y. Para la demostración de este resultado le recomendamos consultar Henrici [17].

Una consecuencia de (1.5.1) es que $e_n \to 0$ cuando $n \to \infty$. Lo anterior en la práctica significa que el error global se encuentra acotado y no aumenta.

1.6. Análisis Estructural

Los modelos matemáticos de procesos dinámicos, en general están representados por un sistema de ecuaciones algebraicas no lineales acoplado con un conjunto de **EDOs** con **VIs**

$$\begin{aligned} \mathbf{g}(\mathbf{y},\mathbf{z}) &= \mathbf{0}, \\ \mathbf{y}'(t) &= \mathbf{f}(t,\mathbf{y},\mathbf{z}) \quad \text{y} \\ \mathbf{y}(a) &= \boldsymbol{\eta}, \end{aligned}$$

donde $\mathbf{g}$ representa el sistema de ecuaciones algebraicas lineales (o no lineales), $\mathbf{y}$ el vector de variables de estado, $\mathbf{z}$ el vector de variables algebraicas, y $\boldsymbol{\eta}$ el vector de **VIs**.

Puesto que es importante que los métodos numéricos para aproximar la solución de los sistemas de **EADs** sean capaces de manejar grandes problemas, es conveniente analizar primero su estructura para detectar defectos en ella. En este trabajo revisaremos que la evaluación de las variables que aparecen en las **EADs** no se efectúe en términos de otras que todavía no hayan sido calculadas, lo que se conoce como *estructura correcta del sistemas de **EADs***

Una estructura incorrecta en las ecuaciones algebraicas del sistema trae como consecuencia un flujo de información no consistente en el tiempo. Habrá ecuaciones que se encuentren evaluadas en un tiempo actual y otras en un tiempo anterior, en el mejor de los casos. La inconsistencia anterior origina que los algoritmos numéricos no se comporten adecuadamente por lo que no se tendrá un modelo eficiente de cómputo, ver Gear [14].

Una manera de ejemplificar cómo una estructura incorrecta influye en el comportamiento de un algoritmo es la siguiente: supongamos que emplea el algoritmo de Euler hacia adelante para resolver un sistema de **EADs**, lo que implica que todas las derivadas se encuentran evaluadas en el tiempo actual. Sin embargo, una estructura incorrecta (el orden incorrecto de ejecución de las ecuaciones) puede ocasionar que algunas derivadas estén evaluadas en el tiempo anterior. Por lo que, el esquema numérico realmente usado para estas ecuaciones es el algoritmo de Euler con retraso. Tales métodos son:

$$\begin{aligned} y_{n+1} &= y_n + h\dot{y}_n \quad \text{Euler hacia adelante} \\ y_{n+1} &= y_n + h\dot{y}_{n-1} \quad \text{Euler con retraso} \end{aligned}$$

donde y_n es la aproximación al tiempo t_n, h es el paso de integración, $\dot{y}_n$ es derivada evaluada en el tiempo t_n.

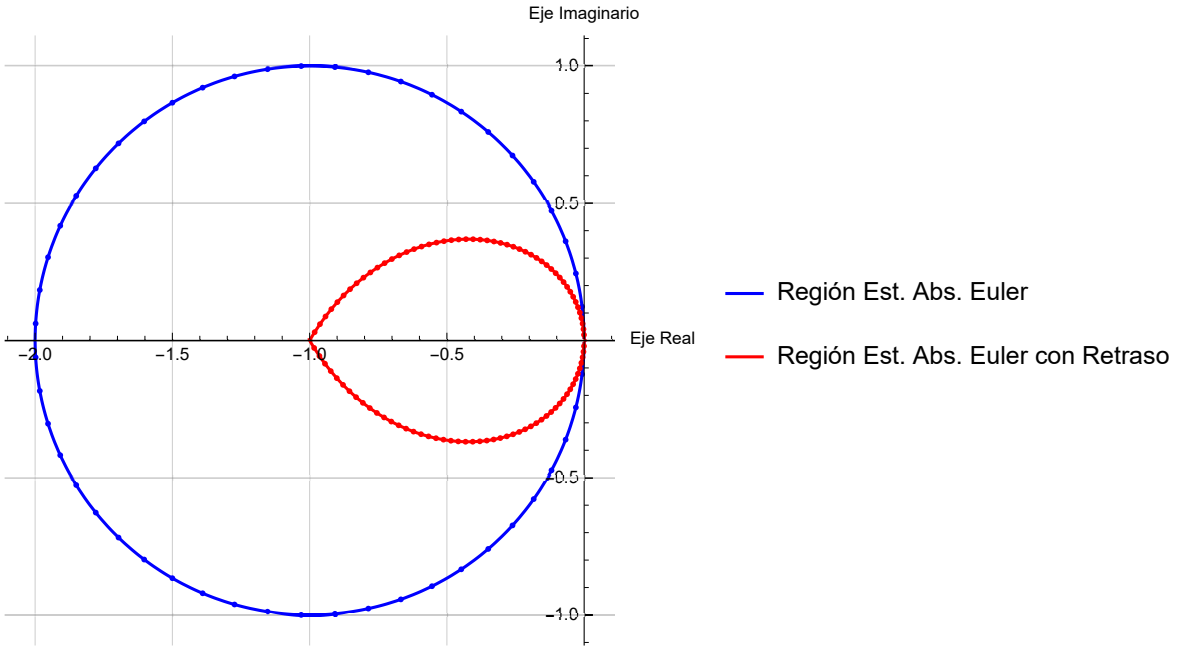

Figura 1.1: Gráfica de Euler hacia delante y Euler con retraso.

Existe una gran diferencia entre estos dos métodos numéricos, la región de estabilidad del primero es mucho mayor que la del segundo y contiene a ésta. En la práctica, los decrementos en las zonas de estabilidad obligan a reducir el paso de integración para lograr la convergencia (observar en la figura 1.1), lo cual causa que el esfuerzo computacional se incremente notablemente.

Puesto que los sistemas de ecuaciones representan un gran conjunto de información, es necesario usar un método compacto para representar el flujo de información entre éstos y lograr obtener el flujo de información correcto del sistema algebraico-diferencial. Para lograr lo anterior se emplean algoritmos de grafos dirigidos por lo que es conveniente relacionar el sistema algebraico-diferencial que representa al modelo como una matriz booleana y ésta a su vez en un grafo dirigido. La matriz booleana que se usa es llamada *matriz de incidencia* S, y se define como sigue:

1. Cada renglón de la matriz de incidencia corresponde a una ecuación del sistema, y cada columna corresponde a una variable del sistema;
2. Un elemento de la matriz, s_{ij}, es un valor booleano 1 ó 0 de acuerdo a la siguiente regla

$$s_{ij} = \begin{cases} 1, & \text{si la variable } j \text{ aparece en el renglón } i \\ 0, & \text{en otro caso} \end{cases}$$

La matriz S indica la ocurrencia de las variables dependientes en cada una de las ecuaciones del sistema. Si el sistema de ecuaciones es independiente, y el número de ecuaciones es igual al de las variables, entonces la matriz de incidencia es cuadrada.

La variable para la cual una ecuación debe ser resuelta es llamada variable de salida, y al conjunto de tales variables se le conoce como conjunto de salida.

En términos de la matriz de incidencia S, un conjunto de salida es un conjunto (no único) de elementos diferentes de cero en dicha matriz tal que uno y solo un elemento aparece en cada renglón y simultáneamente uno y solo un elemento aparece en cada columna.

La matriz de adyacencia A la podemos obtener a través de permutaciones de columnas de la matriz de incidencia S, para llevar a la diagonal a un conjunto de salida, de acuerdo con Steward [36].

A partir de la matriz de adyacencia A obtenemos el flujo de información correcto que, en forma resumida, consiste en encontrar una estructura correcta: dada la matriz de adyacencia A con r elementos diferentes de cero buscamos una matriz de permutación Q tal que

$$QAQ^T = \begin{bmatrix} A_{11} & & & & \\ A_{22} & A_{22} & & 0 & \\ A_{31} & & \ddots & & \\ \vdots & & & \ddots & \\ A_{n1} & & \cdots & & A_{nn} \end{bmatrix}$$

donde los bloques A_{ii} son cuadrados y no se pueden reducir a su forma simétrica triangular por bloques, tal como lo indica Cruz en [10].

1.7. Algoritmos de Ordenamiento

A continuación, damos una breve explicación del algoritmo que permite encontrar a la matriz Q a partir de la matriz A de adyacencia, de tamaño $n \times n$ con r elementos diferentes de cero.

Antes de explicar el algoritmo, definimos lo que es un grafo dirigido, de aquí en adelante grafo, es un conjunto de n nodos (o vértices) v etiquetados $1, 2, \ldots, n$ y un conjunto de aristas (v_i, v_j). Se dirá que la arista (v_i, v_j) deja el nodo i y entra al nodo j. En la terminología de la teoría de grafos, estos no tienen múltiples aristas: dos distintas aristas salientes de un mismo nodo no tienen el mismo punto final. Además, no tienen ciclos a sí mismos: aristas de la forma (v_i, v_i).

El primer paso del algoritmo es asociar a la matriz A con un grafo, donde cada arista (v_i, v_j) es una pareja ordenada de nodos correspondientes a un elemento a_{ij}

diferente de cero, fuera de la diagonal de la matriz.

También, necesitamos la definición de *camino*. Un *camino* del nodo v_1 al nodo v_k es una secuencia de aristas: (v_1, v_2), (v_2, v_3), ..., (v_{k-1}, v_k), denominado *ciclo* si $v_1 = v_k$. Un subgrafo de un grafo G, es un grafo que tiene todos sus vértices y aristas en G.

Se dice que un subgrafo está *fuertemente conectado* si existe un camino de u a v y un camino de v a u para cada pareja distinta de nodos u y v. Luego, un subgrafo es una *componente fuerte* si es fuertemente conectado y no puede ser extendido a otro subgrafo fuertemente conectado añadiendo nodos extras y asociando las aristas. Cada nodo puede pertenecer a una componente fuerte, la cual podría consistir de un solo nodo. Luego, las componentes fuertes definen una partición del grafo. Entonces, debe haber al menos una componente fuerte tal que no hay un camino desde cualquiera de sus nodos a cualquier nodo de otra componente fuerte. Llamemos a esta componente fuerte C_1. El resto de las componentes fuertes C_2, C_3, ..., C_k pueden ser elegidas análogamente de tal forma que no existe un camino desde cualquier nodo de una componente fuerte a un nodo de otra componente fuerte en una secuencia.

Si estas componentes fuertes son identificadas y se encuentra su ordenamiento, $C_1, \ldots, C_n$, y los nodos de C_1 son etiquetados antes de los de C_2 y así sucesivamente; entonces, la matriz asociada al grafo G es triangular inferior por bloques, con los bloques correspondientes a las componentes fuertes. Lo cual es equivalente a obtener una matriz Q, mediante permutaciones de filas y columnas, tal que, QAQ^T sea una matriz diagonal por bloques con lo que se obtiene la estructura correcta del sistema de ecuaciones.

Existen diversos algoritmos que realizan lo anterior y de acuerdo con un estudio realizado por Duff [11] se sabe que, Harary [16] por medio del uso de matrices booleanas presentó un algoritmo para obtener Q. Sin embargo, este algoritmo requiere al menos $O(n^3)$ operaciones lo que resulta bastante ineficiente en tiempo cómputo para sistemas grandes. Sterward [36] presentó un algoritmo que mediante operaciones "OR" selectivas y fue 10 o hasta 20 veces más rápido que el método de Harary. Posteriormente, Sargent y Westerberg [34] presentaron un método que es dos veces más rápido que el de Steward, este algoritmo requiere $O(n^2)$ operaciones para obtener Q. Sin embargo, Tarjan [38] propuso un algoritmo para encontrar las componentes fuertemente conectadas en un tiempo $O(n + e)$ donde n es el número de nodos y e el de aristas.

Debido a su menor complejidad, respecto a los otros, se tomó el algoritmo de Tarjan para obtener una estructura correcta en un sistema de EADs a gran escala. La idea principal del algoritmo de Tarjan se basa en las búsquedas a profundidad de grafos dirigidos [37].

1.8. Selección del paso y método de integración

Al trabajar con simuladores para entrenamiento o educación, debemos tomar en cuenta que los experimentos realizados se repiten de manera continua y además son ejecutados en **TR** o en un tiempo limitado. La complejidad del modelo que representa a la planta real radica en el gran número de **EADs** o que posee.

Es por eso que, la selección del tamaño de paso y método integrador adecuado para realizar una simulación es un problema frecuente al que se enfrentan los ingenieros encargados de realizar un simulador. En este trabajo se pretende resolver este problema mediante la **REA** del método numérico elegido, puesto que con el conocimiento de los valores característicos del sistema de ecuaciones diferenciales que representan al modelo, la **REA** nos proporciona información acerca del paso de integración.

Los sistemas de ecuaciones diferenciales con los que se trabaja son de la forma:

$$\mathbf{y}'(t) = \mathbf{f}(t, \mathbf{y}) \tag{1.8.1}$$

donde $\mathbf{y} \in \mathbb{R}^n$ es un vector de variables de estado.

Necesitamos conocer el conjunto de valores característicos $\{\lambda_1, \ldots, \lambda_n\}$ en diferentes puntos de operación, asociado a las matrices jacobianas de $\mathbf{f}(t, \mathbf{y})$ denotadas por $J_i = \partial \mathbf{f}(t_i, y_i)/\partial \mathbf{y}$ $(i = 1, \ldots, m)$, con el fin de determinar el método integrador numérico y el paso de integración adecuados para resolver numéricamente el sistema (1.8.1), y que éstos que satisfagan las restricciones de la simulación.

Por medio de la teoría de las **REAs** presentada en las secciones 1.2.1 y 1.4.1, seleccionaremos el paso de integración h_j para cada uno de los p métodos numéricos $(j = 1, \ldots, p)$ que se apegue a las restricciones de eficiencia y precisión para resolver el sistema (1.8.1). De acuerdo a esta teoría es necesario obtener $J_i, i = 1, \ldots, m$ de (1.8.1), por medio de las series de Taylor en los diferentes puntos de operación $\{(t_i, y_i)\}$ para $i = 1, \ldots, m$ donde cada pareja (t_i, y_i) se encuentra en el dominio de definición de $\mathbf{f}(t, \mathbf{y})$.

Para cada matriz jacobiana aproximada J_i determinamos los valores característicos λ_k para $k = 1, \ldots, n$, y mediante ellos es posible establecer una cota superior $H_{i,j}$ $(i = 1, \ldots, m,\ j = 1, \ldots, p)$ para el paso de integración h_j, para cada uno de los j métodos numéricos candidatos.

Finalmente, para determinar el paso de integración h_j asociado al método numérico j $(1 \leq j \leq p)$ hallamos la cota superior de

$$0 < h_j < min\{H_{i,j}\}, \quad i = 1, \ldots, m. \tag{1.8.2}$$

La expresión (1.8.2) nos indica la cota superior para el valor máximo que puede tomar h_j. A partir de ésta se puede determinar el rango de variación para el paso h adecuado

al método j

$$0 < h \leq h_j, \quad j = 1, \ldots, p. \tag{1.8.3}$$

1.9. Limitadores

En la simulación de procesos frecuentemente se requiere modelar estados de la planta que tienen restricciones físicas. Por ejemplo, voltajes, temperaturas, aperturas y cierres de válvulas cuyas variaciones se encuentran dentro de un rango permitido. Esto implica que, en términos matemáticos, se necesita dar solución al siguiente problema de **VIs**

$$y' = f(t, y), \qquad y(t_0) = y_0, \tag{1.9.1}$$

que satisfaga las siguientes restricciones

$$m \leq y(t) \leq M, \quad m, M \in \mathbb{R}, \tag{1.9.2}$$

donde m y M son dos constantes arbitrarias,$\mathbf{f} : I \times D \to \mathbb{R}^n$, I es un conjunto abierto de $\mathbb{R}$, y D subconjunto de $\mathbb{R}^n$. Por simplicidad en la notación supondremos que $n = 1$, e identificamos a $\mathbf{f}(t, \mathbf{y})$ con $f(y)$.

También, suponemos que el problema de **VIs** (1.9.1) satisface las condiciones del teorema de existencia y unicidad para ecuaciones diferenciales ordinarias, ver Henrici [17], que garantizan la existencia de una única solución continua.

En la sección 3.3.2 mostraremos como en la práctica se da solución a este problema, y daremos una solución con mejores resultados que la pragmática, ésta última podemos encontrar en el trabajo de Gallardo [13].

Capítulo 2

Trabajo Relacionado

A continuación, se presenta una recopilación de trabajos relacionados con el propósito de este escrito. La primera sección de este capítulo muestra la solución de diversos autores al problema de determinar el método numérico y el paso de integración adecuado dentro del contexto de simulación de procesos. La segunda sección se enfoca en comentar escritos relacionados con el análisis estructural de sistemas de ecuaciones que cuentan con un gran número de ecuaciones.

2.1. Determinación del método numérico y paso de integración

Comenzamos esta sección describiendo el trabajo de Gear [14], quien desde 1977 ya reportaba conflictos en la simulación de procesos con restricciones de tiempo, el autor concluía que debido a la longitud del código computacional y a la necesidad de mantener la estructura del sistema en el programa, los pasos de integración no pueden ser muy pequeños, por lo que los métodos de orden y paso variable no deben utilizarse para simulaciones en **TR**. Se recomienda entonces, en general, emplear métodos explícitos de orden bajo y de una sola evaluación de las derivadas.

Posteriormente, con el incremento de los paquetes que proporcionan la solución numérica de sistemas de **EDOs** con **VIs** se volvió cada vez más difícil a los usuarios

seleccionar el método más adecuado para resolver su problema. Por lo que, diversos autores comenzaron a sugerir *software* para la solución de tal problemática.

Fuera del contexto de simulación en **TR** encontramos a los siguientes autores que se centraron en analizar cómo seleccionar las estrategias numéricas para aproximar la solución de sistemas de **EDOs** en función de sus características más relevantes. Addison [1], en 1991 propone un *software* basado en *un árbol de decisión* para la solución numérica de **EDOs** con **VIs**. La idea se centra en identificar las características relevantes del problema a resolver, para obtener un conjunto de códigos que se enfoquen en la solución numérica y cuando sea posible recomendar al usuario códigos de librerías donde el software ha sido extensamente probado y tiene mantenimiento continuo. Sin embargo, el inconveniente de esta propuesta es que no hace énfasis en el tamaño del paso de integración ni en la estructura del sistema de **EDOs** del problema, por lo que la mayoría de las veces no se cuenta con soluciones eficientes.

En 1992 Lucks [22] propone *una selección automatizada* de *software* matemático para la solución numérica de **EDOs** con **VIs**, la cual es una mejora a la propuesta del *árbol de decisión*, puesto que no sólo se basa en las características cualitativas, sino que también en las cuantitativas. Aunque esta propuesta es marcadamente mejor que la Addison, tampoco hace un análisis para determinar el paso de integración más apropiado del problema en cuestión ni en la estructura del sistema.

En 1993 Kamel [19] presenta un prototipo de *un sistema experto* llamado ODEXPERT para la selección apropiada de solucionadores para sistemas de **EDOs** con **VIs**. Una cualidad importante del programa es que se incorporaron varias pruebas automatizadas para investigar propiedades críticas del problema de entrada, tales como linealidad, rigidez, estructura y jacobianos. El autor menciona que para utilizar las propuestas anteriores a la suya, el usuario debía contar con conocimientos en análisis numérico, lo cual no sucedía la mayoría de las veces. Aunque este prototipo sí toma en cuenta más propiedades del sistema de ecuaciones, tampoco se enfoca en el orden de ejecución de las ecuaciones ni en análisis para determinar el tamaño del paso de integración.

Ahora bien, dentro del contexto de **TR** encontramos los siguientes trabajos.

En Julio de 1995, Rodríguez-Gómez [30] realiza un análisis de técnicas numéricas para **TR** en donde hace las siguientes observaciones. Las simulaciones en **TR** se analizan considerando el lazo que conforman el operador y la máquina, no como un problema auto-contenido, además se dispone un recurso limitado de tiempo para realizar los cálculos numéricos, la interpolación no se puede usar libremente es necesario evaluar cuándo aplicarla sin afectar la operación en tiempo real. Las discontinuidades originan dificultades a los algoritmos de integración numérica de las **EDOs**. Los retardos en la información colapsan la zona de estabilidad numérica, lo que puede originar inestabilidad numérica. No es posible el uso de métodos de integración de paso y orden variable con control de error en simulación en **TR** debido a su alto costo computacional.

Después, en septiembre de 1995, Rodríguez-Gómez [31] muestra una implementación de las **Backward Differentiation Formulas (BDF)** de orden uno a paso fijo para simulaciones de procesos en tiempo real, en el contexto de simulaciones de plantas termoeléctricas. Es importante resaltar que las **BDF** requieren de la matriz jacobiana, por lo que se tiene que aproximar numéricamente o proporcionar la forma analítica. La implementación mostrada por Rodríguez-Gómez tiene un bajo costo computacional ya que requiere de pocas aproximaciones de la matriz jacobiana y es capaz de manejar discontinuidades que afectan a la función de excitación del sistema. Una limitación es que el usuario tiene la responsabilidad de dar las **EADs** en forma residual y también los valores iniciales de la parte diferencial y algebraica.

Cuando se trabaja con problemas complejos y de gran escala, por lo general se tiene que hacer frente a sistemas muy diferentes incluyendo escalas de tiempo. En 1998, Rukgauer [33] introduce una solución para este tipo de simulaciones, señala que el sistema dinámico modular puede ser simulado por una integración multitasa para cubrir las necesidades de eficiencia en el tiempo de integración de la simulación en cuestión, explotando la modularidad del sistema. El inconveniente de este trabajo es que se limita al contexto de la mecatrónica.

Seleccionar la solución numérica correcta o el paquete numérico más adecuado para un problema de simulación en particular se ha vuelto cada vez más difícil para los usuarios que no cuentan con una extensa formación matemática y mucho menos con conocimientos profundos en el análisis numérico.

La resolución de problemas en la ciencia y la ingeniería a menudo requiere solución numérica de grandes sistemas de ecuaciones. Es una práctica común de hoy en día usar *software* "fuera de plataforma" para resolver problemas numéricos. Esta tendencia se ve apoyada por la disponibilidad de grandes librerías de subrutinas matemáticas de propósitos generales tales como la *Collected Algorithms* publicado por la revista **Association for Computing Machinery (ACM)**, de acuerdo con Bonus [6].

En el 2006, Bunus [6] comenta que el uso de rutinas numéricas fácilmente disponibles plantea varias dificultades a un usuario normal:

- A pesar de que estos códigos numéricos estándar suelen ofrecer muy alta calidad si se utilizan solos, el uso integrado con otros componentes de software rara vez conduce a soluciones óptimas. Muchos paquetes numéricos están anidados uno dentro del otro. Por lo tanto, el usuario de un entorno de modelado de ingeniería tiene que jugar con detalles numéricos en los que ni está interesado y rara vez es un experto.
- A fin de garantizar la robustez, los parámetros de ajuste de los solucionadores numéricos individuales deben estar disponibles para el usuario. Esto conduce a menudo a una interfaz de usuario torpe y apenas intuitiva interfiriendo seriamente con el problema de la aplicación del cual el usuario está realmente interesado.

- La eficiencia computacional de la simulación numérica está fuertemente influenciada por una correcta elección del método numérico. Esta opción sólo puede ser realizada por un usuario profesional y la selección de un paquete matemático adecuado requiere una profunda comprensión del problema que se necesita resolver y una familiaridad con los paquetes matemáticos disponibles.

Otra práctica común que Bunus [6] refiere es la utilización de entornos de modelado y simulación que ayudan a los ingenieros a crear modelos a través de una interfaz gráfica de usuario o escribiendo código fuente de modelado personalizado utilizando un lenguaje de modelado matemático de alto nivel, como MATLAB/Simulink. Este enfoque ofrece una gran flexibilidad a nivel de entrada del modelo, pero no es lo suficientemente flexible a nivel de simulación. El usuario está obligado a especificar de antemano la elección de un programa de solución numérica que se debe utilizar en combinación con un cierto modelo de simulación. En algunos casos, esto significa que los usuarios necesitan tener conocimientos avanzados de análisis numérico y un buen conocimiento de las propiedades estructurales y del comportamiento del modelo a simular.

Las características deseables que un marco de decisión para la selección de las estrategias numéricas debe poseer, sin ningún orden de importancia en particular son:

- Aplicabilidad: detecta y resuelve una amplia gama de **EDOs**, **EADs**, **Ecuaciones Diferenciales Parciales (EDPs)**, lineales y sistemas de ecuaciones no lineales de forma automática o semi-automática (la intervención del usuario se reduce al mínimo).
- Eficiencia: Los requisitos de tiempo de ejecución y memoria debe estar cerca de un mínimo.
- Extensibilidad: nuevos algoritmos y nuevos paquetes numéricos se pueden adaptar fácilmente a los ya existentes.
- Generalidad: Es independiente del lenguaje de entrada del modelado y genera una plataforma independiente de código fuente.
- La capacidad de depuración: En caso de fallo debe producir mensajes de error significativos que ayudan a corregir la falla. También debe identificar estáticamente problemas mal especificados y ofrecer alternativas de depuración.

El desarrollo de un programa final eficiente para entornos de simulación con las características antes mencionadas que se adapte a diversos paquetes numéricos existentes (a menudo escritos en varios idiomas) es una tarea difícil y propensa a errores. Según Bunus [6] el modelado y la generación automática de código es la forma más prometedora para abordar este problema. La **Model Driven Architecture (MDA)** es una iniciativa de la **Object Management Group (OMG)** para definir un enfoque

para el desarrollo de software basado en el modelado y mapeo automático de modelos para las implementaciones.

Bonus en [6] también plantea depender de la ingeniería basada en modelos, y meta modelado para automatizar el proceso de selección de solución numérica para un sistema de ecuaciones dado o para un modelo de simulación basado en ecuaciones. También propone una **MDA** combinada con un marco de decisión para seleccionar automáticamente un método numérico para un determinado conjunto de ecuaciones del sistema. Proporciona un prototipo formal basado en lenguajes de dominio específicos para la explicación de los aspectos estructurales y de comportamiento del proceso de resolución de ecuaciones numéricas denominado generador prototipo de soluciones del modelo.

Además, en su artículo, Bonus [6], presenta *ModSimPack*, un paquete de Modelado y Simulación para la selección automática de solucionadores numéricos para la simulación de un problema en particular. *ModSimPack* toma como entrada un modelo de simulación expresado en la declarativa de modelado y un lenguaje de simulación como *Modelica*, ver Fritzon y Engelson [12]. El sistema es capaz de detectar los tipos de las ecuaciones, realizar manipulaciones simbólicas en ellos y decidir qué (programa de) solución numérica es adecuada para resolver el problema. *ModSimPack* proporciona un mecanismo de retroalimentación avanzado si inconsistencias y singularidades estructurales están presentes en el modelo, ver Bonus [7].

La salida es un modelo de dominio específico del proceso de resolución de ecuaciones. Un intérprete recorre el proceso de la solución generada del modelo en cuestión y emite código de procedimiento indispensable que puede ser compilado y ejecutado por un entorno de simulación.

ModSimPack pretende diseñar una metodología para el diagnóstico, la simplificación y la depuración de los modelos de simulación expresados por ecuaciones matemáticas mediante la presentación del método numérico y la selección del solucionador como un proceso de modelado. En la ingeniería de software tradicional y análisis matemático esta tarea rara vez es vista como un proceso de modelado.

El objetivo a largo plazo de la investigación de Bonus [6] es proporcionar un marco de decisiones más completo que implicaría la ampliación de *Solver Knowledge Base* para incluir más solucionadores numéricos disponibles en los repositorios de software matemático.

Sin embargo, presenta las siguientes limitaciones:

ModSimPack utiliza *Solver Knowledge Base*, pero sólo incluye un número limitado de solucionadores numéricos. Esto es debido a que el sistema estaba pensado originalmente como un motor simbólico y numérico para un entorno de simulación basada en *Modelica* donde un número limitado de solucionadores de propósito

general para **EADs** son suficientes para proporcionar la funcionalidad requerida.

ModSimPack está limitado en cuanto al tamaño del sistema de **EADs**, porque, por ejemplo, no es capaz de simular el modelo presentado en la sección 4.4 a pesar de que sólo cuenta con veintisiete variables de estado.

Durante las últimas décadas se han estado desarrollando e investigando una amplia variedad de métodos numéricos para resolver **EDOs** y **EADs**. Con frecuencia, estos métodos están disponibles libremente en diversos lenguajes de programación y con diferentes interfaces. Anderson [2] advierte que acceder a ellos mediante una interfaz unificada es una necesidad no sólo de la comunidad de investigación y con fines educativos, sino también para que estén disponibles en contextos industriales.

Un modelo industrial de un sistema dinámico no siempre es sólo un conjunto de ecuaciones diferenciales. Actualmente los modelos pueden contener controladores discretos, impactos o fricciones resultantes en discontinuidades que necesitan ser manejadas por un programa de solución moderna de una manera correcta y eficiente. Además, los modelos pueden producir una enorme cantidad de datos que afectan la eficiencia computacional del software de simulación.

En este contexto, en el 2015 Anderson [2] presenta a *Assimulo*. El cual es una interfaz unificada a alto nivel para resolver **EDOs** y diseñada para resolver las necesidades en la investigación y la educación junto con los requerimientos para la resolución de modelos industriales con discontinuidades y el manejo de datos. Éste combina solucionadores clásicos y modernos originales, independientemente de su lenguaje de programación con una interfaz de Python / Cython bien estructurada. Lo que permite controlar fácilmente el ajuste de parámetros y la manipulación de discontinuidades para una amplia gama de clases de problemas.

Assimulo está libremente disponible y su plan a futuro es incluir una variedad creciente de códigos originales, que estarán disponibles a través del marco de trabajo presentado.

Aunque *Assimulo* tiene una interfaz amigable y fácil de trabajar presta poca atención a la determinación del paso de integración para lograr una simulación que cumpla con los requerimientos de **TR**.

2.2. Análisis estructural en la simulación de procesos

En esta sección recopilamos de diversos autores los beneficios de utilizar el análisis estructural en la simulación de procesos.

En 1995 Mattson [24] presenta un enfoque orientado a objetos en donde trabaja con la estructura del sistema de ecuaciones en modelos de tiempo continuo. Argumenta que como se trabaja en tiempo real, el enfoque simultáneo es mejor para simulaciones con modelos en tiempo continuo, mientras que para simulaciones de eventos discretos es mejor el enfoque modular. El autor trabaja el análisis estructural por medio de la representación matricial del sistema de **EADs** al que nombra *Jacobiano,* donde cada entrada es cero si no aparece la variable j en la ecuación i, y uno en otro caso. Después, mediante un algoritmo se lleva a cabo la transformación del Jacobiano en una matriz triangular inferior por bloques, luego ésta es la que brinda el orden de ejecución secuencial de las ecuaciones del modelo. El principal beneficio del análisis estructural mencionado por Matsson, es la reducción del tamaño del problema a resolver numéricamente a un décimo del problema original.

También en 1995, Rodríguez-Gómez [30] nos dice que es necesario contar con la estructura correcta del sistema de **EADs** cuando realizamos simulaciones en **TR**.

En 1998 Carpazano [8] presenta técnicas que posibilitan la manipulación simbólica de sistemas de **EADs** no lineales y la simplificación del modelo para lograr una simulación eficiente de sistemas continuos a gran escala en un ambiente de modelación orientado a objetos. Al realizar su trabajo el autor considera que los sistemas son matemáticamente correctos (en el caso general un modelo es correcto si es sintáctico y semánticamente correcto de acuerdo con la reglas del lenguaje de modelado, y si representa un problema completo y consistente desde el punto de vista matemático) y de índice 1 (el mínimo número de veces que todo o parte del sistema de **EADs** debe diferenciarse con respecto al tiempo para transformarlo en la forma explícita de **EDOs** es definido índice del sistema, ver Petzold [4]). También, Carpazano [8] introduce un algoritmo eficiente de sustitución para eliminar ecuaciones obvias (escalares y matriciales), realiza un análisis estructural de los datos y utiliza el algoritmo de Tarjan [38] para obtener una matriz triangular inferior por bloques y propone un algoritmo de rompimiento para las variables algebraicas (en particular se quiere dividir el sistema de **EADs** dado en dos subconjuntos, el de asignaciones que será lo más grande posible y el ecuaciones implícitas). Ofrece un *software* en donde realiza todo lo anterior y aplica las técnicas de manipulación de propuestas se han utilizado para reducir el esfuerzo computacional del proceso de simulación.

Por su parte, en 1998 Rukgauer [33] muestra una estrategia para la detección y

tratamiento de ciclos debidos a la estructura modular. Tal estrategia es implementada en el programa de simulación NEWMOS. Observemos que este autor hace lo mismo que lo señalado por Rodríguez-Gómez en [31], utiliza una estructura modular e integración multitasa, pero en el contexto de mecatrónica.

En el 2003 Cruz-Tavira y Rodríguez-Gómez [9] presenta los beneficios de mantener la estructura correcta en las ecuaciones que conforman un modelo matemático, sobre todo en sistemas de gran escala. El desempeño de un modelo con estructura correcta se ve reflejado en la eficiencia que logra este al ejecutarse en un sistema de cómputo, logrando un mejor uso de espacio y reducción del tiempo computacional. Además, el autor presenta una prueba de estructural original, la cual verifica la estructura de las ecuaciones de un modelo sin necesidad de crear una matriz de incidencia. De acuerdo con Cruz-Tavira yRodríguez-Gómez [9] una estructura incorrecta en el sistema de **EADs** eleva el costo computacional ya que se trabaja con un sistema de mayor dimensión, en el caso de sistemas dinámicos se incrementa el número de llamadas por el método numérico al modelo. El desarrollo de un modelo grande consume bastante tiempo, por ejemplo, la construcción de un simulador para entrenamiento de alcance total de una planta de potencia se construye en alrededor de tres años. El detectar errores, como una estructura incorrecta, al final del proyecto podría implicar que no se cumplan con algunos de los requerimientos funcionales estipulados, por ejemplo, la ejecución en **TR**. También, remarca que en sistemas que requieren **TR** es aún más crítica la necesidad de ejecutar un modelo matemático con eficiencia, de no ser así, podría no cumplirse las metas establecidas para un modelo, ver Cruz-Tavira y Rodríguez-Gómez [10].

En 2006, Bunus menciona en su artículo [7] que el uso de herramientas basadas en el análisis gráfico-estructural es de gran interés tanto en la visualización de propiedades de los sistemas de ecuaciones como en el seguimiento y la realización de manipulaciones simbólicas de variables y ecuaciones al modelar con lenguajes basados en ecuaciones. El autor muestra cómo las técnicas existentes de descomposición gráfica-teórica pueden ser adaptadas e integrados en herramientas de depuración integradas en entornos de simulación que emplean dichos lenguajes. También, trata de automatizar su depurador tanto como le es posible y mantener la interacción del usuario a un mínimo. Para lograr su objetivo, propone un marco automatizado de depuración. Además, afirma que las técnicas desarrolladas y propuestas en su artículo son adecuadas para una amplia gama de lenguajes basados en ecuaciones y no solo para el lenguaje *Modelica*. Bunus [7] indica que tales técnicas se pueden adaptar fácilmente a las características específicas de un entorno de simulación particular. Esta afirmación la basa en la estrecha integración de las técnicas de depuración y el desarrollo del proceso de compilación, porque la mayoría de los compiladores existentes para lenguajes basados en ecuaciones comparten los mismos principios.

En cuanto a la forma de programar los **EDOs** encontramos al siguiente autor.

En su trabajo Kecskemethy [20] no hace referencia al análisis de los métodos numéricos y tampoco al paso de integración de los mismos, pero si hace una observación

importante, dividir al sistema en subsistemas para resolverlos de forma independiente, el autor propone utilizar la programación orientada a objetos y la geometría diferencial para obtener una biblioteca de computadora que puede ser utilizada como un sistema de componente básico para realizar programas de simulación de dinámica de vehículos.

Con respecto a las pruebas y validación de los modelos simulados, Murray-Smith [27] señalaba que las herramientas de simulación ofrecían buenos servicios para la implementación eficiente de los modelos de simulación y para la experimentación interactiva de los mismos. Sin embargo, en la mayoría de los casos, estos ambientes carecían de características importantes que pudieran ayudar al usuario en cuestiones más generales que surgen en el proceso de desarrollo del modelo, tales como pruebas y validación del modelo.

En conclusión los problemas que se presentan en el contexto de simulación en tiempo real son los siguientes:

- La determinación de los métodos numéricos más adecuados para **EDOs** con **VIs**.
- El tamaño del paso de integración .
- Las pruebas y la validación del modelo del sistema de **EDOs**.
- La interpolación no se puede usar libremente es necesario evaluar cuándo se aplica sin afectar la operación en **TR**.
- Las discontinuidades originan dificultades a los algoritmos de integración numérica de **EDOs**.
- Los retardos en la información colapsan la zona de estabilidad numérica, lo que puede originar inestabilidad numérica.
- No es posible el uso de métodos de integración de paso y orden variable con control de error en simulación en **TR** debido a su alto costo computacional.
- Cuando se trabaja con problemas complejos y de gran escala por lo general se tiene que hacer frente a sistemas muy diferentes incluyendo escalas de tiempo.
- Los modelos pueden contener controladores discretos, impactos o fricciones resultantes en discontinuidades que necesitan ser manejadas por un programa de solución moderna de una manera correcta y eficiente.
- Los modelos pueden producir una enorme cantidad de datos que afectan la eficiencia computacional del *software* de simulación.
- La longitud del código computacional.
- La necesidad de mantener la estructura del sistema en el programa y el orden de ejecución de las ecuaciones del modelo que representa la planta real.

- Una estructura incorrecta en el sistema de **EADs** eleva el costo computacional ya que se trabaja con un sistema de mayor dimensión, en el caso de sistemas dinámicos se incrementa el número de llamadas por el método numérico al modelo.

Capítulo 3

Planteamiento del problema y metodología propuesta

En este capítulo, primero describimos la importancia del problema, después presentamos la metodología propuesta dividida en secciones. Además, proporcionamos ejemplos académicos acerca del análisis estructural y los limitadores. Por último, exhibimos un procedimiento para hallar las **REA** de los **MML**.

3.1. Importancia del problema dentro de la simulación

Cuando trabajamos con simuladores de procesos para educación (entrenamiento) o para análisis y diseño de procesos es importante cumplir con los requisitos de tiempo de ejecución de los modelos (por ejemplo que se ejecuten en **TR**) y brindar resultados que satisfagan el comportamiento del proceso modelado. Además, es necesario que se puedan realizar muchas repeticiones de la misma simulación y tener la opción de cambiar ciertos parámetros. Sin embargo, se presentan diversos problemas, que surgen desde la formulación del modelo de la planta física hasta su implementación computacional (la codificación del modelo en un lenguaje de alto nivel). Tal como vemos en la figura 3.1, el ingeniero de procesos es quien realiza las hipótesis y simplificaciones para construir el modelo, el cual debe ser validado y verificado, es decir los resultados se deben

corresponder con la realidad. Este proceso es iterativo y no necesariamente secuencial y en muchas ocasiones puede llevar un tiempo considerable el detectar las fuentes de los errores cuando estas se presentan.

Murray-Smith [27] proporciona una distinción entre "verificación" y "validación", la primera tiene que ver con la palabra "interno" y la segunda con "externo", y nos da las siguientes definiciones.

Verificación interna se define como el proceso de probar que una simulación computarizada es consistente con el modelo subyacente a un grado específico de precisión, se debe revisar la consistencia interna del programa de simulación con el modelo matemático en el que se basa y la validez algorítmica del programa de manera que todos los algoritmos numéricos y rutinas de software asociados muestren ser apropiados y proporcionen soluciones que tengan una precisión numérica especificada.

Por otro lado, la validación externa conlleva a demostrar que el modelo matemático o conceptual tiene una precisión aceptable en el rango de las condiciones importantes para su aplicación. Se deben observar los siguientes aspectos: la validez teórica, en el sentido de que el modelo muestra una coherencia global con las teorías aceptadas o se basa en un fundamento teórico satisfactorio, y la validez empírica, con la convención de que se tiene un comportamiento apropiado entre el modelo y el sistema real representado por el mismo.

En los entornos de desarrollo de simuladores, en algunas ocasiones únicamente se realizan pruebas de verificación al modelo sin realizar las pruebas de validación y puede suceder que los resultados obtenidos por medio de las simulaciones correspondan al comportamiento real del proceso dentro de un margen de error. Sin embargo, como podremos ver en los experimentos que presentamos en el capítulo 4, estos modelos no se pueden ejecutar en **TR** o reducir sus tiempos de ejecución dentro de los márgenes requeridos, pueden dar información no correcta respecto a sus constantes de tiempo, que podría llevar a la conclusión que el proceso modelado no es, por ejemplo, completamente controlable, cuando la planta real si lo es. Es por ello que en este trabajo proponemos realizar las dos pruebas: la verificación y la validación al modelo.

A continuación damos una lista que describe fuentes de errores que podemos encontrar en el proceso de modelado, programación y simulación.

Primero, durante el modelado pueden presentarse las siguientes fuentes de error:

Hipótesis y simplificaciones si estas no fueron las adecuadas, el modelo no representa apropiadamente al proceso físico.

La incertudimbre en los datos del problema. En muchas ocasiones, por ejemplo, los datos con que se inician los cálculos contienen un cierto error debido a que los hemos obtenido obtenido mediante la medida experimental de una determinada

magnitud física. La mayor parte de tales equivocaciones las podemos atribuir a errores humanos.

Después, las fuentes de error que podemos encontrar durante la programación son:

La gran cantidad de EADs disminuyen la eficiencia computacional y pueden ocasionar que su implementación computacional sea errónea, porque el orden de ejecución podría no ser el correcto. En consecuencia, es importante trabajar con la estructura correcta del sistema de ecuaciones.

La discretización del modelo (o error de truncamiento) tienen su origen en tomar únicamente un número finito de términos de la expresión en series de Taylor de la fórmula matemática.

Redondeo. Tales errores tienen su origen en los cálculos numéricos cuya representación en la computadora está restringida a un número finito de dígitos, estos errores dependen de la computadora.

Errores de codificación. Debido a que el modelo se debe programar en algún lenguaje de alto nivel pueden ocurrir este tipo de errores.

Problemas que se pueden presentar durante la simulación

Algoritmos inestables se trata de algoritmos que no se estabilizan, cualquier error en el procesamiento se magnifica conforme el cálculo procede.

Selección inadecuada de los métodos numéricos, en general, el usuario no hace un análisis de estabilidad numérica y en consecuencia no elige el mejor método para la simulación de procesos con restricciones de tiempo. Los métodos de integración se limitan a métodos explícitos de evaluación de una sola función.

La limitación de variables de estado por parte del usuario. Los modelos pueden tener restricciones físicas. Por ejemplo, voltajes, temperaturas, aperturas y cierres de válvulas cuyas variaciones se encuentran dentro de un rango permitido. Estas restricciones pueden afectar a alguna(s) variable(s) de estado, por lo que necesitan ser manejadas adecuadamente por el programa computacional para no afectar el resultado de los métodos de integración numérica y evitar que tales variables sean manejadas por el usuario final.

Problemas mal condicionados o sensibles son aquellos que al efectuar pequeños cambios en los parámetros de entrada originan grandes cambios en los parámetros de salida.

Nuestro trabajo se centra en resolver los problemas que ocasionan *la gran cantidad de* ***EADs***, *la elección inadecuada de los métodos numéricos* y *la limitación de variables de estado por parte del usuario.*

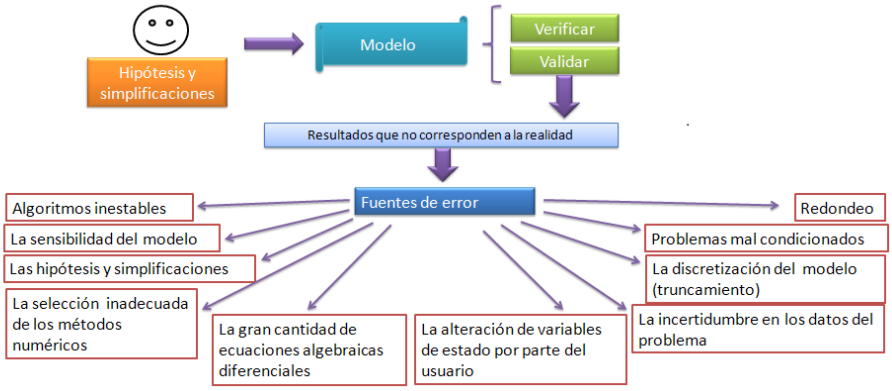

Figura 3.1: Diagrama que muestra el desarrollo de un modelo.

3.2. Metodología propuesta

A continuación explicamos detalladamente el desarrollo de la solución propuesta, dividiéndola en secciones.

3.2.1. Primera parte de la metodología

Tal como observamos en el diagrama de flujo de la figura 3.2, en la primera parte de nuestra metodología, recibimos el sistema de **EADs**, con el que realizamos la *prueba estructural mejorada*, ver la figura 3.3, que está basada en el trabajo de Cruz-Tavira y Rodríguez-Gómez [10]. Esta prueba consiste en recibir los parámetros de entrada del modelo mientras el tiempo lo mantenemos constante, y evaluamos una vez el modelo, guardando este primer resultado. Después, evaluamos el modelo por segunda ocasión y también guardamos este segundo resultado. Si la diferencia entre el primer y segundo resultado del modelo es igual a cero, significa que el modelo tiene una estructura correcta y procedemos al siguiente paso de la metodología. En caso contrario se presentan inconsistencias estructurales en el modelo por lo que es necesario elaborar la matriz de incidencia del sistema de **EADs** y por ejemplo, con el algoritmo de Tarjan [38] encontrar las inconsistencias en el sistema, y después regresar al inicio. Es importante resaltar que, para que esta prueba estructural tenga sentido, es necesario que, en la implementación computacional del modelo, sus parámetros de entrada sean pasados por referencia.

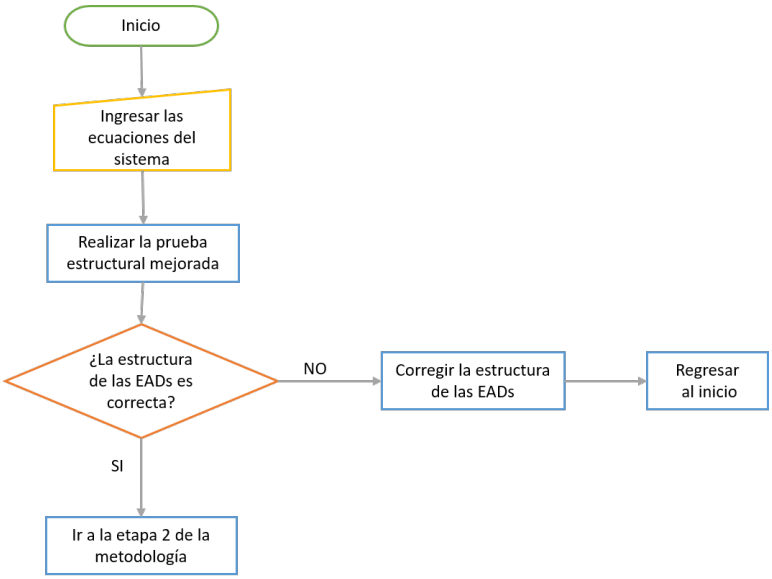

Figura 3.2: Primera parte de la metodología.

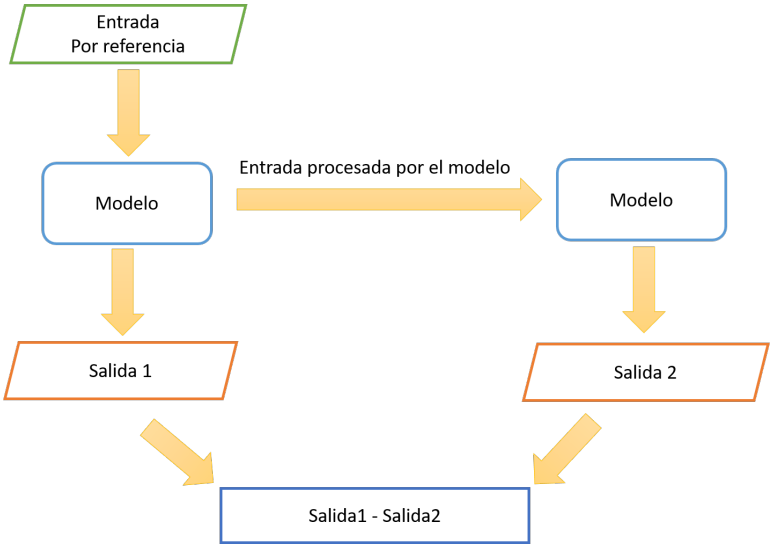

Figura 3.3: Esquema de la prueba estructural mejorada.

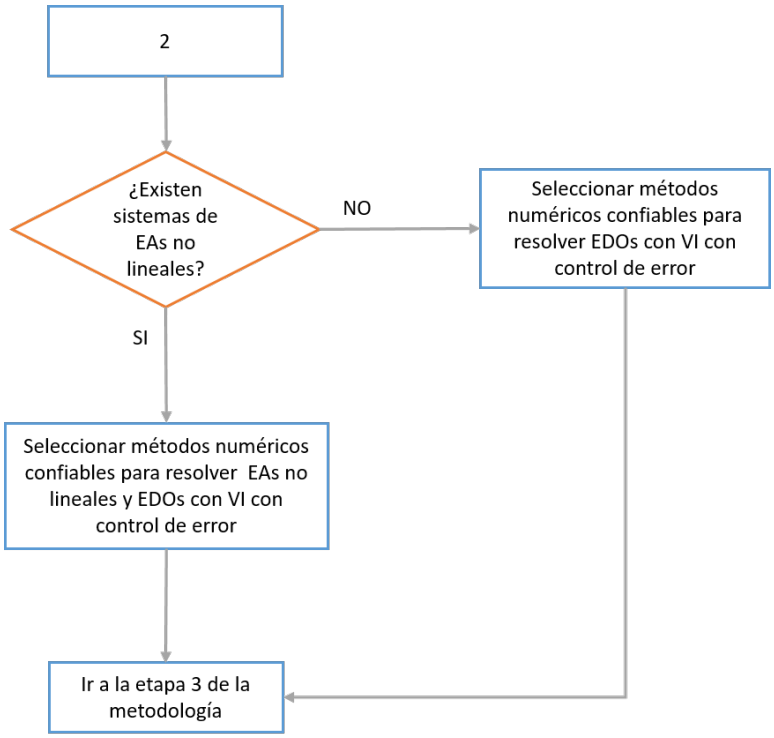

Figura 3.4: Segunda parte de la metodología.

3.2.2. Segunda parte de la metodología

Después, procedemos a comprobar si el modelo posee sistemas de **Ecuaciones Algebraicas (EAs)** no lineales por medio del análisis estructural del modelo, de ser el caso procederemos a seleccionar estrategias numéricas confiables para solucionarlos junto con las **EDOs**. Cuando lo anterior no suceda, buscaremos métodos numéricos que resuelvan solo **EDOs**. Esta parte de la metodología, la observamos en el diagrama de la figura 3.4.

Los métodos numéricos para resolver el o los sistemas de ecuaciones lineales o no lineales se elegirán, de preferencia, de repositorios confiables como Netlib [5] o los proporcionados por la revista **TOMS**, que poseen colecciones de software matemático validado. Por ejemplo, el presentado por Grosse en [15]. Además el método numérico seleccionado deberá tener control de error y ser de paso y orden variable.

3.2.3. Tercera parte de la metodología

Esta sección de la metodología la ilustramos con la figura 3.5, lo que prosigue es realizar una prueba para que el modelo mantenga un estado estacionario. El estado

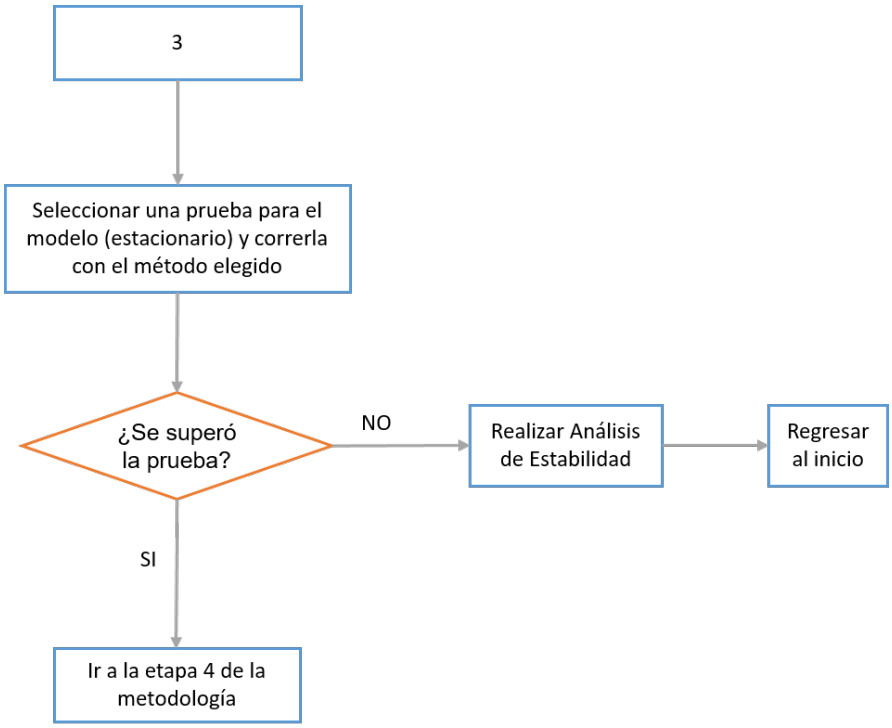

Figura 3.5: Tercera parte de la metodología.

estacionario lo elegimos a partir del conocimiento físico que se tiene del modelo en cuestión, puede ser necesario apoyarse de la experiencia de los ingenieros que conocen el proceso físico modelado. Si la prueba es superada, avanzamos al siguiente paso.

Cuando la prueba no es superada, es necesario que realicemos un análisis de estabilidad para encontrar las ecuaciones del modelo que dan origen al problema y regresamos al inicio de la metodología.

3.2.4. Cuarta parte de la metodología

El diagrama de la figura 3.6 nos muestra la penúltima parte de la metodología, la cual consiste en seleccionar un conjunto de transitorios y realizar simulaciones con el método numérico seccionado. Si los transitorios no fueron superados, entonces realizamos un análisis de estabilidad para encontrar las ecuaciones del modelo que originan el problema y regresamos al inicio, en caso contrario pasamos a la siguiente sección de la metodología.

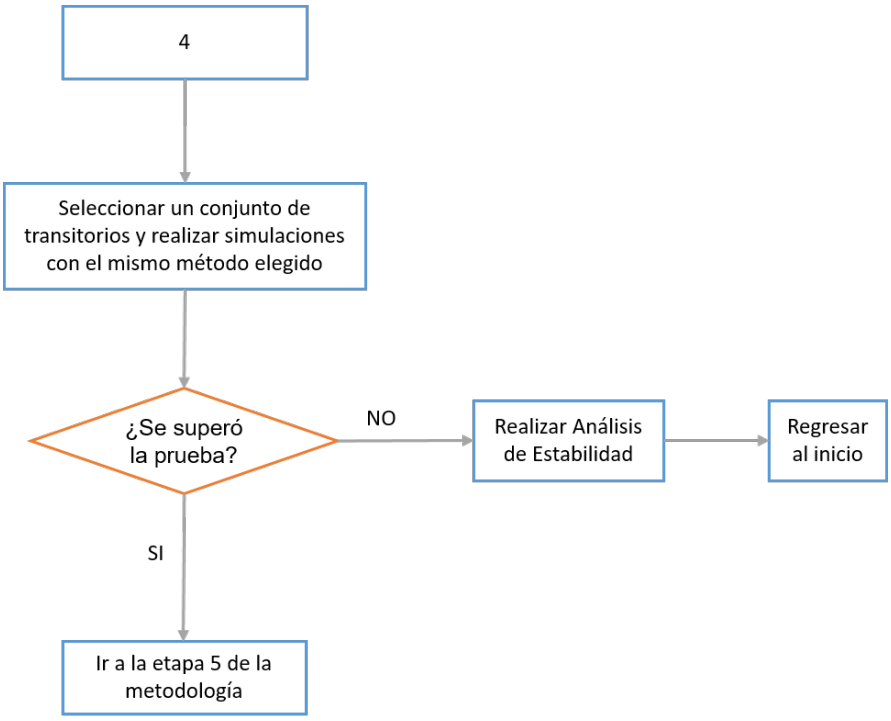

Figura 3.6: Cuarta parte de la metodología.

3.2.5. Quinta parte de la metodología

En la figura 3.7 ilustramos la última parte de la metodología. Aplicamos la metodología propuesta en la sección 1.8 para hallar el método numérico y paso de integración más adecuado para todo el modelo, luego construimos una tabla de costos y elegimos el método numérico y tamaño de paso que mejor convenga a nuestros propósitos.

Una vez que hemos llegado hasta este punto, es importante mencionar el método para la solución de ecuaciones no lineales se simplifica.

3.3. Ejemplos académicos

A continuación mostramos dos ejemplos académicos: el primero para aclarar la aplicación e ilustrar la utilidad del análisis estructural, y el segundo para mostrar las desventajas de la solución pragmática de los limitadores.

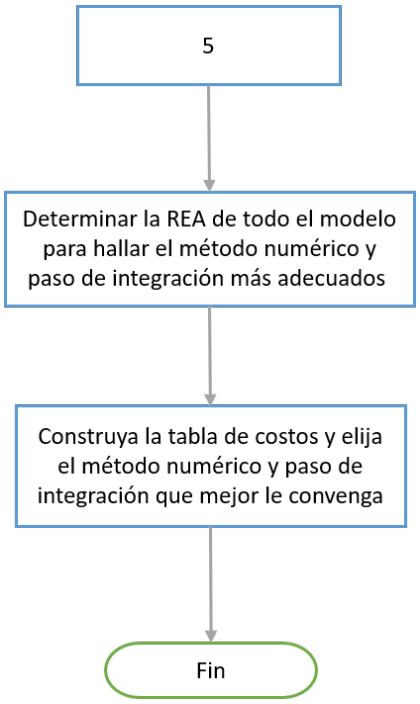

Figura 3.7: Quinta parte de la metodología.

3.3.1. Ejemplo de análisis estructural

En un sistema algebraico-diferencial que posee una gran cantidad de ecuaciones, es muy común encontrar una estructura incorrecta, porque ésta no puede ser determinada de forma simple manualmente.

A continuación, presentamos un ejemplo de análisis estructural mediante un modelo, tomado de Rodríguez-Gómez [32], que representa *tres reactores tipo tanque con control* ***Proporcional Integral (PI)*** *realimentado*, de aquí en adelante lo llamaremos brevemente: *modelo de los tres reactores*. En estos reactores se produce el producto B, y el reactante A se consume por una reacción de primer orden que se produce en el líquido. Se asume que las temperaturas y los volúmenes de los tres tanques pueden ser distintos, pero la temperatura y el volumen del líquido en cada tanque se considera constante. La representación matemática del modelo es la siguiente

$$Ca_m = 14 + K_c\left(y_4' + \frac{y_4}{\tau_i}\right) \tag{3.3.1}$$

$$Ca_0 = Ca_d + Ca_m \tag{3.3.2}$$

$$y_1' = \frac{Ca_0 - y_1}{\tau} - k_r y_1 \tag{3.3.3}$$

Parámetro	Valor	Descripción
Ca_d	21	concentración inicial del reactivo A
K_c	1	ganancia del controlador realimentado
k_r	0.5	constante de reacción
$SPCa_3$	3.5	concentración deseada en la salida del reactivo A
τ	2.0 mín	tiempo de residencia en el reactor
τ_i	0.6 mín	tiempo integral del controlador

Tabla 3.1: Descripción de parámetros.

$$y_2' = \frac{y_1 - y_2}{\tau} - k_r y_2 \tag{3.3.4}$$

$$y_3' = \frac{y_2 - y_3}{\tau} - k_r y_3 \tag{3.3.5}$$

$$y_4' = Spca_3 - y_3 \tag{3.3.6}$$

Donde y_1, y_2, y_3, y_4 son las variables de estado que representan la concentración del producto i, $i = 1, 2, 3, 4$. Ca_0 es la concentración de entrada del reactivo A, la cual está formada por la concentración de disturbio Ca_d y la manipulada por el controlador Ca_m. $SPCa_3$ es el punto de ajuste. K_c es la ganancia del controlador y τ_i es la constante integral del controlador.

Tenemos cuatro ecuaciones diferenciales ordinarias: (3.3.1), (3.3.3), (3.3.4), (3.3.5) y (3.3.6), y una algebraica: (3.3.2).

Las condiciones iniciales son: $y_1 = 0.4$, $y_2 = 0.2$, $y_3 = 1$, $y_4 = 0$.

En la tabla 3.1 listamos los valores de los parámetros.

En general, el usuario tiende a programar el modelo tal como se encuentra escrito en este trabajo: primero la ecuación (3.3.1) y por último la ecuación (3.3.6). Sin embargo, observemos que la ecuación (3.3.1) depende de la (3.3.6). Por lo que, debería de programarse primero la (3.3.6). En los casos en que el número de ecuaciones es muy grande, es recomendable seguir una metodología para obtener la estructura correcta del sistema. A continuación, se presenta un método (que podría utilizarse) ejemplificado con el *modelo de los tres reactores*, para finalmente corroborar que se llega al mismo resultado que hemos obtenido de manera pragmática.

Observemos que las ecuaciones (3.3.4) y (3.3.5), las podemos calcular en cualquier orden sin que influyan en las otras cuatro ecuaciones, por lo que no las consideramos para formar la matriz de incidencia.

La relación funcional de dependencia es

$$
\begin{aligned}
f_1(Ca_m, y_4') &= 14 + K_c\left(y_4' + \frac{y_4}{\tau_i}\right) - Ca_m \\
f_2(Ca_0, Ca_m) &= \frac{y_1 - y_2}{\tau} - k_r y_2 - Ca_0 \\
f_3(y_1', Ca_0) &= \frac{y_2 - y_3}{\tau} - k_r y_3 - y_1' \\
f_6(y_4') &= Spca_3 - y_3 - y_4'
\end{aligned}
$$

A continuación formamos la matriz incidencia y sin un criterio específico seleccionamos los elementos que formarán el conjunto de de salida, en este caso son los elementos encerrados en paréntesis (ver sección 1.6),

		$Variables$			
		Ca_m	y_4'	Ca_0	y_1'
	f_1	(1)	1	0	0
$Ecuaciones$	f_2	1	0	(1)	0
	f_3	0	0	1	(1)
	f_6	0	(1)	0	0

.

Después, llevamos a todos los elementos encerrados entre paréntesis a la diagonal,

		$Variables$			
		Ca_m	Ca_0	y_1'	y_4'
	f_1	(1)	0	0	1
$Ecuaciones$	f_2	1	(1)	0	0
	f_3	0	1	(1)	0
	f_6	0	0	0	(1)

,

luego, permutando filas y columnas de la matriz anterior obtenemos la siguiente matriz

		$Variables$			
		y_4'	Ca_m	Ca_0	y_1'
	f_6	(1)	0	0	0
$Ecuaciones$	f_1	1	(1)	0	0
	f_2	0	1	(1)	0
	f_3	0	0	1	(1)

.

En consecuencia, la secuencia correcta del cálculo es f_6, f_1, f_2 y f_3. Con la aplicación del método corroboramos nuestras afirmaciones al principio de este ejemplo, para que el modelo posea una estructura correcta, primero debemos codificar la ecuación (3.3.6) y posteriormente las restantes.

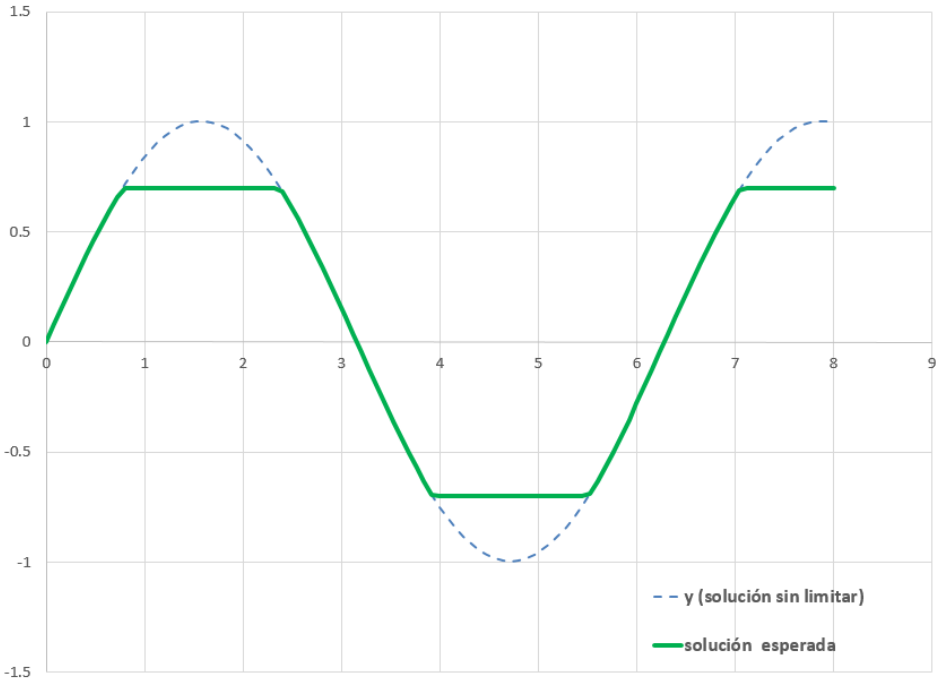

Figura 3.8: Gráfica de la solución sin limitar y la solución esperada.

3.3.2. Ejemplificación de limitadores

Con las ecuaciones

$$\begin{aligned} &y'(t) = \cos(t), \quad y(0) = 0 \\ &-0.7 \leq y(t) \leq 0.7 \end{aligned} \tag{3.3.7}$$

ejemplificamos el uso de los limitadores, para mostrar los inconvenientes de la solución pragmática y además, daremos otra solución a los limitadores que da mejores resultados.

En la figura 3.8 mostramos la solución de la ecuación 3.3.7 sin limitar (línea punteada) y la solución esperada al problema de los limitadores (línea verde).

En la figura 3.9 mostramos un diagrama que ilustra el tratamiento pragmático que se da a los limitadores, tal como lo encontramos en el trabajo de Gallardo [13]. Sin embargo, al observar las gráficas de la figura 3.11 nos percatamos que la solución pragmática (línea morada) dista mucho de la solución sin limitar (línea punteada) y, en consecuencia, de la solución esperada.

Ahora bien, en la figura 3.10 mostramos el diagrama que representa nuestra propuesta para el manejo de los limitadores, y en la figura 3.11 observamos la gráfica de nuestra propuesta (línea roja) y la solución sin limitar (línea punteada). En este caso, la solución propuesta y la solución esperada coinciden.

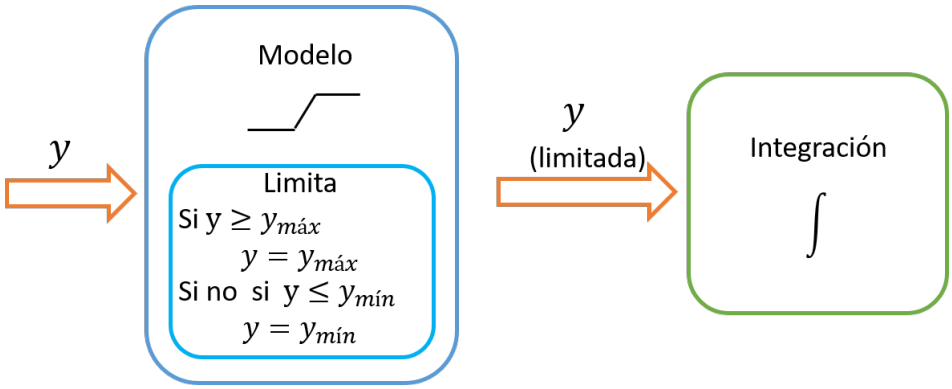

Figura 3.9: Diagrama que representa la solución a los limitadores de forma pragmática.

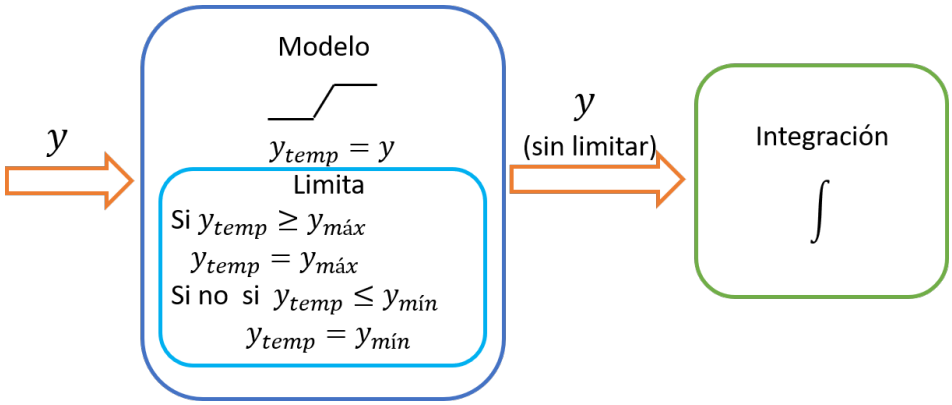

Figura 3.10: Diagrama que representa nuestra propuesta para dar solución a los limitadores.

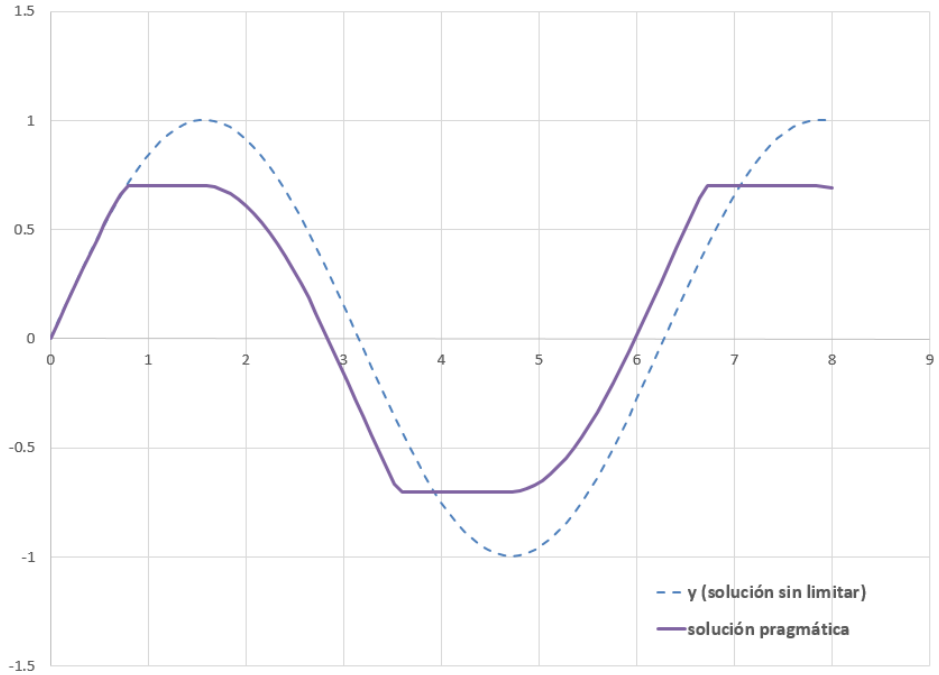

Figura 3.11: Gráfica de la solución pragmática y la solución sin limitar.

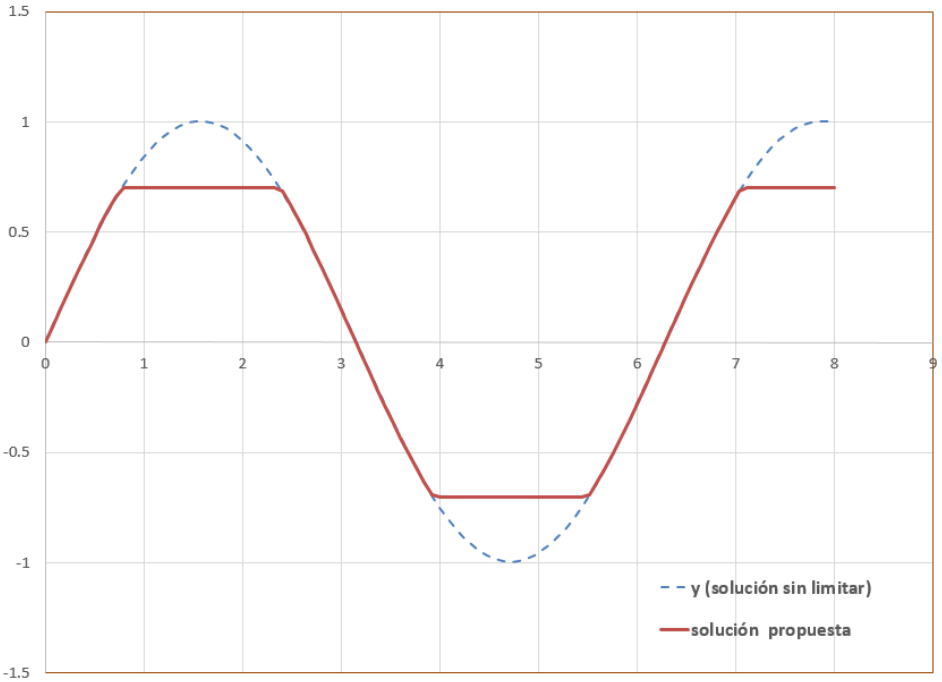

Figura 3.12: Gráfica de la solución propuesta y la solución correcta.

En conclusión, la solución prágmatica al problema de los limitadores altera la variable en cuestión, la mayoría de las veces ocurre que ésta es una variable de estado, lo que puede provocar errores en la simulación. Por lo que es recomendable tratar este problema con la solución propuesta.

3.4. Procedimiento para hallar la REA de los MML

En esta sección mostramos el procedimiento que seguimos para hallar las **REA** de los **MML**, en específico para los **R–K** porque son los que utilizamos en el capítulo 4 de experimentos.

La frontera de la región $\mathcal{R}$, definida en (1.2.4), la denotamos por $\partial\mathcal{R}$. Puesto que las raíces de $\pi(r, z) = 0$ son funciones continuas de z, entonces $z \in \partial\mathcal{R}$ cuando $\mid r_s \mid = 1$ para $s = 1, 2, \ldots, k$, esto es

$$\partial\mathcal{R} = \{z \in \mathbb{C} \mid z = \frac{\rho(e^{i\theta})}{\sigma(e^{i\theta})}, \theta \in [0, 2\pi]\}.$$

El lugar geométrico de $\partial\mathcal{R}$ lo encontramos sustituyendo $r = e^{i\theta}$ en (1.2.3a) y (1.2.3b). Sea

$$R = \sum_{j=0}^{k} \alpha_j \cos(j\theta), \qquad S = \sum_{j=0}^{k} \alpha_j \operatorname{sen}(j\theta), \tag{3.4.1}$$

$$A = \sum_{j=0}^{k} \beta_j \cos(j\theta), \qquad B = \sum_{j=0}^{k} \beta_j \operatorname{sen}(j\theta). \tag{3.4.2}$$

De acuerdo a lo anterior obtenemos que

$$\mathrm{Re}(z) = x = \frac{RA + SB}{A^2 + B^2}, \tag{3.4.3}$$

$$\mathrm{Im}(z) = y = \frac{SA - RB}{A^2 + B^2} \tag{3.4.4}$$

donde Re es la parte real del número complejo e Im la parte imaginaria.

Entonces, podemos redefinir a $\partial\mathcal{R}$ como

$$\partial\mathcal{R} = \{(x, y) \mid x = \frac{RA + SB}{A^2 + B^2}, y = \frac{SA - RB}{A^2 + B^2}\},$$

la región delimitada por este conjunto, es la zona de estabilidad absoluta del método (1.2.1).

3.4.1. Estimación de una cota superior para el paso de integración de los métodos R–K

En esta sección explicaremos cómo a partir del conocimiento de los valores característicos del sistema de **EADs**, que definen al modelo, obtenemos el tamaño del paso de integración. Lo anterior sólo lo realizamos para métodos de integración del tipo **R–K**.

Dado R_k, el número de etapas del método **R–K**, y λ el valor característico, estimamos una cota superior H para el paso de integración h del método **R–K**, tal que si $0 < h < H$ el método numérico se encuentra dentro de su **REA**.

A continuación presentamos las estimaciones para las cotas superiores de los métodos Euler, **R–K** 2, **R–K** 3 y **R–K** 4.

3.4.2. Cota superior para el método de Euler

Consideremos el polinomio de estabilidad del método de Euler

$$p_1(\overline{h}) = 1 + \overline{h}, \tag{3.4.5}$$

donde $\overline{h} = \lambda h$, y $\lambda = x + iy$, con $i = \sqrt{-1}$, $x < 0$ y $h > 0$. Entonces, reescribimos el polinomio de estabilidad de la siguiente forma, ver Lambert [21],

$$p_1(h) = 1 + h(x + iy). \tag{3.4.6}$$

Además, la región de estabilidad absoluta del método numérico la encontramos en el plano complejo $\mathbb{C}$ y está dada por

$$\mathcal{R}_1 = \left\{\overline{h} = \lambda h : |p_1(\overline{h})| < 1\right\}. \tag{3.4.7}$$

De acuerdo con la ecuación 3.4.7, buscamos el conjunto de las h tal que al multiplicarlas por λ satisfagan

$$|p_1(\overline{h})| < 1. \tag{3.4.8}$$

Entonces, la **REA** la determinamos por medio de su frontera (las h que satisfacen la igualdad de la inecuación anterior), esto es, encontramos las h que satisfagan:

$$|p_1(\overline{h})|^2 - 1 = 0, \tag{3.4.9}$$

y después hallamos una cota superior de las soluciones h.

Recordemos que $|z|^2 = z \cdot \overline{z}$. Entonces, sustituyendo la ecuación 3.4.6 en la ecuación 3.4.9 y tomando en cuenta la ecuación 3.4.5 obtenemos

$$h(2x + h(x^2 + y^2)) = 0. \tag{3.4.10}$$

Observamos que el polinomio tiene una raíz en $h = 0$ por lo cual el polinomio para encontrar H (la cota superior), se simplifica a la siguiente expresión

$$p_1(x, y) = 2x + h(x^2 + y^2). \tag{3.4.11}$$

Análogamente encontramos los polinomios asociados a los métodos **R–K** dos, tres y cuatro, que nos permiten determinar la cota superior H para el paso de integración h asociados a estas estrategias numéricas, los cuales presentamos a continuación.

$$\begin{aligned}
p_2(x, y) &= \frac{1}{4}h^3(x^2 + y^2)^2 + h^2x(x^2 + y^2) + 2hx^2 + 2x, \\
p_3(x, y) &= h^5\left(\frac{x^6}{36} + \frac{x^4y^2}{12} + \frac{x^2y^4}{12} + \frac{y^6}{36}\right) + h^4\left(\frac{x^5}{6} + \frac{x^3y^2}{3} + \frac{xy^4}{6}\right) \\
&\quad + h^3\left(\frac{7x^4}{12} + \frac{x^2y^2}{2} - \frac{y^4}{12}\right) + \frac{4h^2x^3}{3} + 2hx^2 + 2x, \\
p_4(x, y) &= h^7\left(\frac{x^8}{576} + \frac{x^6y^2}{144} + \frac{x^4y^4}{96} + \frac{x^2y^6}{144} + \frac{y^8}{576}\right) + h^6\left(\frac{x^7}{72} + \frac{x^5y^2}{24} + \frac{x^3y^4}{24} + \frac{xy^6}{72}\right) \\
&\quad + h^5\left(\frac{5x^6}{72} + \frac{x^4y^2}{8} + \frac{x^2y^4}{24} - \frac{y^6}{72}\right) + h^4\left(\frac{x^5}{4} + \frac{x^3y^2}{6} - \frac{xy^4}{12}\right) \\
&\quad + \frac{2h^3x^4}{3} + \frac{4h^2x^3}{3} + 2hx^2 + 2x,
\end{aligned}$$

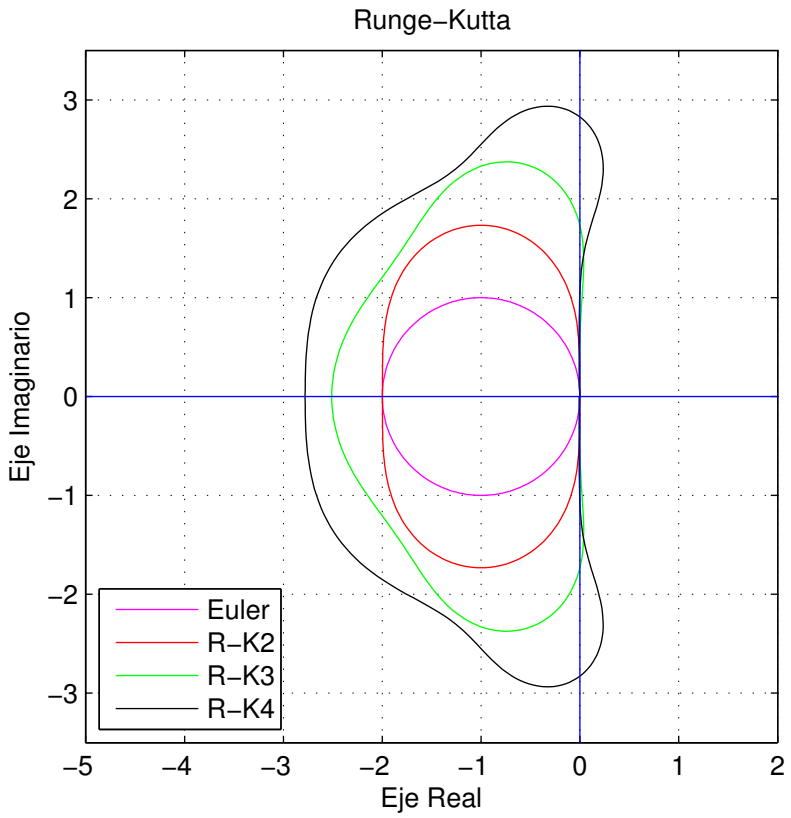

Figura 3.13: Regiones de estabilidad absoluta de los Métodos R–K.

donde $p_2(x, y)$ es el polinomio del **R–K** de orden dos, $p_3(x, y)$ el de orden tres y $p_4(x, y)$ el de orden cuatro.

La figura 3.13 nos muestra gráficamente cada una de las **REAs** de los métodos **R–K**.

Al lector interesado en el detalle de las manipulaciones algebraicas para

obtener los polinomios asociados a los métodos **R–K**, consultar el apéndice A.

Capítulo 4

Experimentos

A continuación, presentamos los experimentos numéricos realizados para evaluar el desempeño de la metodología propuesta en el capítulo 3. Simulamos tres procesos de mediana complejidad y uno que tiene todas las características genéricas de una planta de potencia, estos son: un modelo que representa Tres Reactores Tipo Tanque con Control **PI**, un conjunto de Resortes Acoplados, un modelo no lineal de un *Quadrotor* y un proceso representado por una Planta Termoeléctrica.

4.1. Tres Reactores Tipo Tanque con Control PI

En este primer experimento consideramos el *modelo de los tres reactores tipo tanque con un control **PI** realimentado*, brevemente nombrado *modelo de los tres reactores*, tal como lo presentamos en la sección 3.3.1.

4.1.1. Resultados de la aplicación de la metodología propuesta

A continuación presentamos los resultados de la aplicación de la metodología propuesta (ver sección 3.2) *al modelo de los tres reactores*. Este modelo posee cuatro ecuaciones diferenciales ordinarias, las cuales están dadas por (3.3.1), (3.3.3), (3.3.4), (3.3.5) y (3.3.6), y una ecuación algebraica (3.3.2).

De acuerdo con la metodología propuesta en la sección 3.2, primero realizamos la *prueba estructural mejorada*, para analizar si el orden de ejecución de las ecuaciones del modelo están en forma correcta, en los tiempos $t = \{0, 5, 10, 15, 25\}$, y tal como lo reportamos en la sección 3.3.1, fue necesario cambiar el orden de las ecuaciones que presentamos en 3.3.1 para superar la prueba estructural.

Luego, siguiendo nuestra metodología revisamos la composición del modelo. De los resultados obtenidos concluimos que el modelo solo contiene **EAs** lineales, por lo que únicamente utilizamos el método integrador **Ordinary Differential Equations (ODE)** de Shampine [35] que integra un sistema de **EDOs** a valores iniciales por medio de las fórmulas *Adams-Bashforth* y *Adams-Moulton* con control de error local, paso y orden variable.

Posteriormente, simulamos un estado estacionario, donde la variable controlada $y_3 = 0.101$ se encuentra en un valor deseado (o *set point*), observamos que el estacionario se mantenía, por lo que pasamos a la siguiente etapa de la metodología.

Después, elegimos un transitorio en el cual la variable controlada $y_3 = 1.0$ se encuentra fuera de su punto de referencia y por medio de la acción del control **PI** se le obliga a la corriente y_3 ir al valor deseado $SPCa_3 = 0.1$. Con el transitorio seleccionado, realizamos una simulación de 50 minutos (min), tomando muestras de los datos cada 0.25 min. El transitorio fue superado y en consecuencia esta etapa de la metodología.

Luego, para calcular el paso de integración más adecuado para cada método integrador de paso fijo, Euler, Runge–Kutta 2, Runge–Kutta 3 y Runge–Kuta 4, fue necesario disponer de la matriz jacobiana del modelo en diferentes tiempos a lo largo de la simulación. Los tiempos que elegimos en este caso, fueron $t = \{0, 5, 10, 15, 25\}$; ver sección 1.8. Cabe mencionar que tales métodos integradores son los que utilizaremos en los experimentos restantes a lo largo del presente capítulo.

En seguida, a partir de tales matrices jacobianas obtuvimos el conjunto de valores característicos para cada una de ellas, estos valores los mostramos graficados en la figura 4.1. En este caso, todos los valores característicos encontrados tienen parte real negativa. Lo que significa que el modelo es estable y se puede encontrar al menos un paso de integración h que satisfaga las condiciones de las regiones de estabilidad absoluta de los métodos numéricos integrales.

Un punto importante que resaltar es el siguiente: a mayor negatividad de la parte real del valor característico el tamaño del paso es más pequeño para los métodos propuestos.

La gráfica de barras de la figura 4.2 muestra el paso de integración recomendado por cada método integrador.

En este trabajo, ilustramos el costo computacional de cada uno de los métodos

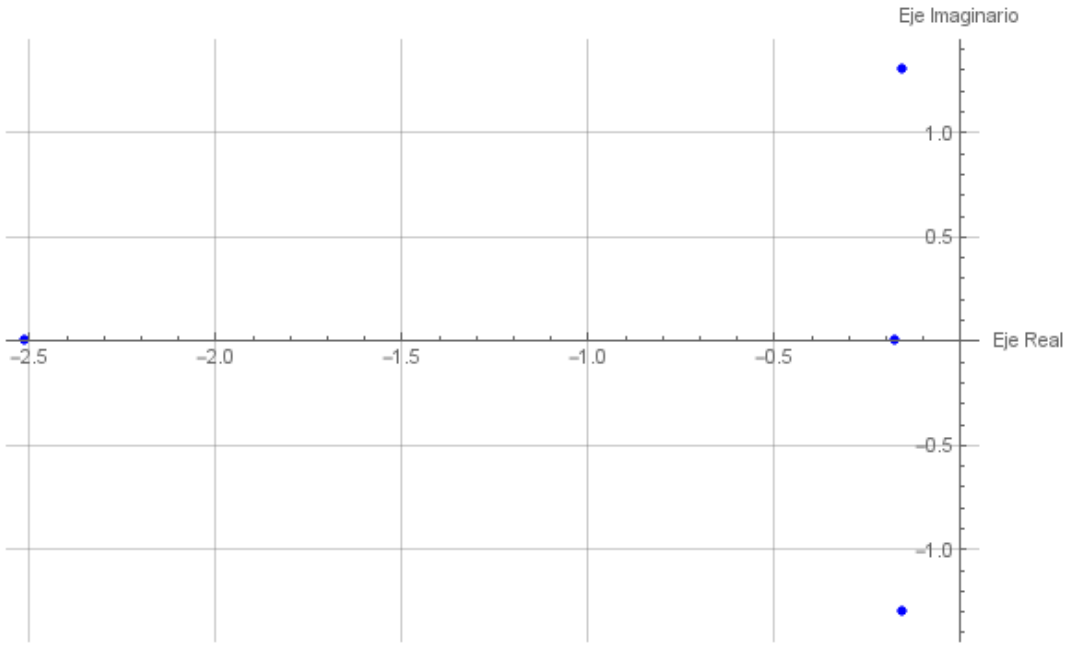

Figura 4.1: Gráfica de la distribución los valores característicos con estructura correcta del modelo de los tres reactores.

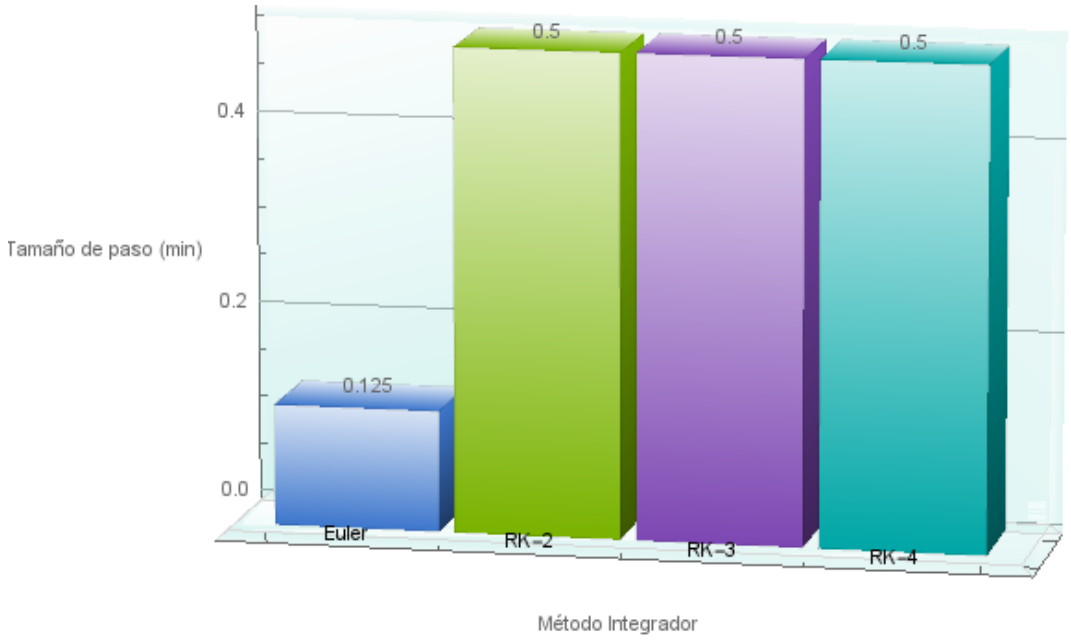

Figura 4.2: Gráfica de barras con el paso de integración recomendado para cada método integrador con estructura correcta del modelo de los tres reactores.

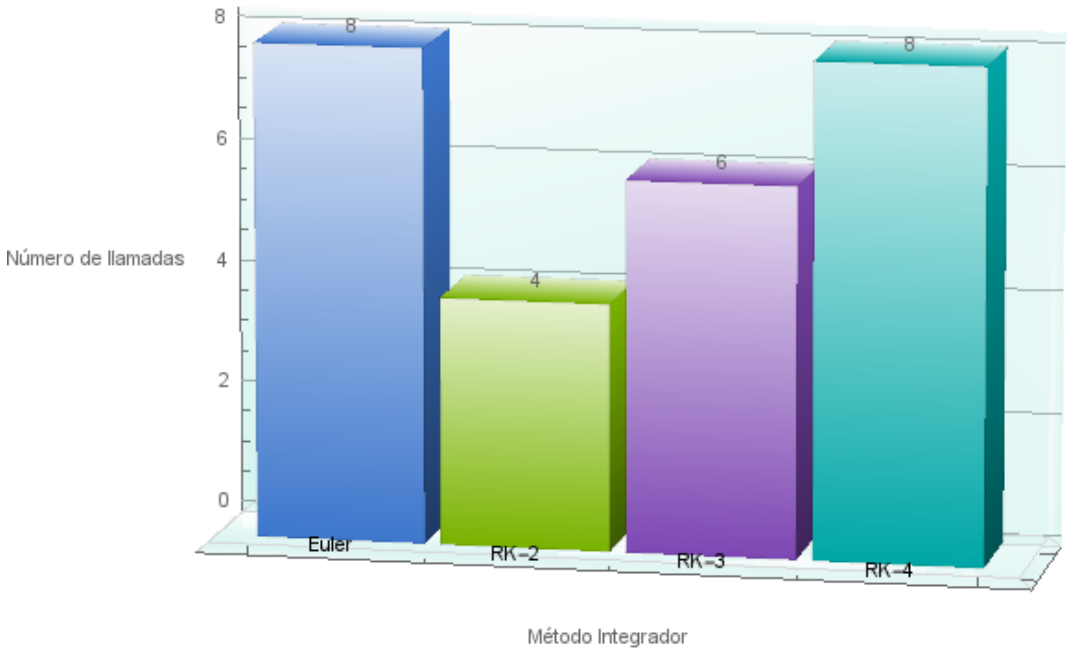

Figura 4.3: Gráfica de barras del costo computacional para cada método de integración con estructura correcta del modelo de los tres reactores.

integradores cada 0.25 *min* mediante la gráfica de barras de la figura 4.3. El cual fue calculado contando el número de llamadas realizadas por el código **ODE** al programa que contiene al modelo por periodo de muestreo.

De acuerdo con nuestra metodología el método y paso de integración recomendados son **R–K 2** y paso de integración 0.5 *min*, puesto que proporcionan el menor costo computacional.

A continuación, analizamos cómo la estructura incorrecta afecta la eficiencia computacional de la simulación. Para ello, hemos permitido que el *modelo de los tres reactores* posea una estructura incorrecta y utilizamos el método integrador, **ODE** de Shampine, con los mismos parámetros que cuando el modelo tenía una estructura correcta.

La *prueba estructural mejorada* nos arrojó los resultados esperados porque al aplicarla en los mismos tiempos que antes ($t = \{0, 5, 10, 15, 25\}$), desde el tiempo $t = 0$, la prueba mostró resultados diferentes de cero, tal como se observa en la tabla 4.1. En consecuencia, los resultados fueron congruentes, porque ya sabíamos que el sistema sí poseía una estructura incorrecta.

Después, simulamos el mismo estado estacionario ($y_3 = 0.101$), el cual fue superado a pesar de que el modelo mantenía una estructura incorrecta.

Tiempo	Ecuación	Resultados de la prueba estructural
0	1	14
5	1	3.9
10	1	1.6
15	1	0.64
25	1	0.069

Tabla 4.1: Resultados de la *Prueba estructural mejorada* del modelo de los tres reactores.

Posteriormente, realizamos simulaciones con el mismo transitorio ($y_3 = 1.0$), el cual también fue superado. Sin embargo, al calcular los los pasos de integración para cada método integrador obtuvimos resultados muy diferentes a los antes presentados. Al aproximar las matrices jacobianas en los mismos tiempos que antes ($t = \{0, 5, 10, 15, 25\}$), obtuvimos los valores característicos graficados en 4.4. Notamos que existen valores con parte real positiva muy grande y tales valores son los que afectan el tamaño del paso de integración porque no se encuentran dentro de las **REAs** de los métodos integradores candidatos, tal como lo mostramos en la sección 3.4.1.

La gráfica de barras de figura 4.5 muestra el paso de integración recomendado por cada método integrador. Observamos que los pasos de integración disminuyen notablemente en comparación con los que obtuvimos con la estructura correcta, sobre todo para los métodos de Euler y **R–K2** , puesto que antes para el primero reportaba un paso de integración de 0.125 min y ahora 7.83×10^{-13} min, y para el segundo 0.5 min y 1.1152×10^{-6} min respectivamente. El costo computacional lo mostramos en la figura 4.6 mediante una gráfica de barras, observamos que los resultados obtenidos aumentan bastante en comparación con los anteriores (ver figura 4.3). Las mayores diferencias las encontramos en los métodos de Euler y **R–K2**.

Como hemos mencionamos, el *modelo de los tres reactores* poseía una estructura incorrecta y aún así era capaz de simular un transitorio con un método de integración de paso variable con control de error. A continuación, comparamos el comportamiento de la variable y_3 (porque es crucial en la simulación) y el costo computacional cuando el sistema contaba con una estructura correcta y después con una estructura incorrecta.

La gráfica de la figura 4.7 presenta los resultados de la simulación del transitorio elegido con la estructura correcta. En el eje primario graficamos la variable y_3 la cual oscila entre -0.3 y 1, esta oscilación se amortigua y tiende al valor de 0.1 que es el valor deseado de $SPCa_3$, en el eje secundario se grafica el costo computacional por cada 0.25 minutos. El rango de valores del costo computacional varía entre 0 y 37 llamadas, la parte más costosa se da al inicio de la simulación con 37 llamadas, después de que transcurre el primer minuto de simulación, el número de llamadas varía entre 2 y 4.

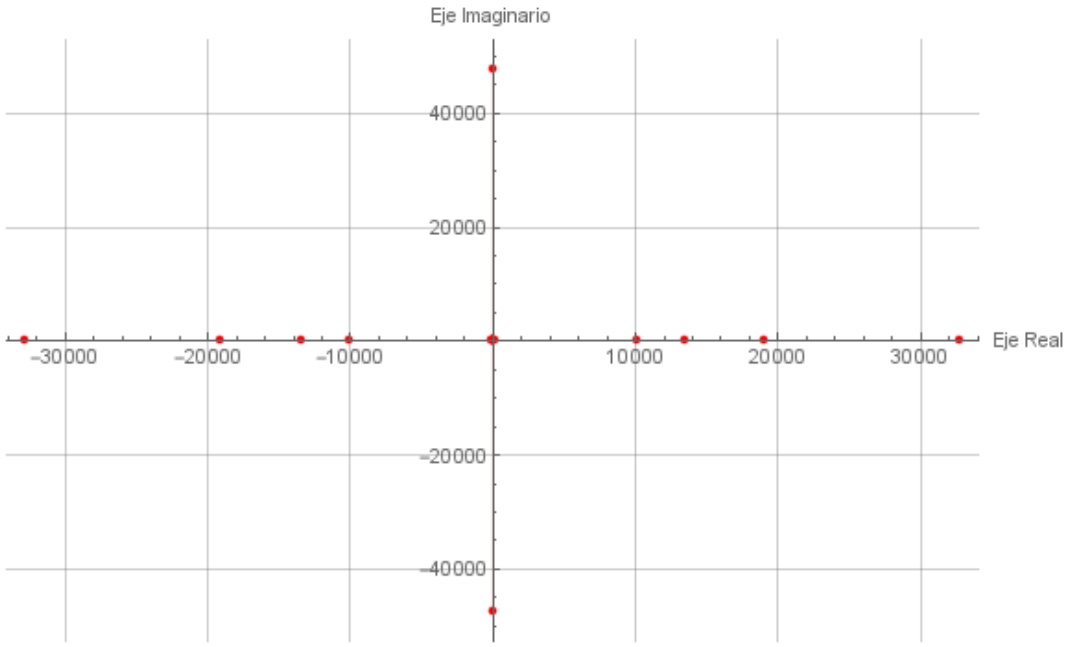

Figura 4.4: Gráfica de la distribución de los valores característicos con estructura incorrecta del modelo de los tres reactores.

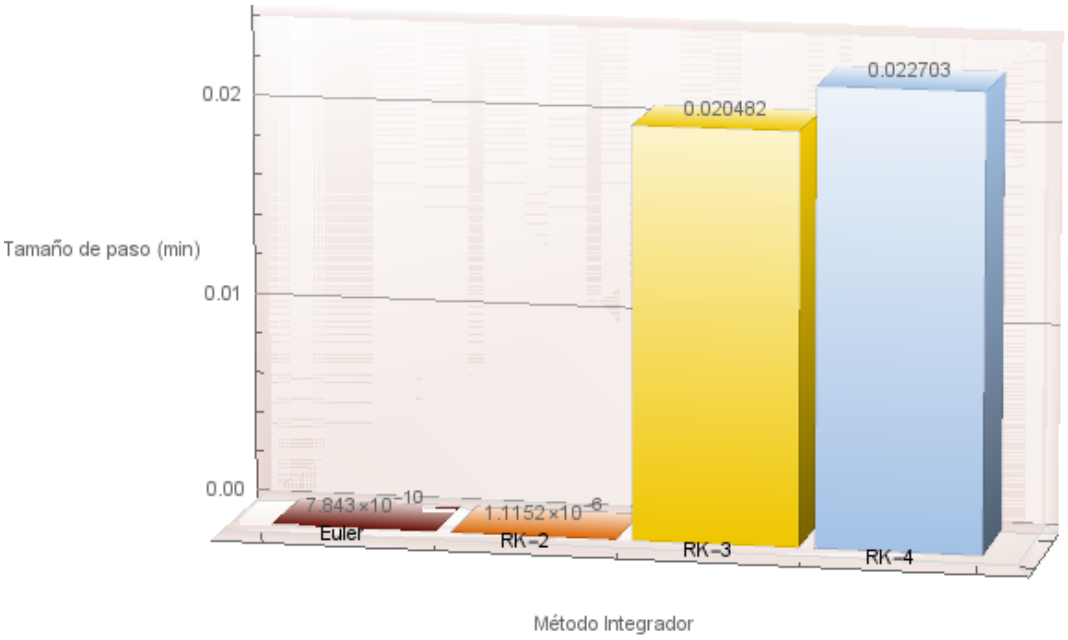

Figura 4.5: Gráfica de barras con el paso de integración recomendado para cada método integrador con estructura incorrecta del modelo de los tres reactores.

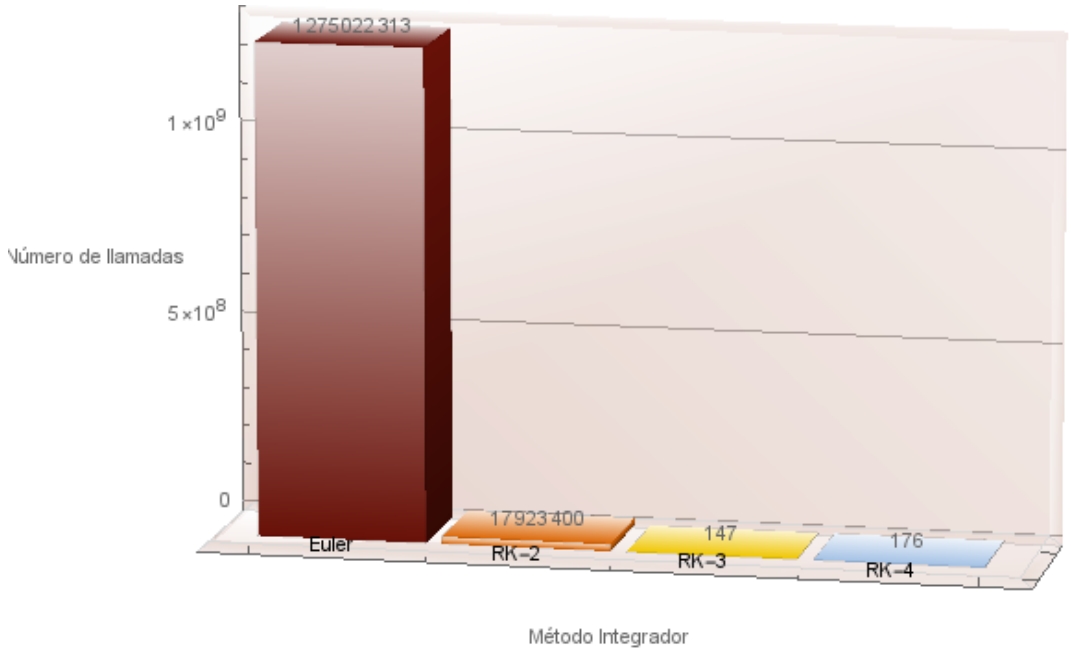

Figura 4.6: Gráfica de barras del costo computacional para cada método de integración con estructura incorrecta del modelo de los tres reactores.

Por otro lado, la gráfica de la figura 4.8 muestra los resultados obtenidos al simular el mismo transitorio pero con la estructura incorrecta. Observamos que el comportamiento de la variable y_3 es el mismo que en la gráfica anterior 4.7. Sin embargo, el costo computacional es muy distinto. Ahora varía entre 0 y 616 llamadas. El mayor número de llamadas es 616 y se da al principio, después a medida que se supera el transitorio y se alcanza el nuevo estado estacionario el número de llamadas desciende hasta quedar en 16.

Finalmente, para comparar el ahorro en el costo computacional que obtenemos al utilizar la estructura correcta del modelo en lugar de la incorrecta, calculamos el número total de llamadas para cada una de las simulaciones realizadas. Con la estructura correcta fueron 641 llamadas, mientras que con la incorrecta fueron $19,994$. Observemos que 641 representa el $3.21\,\%$ de $19,994$, por lo que tenemos un ahorro del $96.79\,\%$ si utilizamos la estructura correcta del modelo en lugar de la incorrecta.

4.2. Conjunto de Resortes Acoplados

En el segundo experimento se considera un modelo de Resortes Acoplados. Este modelo se encuentra constituido por un conjunto de masas, resortes y amortiguadores. El modelo se tomó de Ávila [3].

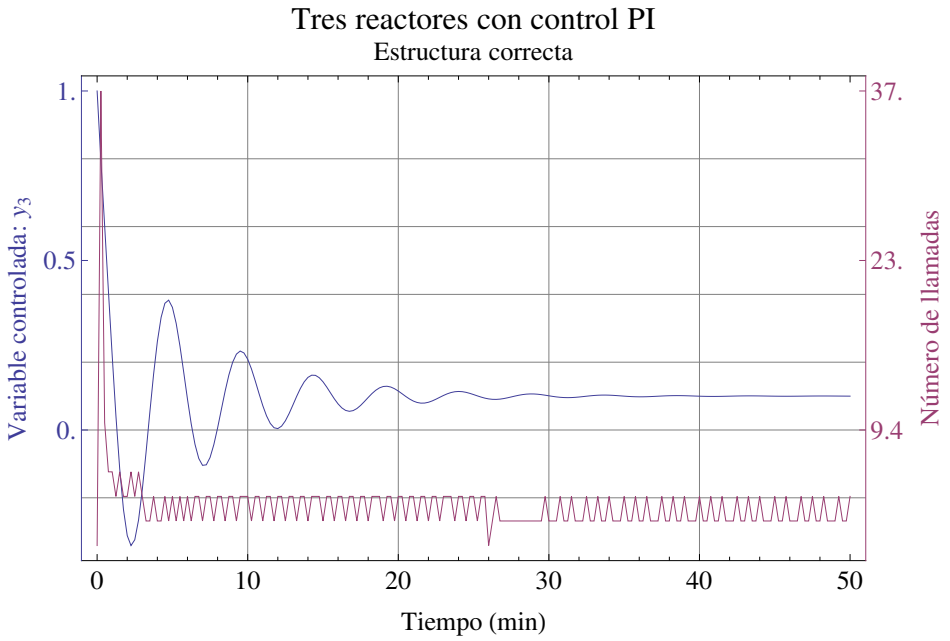

Figura 4.7: Gráfica de la variable y_3 y el costo computacional cuando la simulación posee una estructura correcta del modelo de los tres reactores.

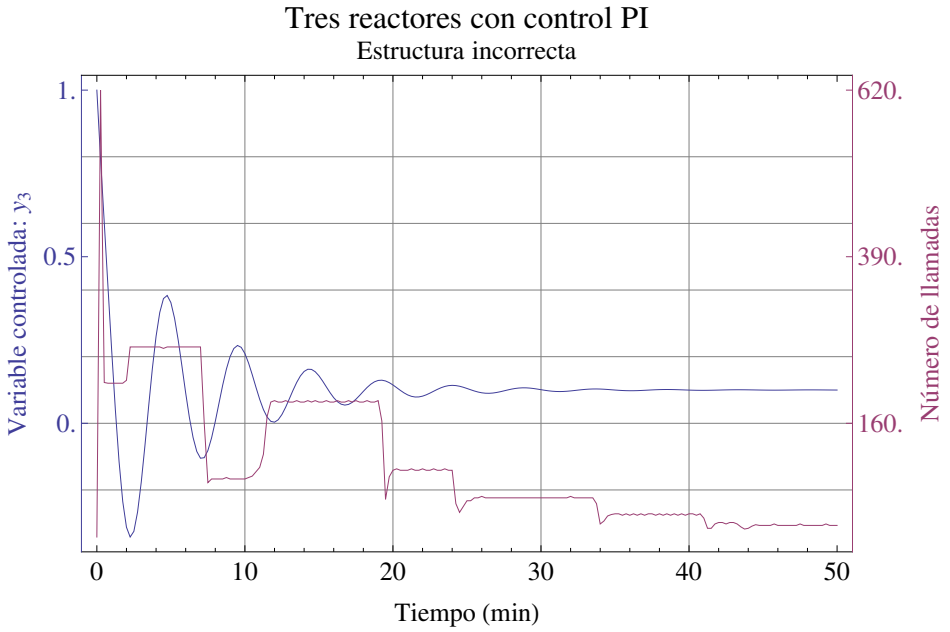

Figura 4.8: Gráfica de la variable y_3 y el costo computacional cuando la simulación posee una estructura incorrecta del modelo de los tres reactores.

El modelo matemático está compuesto por las siguientes ecuaciones

$$m_1\ddot{x}_1 = k_1(x_2 - x_1) + c_1(\dot{x}_2 - \dot{x}_1) + k_6(x_6 - x_1) + f(t), \tag{4.2.1}$$
$$m_2\ddot{x}_2 = -k_1(x_2 - x_1) - c_1(\dot{x}_2 - \dot{x}_1) + k_2(x_3 - x_2), \tag{4.2.2}$$
$$m_3\ddot{x}_3 = -k_2(x_3 - x_2) + c_2(\dot{x}_4 - \dot{x}_3) + k_3(x_4 - x_3), \tag{4.2.3}$$
$$m_4\ddot{x}_4 = -c_2(\dot{x}_4 - \dot{x}_3) - k_3(x_4 - x_3) + k_4(x_5 - x_4), \tag{4.2.4}$$
$$m_5\ddot{x}_5 = -k_4(x_5 - x_4) + c_3(\dot{x}_6 - \dot{x}_5) + k_5(x_6 - x_5), \tag{4.2.5}$$
$$m_6\ddot{x}_6 = -c_3(\dot{x}_6 - \dot{x}_5) - k_5(x_6 - x_5) - k_6(x_6 - x_1) - k_7x_6, \tag{4.2.6}$$

donde $\dot{x} = \frac{dx}{dt}$, $\ddot{x} = \frac{d^2x}{dt^2}$, las x representan los estados.

Al sistema de ecuaciones diferenciales de segundo orden representado por las ecuaciones (4.2.1) hasta (4.2.6), lo transformamos en un sistema de primer orden con doce **EDOs** por medio de los siguientes cambios de variables

$$y_1 = x_1, \quad y_3 = x_2, \quad y_5 = x_3, \quad y_7 = x_4, \quad y_9 = x_5, \quad y_{11} = x_6,$$
$$y_2 = \dot{x}_1, \quad y_4 = \dot{x}_2, \quad y_6 = \dot{x}_3, \quad y_8 = \dot{x}_4, \quad y_{10} = \dot{x}_5, \quad y_{12} = \dot{x}_6.$$

En consecuencia, el sistema de **EDOs** de primer orden es el siguiente

$$\begin{aligned}
\dot{y}_1 &= y_2,\\
\dot{y}_2 &= \frac{k_1}{m_1}(y_3 - y_1) + \frac{k_6}{m_1}(y_{11} - y_1) + \frac{c_1}{m_1}(y_4 - y_2) + \frac{f(t)}{m_1},\\
\dot{y}_3 &= y_4,\\
\dot{y}_4 &= -\frac{k_1}{m_2}(y_3 - y_1) + \frac{k_2}{m_2}(y_5 - y_3) - \frac{c_1}{m_2}(y_4 - y_2),\\
\dot{y}_5 &= y_6,\\
\dot{y}_6 &= -\frac{k_2}{m_3}(y_5 - y_3) + \frac{k_3}{m_3}(y_7 - y_5) + \frac{c_2}{m_3}(y_8 - y_6),\\
\dot{y}_7 &= y_8,\\
\dot{y}_8 &= -\frac{k_3}{m_4}(y_7 - y_5) - \frac{c_2}{m_4}(y_8 - y_6) + \frac{k_4}{m_4}(y_9 - y_7),\\
\dot{y}_9 &= y_{10},\\
\dot{y}_{10} &= -\frac{k_4}{m_5}(y_9 - y_7) + \frac{k_5}{m_5}(y_{11} - y_9) + \frac{c_3}{m_5}(y_{12} - y_{10}),\\
\dot{y}_{11} &= y_{12},\\
\dot{y}_{12} &= -\frac{k_5}{m_6}(y_{11} - y_9) - \frac{k_6}{m_6}(y_{11} - y_1) + \frac{k_7}{m_6}y_{11-} \frac{c_3}{m_6}(y_{12} - y_{10}).
\end{aligned}$$

Los parámetros del modelo de los resortes se muestran en la tabla 4.2.

Parámetro	Valor	Descripción
$m_1, m_2, m_3, m_4, m_5, m_6$	1	Masas de los dispositivos
$f(t)$	0.5N	Fuerza aplicada
k_1	1	Constante de resorte
k_2	0.1	Constante de resorte
k_3	100	Constante de resorte
k_4	0.1	Constante de resorte
k_5	0.5	Constante de resorte
k_6	0.05	Constante de resorte
k_7	1	Constante de resorte
c_1	0.1	Constante de amortiguamento
c_2	0.1	Constante de amortiguamento
c_3	0.1	Constante de amortiguamento

Tabla 4.2: Tabla de parámetros del modelo de los resortes.

4.2.1. Resultados de la aplicación de la metodología propuesta

A continuación presentamos los resultados de la aplicación de la metodología propuesta (ver sección 3.2) al *conjunto de resortes acoplados* de forma análoga al experimento de los *tres reactores* (ver sección 4.1).

Así, primero realizamos la *prueba estructural mejorada*, para analizar si el orden de ejecución de las ecuaciones del modelo tenía la forma correcta, en los tiempos $t = \{0, 3, 50, 100, 250\}$. La prueba mostró que el modelo si poseía una estructura correcta.

Luego, revisamos la composición del modelo. De los resultados obtenidos concluimos que el modelo no contiene **EAs** no lineales, por lo que también utilizamos el método integrador **ODE** de Shampine [35].

Posteriormente, simulamos un estado estacionario en donde la fuerza aplicada a los resortes acoplados es 0 Newtons (N), y observamos que se mantuvo, por lo que pasamos a la siguiente etapa de la metodología.

Después, elegimos un transitorio en donde la fuerza aplicada al conjunto de resortes acoplados es de $0.5N$, para que los resortes oscilaran. Con el transitorio seleccionado realizamos una simulación de 300 segundos (s), tomando muestras de los datos cada $1.5s$. Esta etapa de la metodología también fue superada, por lo que

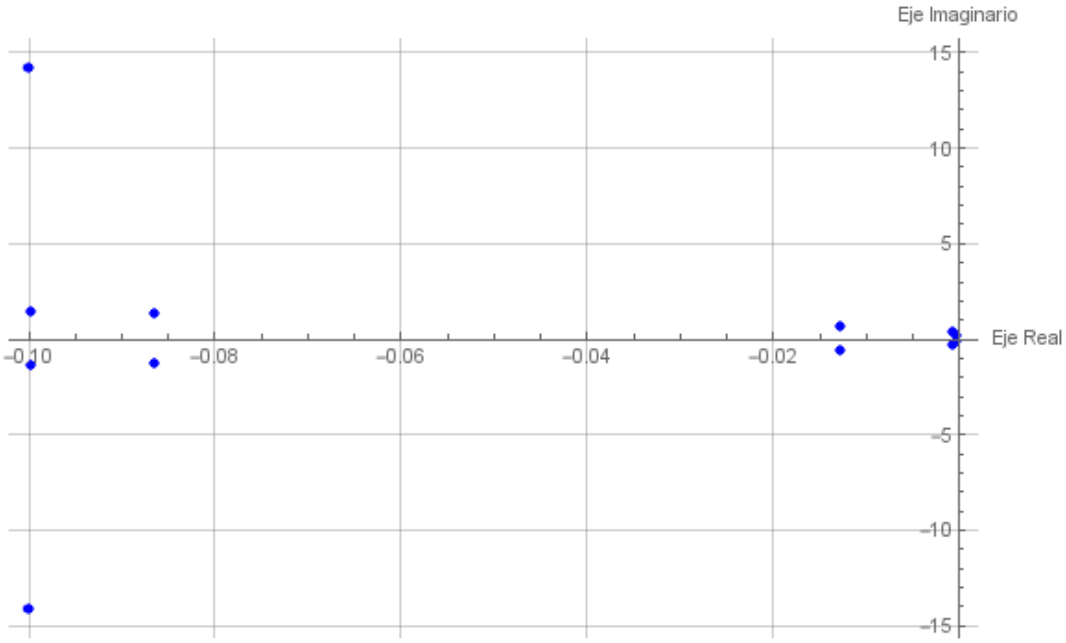

Figura 4.9: Gráfica de la distribución los valores característicos con estructura correcta del modelo de los resortes.

avanzamos a la siguiente.

Luego, para calcular los pasos de integración más adecuados a cada método integrador de paso fijo, Euler, **R–K** 2, **R–K** 3 y **R–K** 4, fue necesario disponer de la matriz jacobiana de modelo en diferentes tiempos a lo largo de la simulación. Los tiempos que elegimos fueron $t = \{0, 5, 10, 15, 20, 80, 100, 260\}$ (ver sección 1.8).

En seguida, a partir de las matrices jacobianas aproximadas obtuvimos el conjunto de valores característicos para cada una de ellas. Los cuales los mostramos graficados en la figura 4.9. En este caso, todos los valores característicos encontrados tienen parte real negativa. Lo que significa que el modelo es estable y se puede encontrar al menos un paso de integración h que satisfaga las condiciones de las regiones de estabilidad absoluta de los métodos numéricos integrales.

La gráfica de barras de la figura 4.10 muestra el paso de integración recomendado por cada método integrador. El costo computacional de cada uno de estos métodos cada $1.5s$ lo ilustramos mediante la gráfica de barras de la figura 4.11.

Finalmente, el método y paso de integración recomendados son **R–K 3** y paso de integración de $0.125s$, puesto que proporcionan el menor costo computacional.

Cabe mencionar que en este modelo no aplicamos la metodología con una estructura incorrecta, porque las ecuaciones del modelo no son dependientes unas de otras.

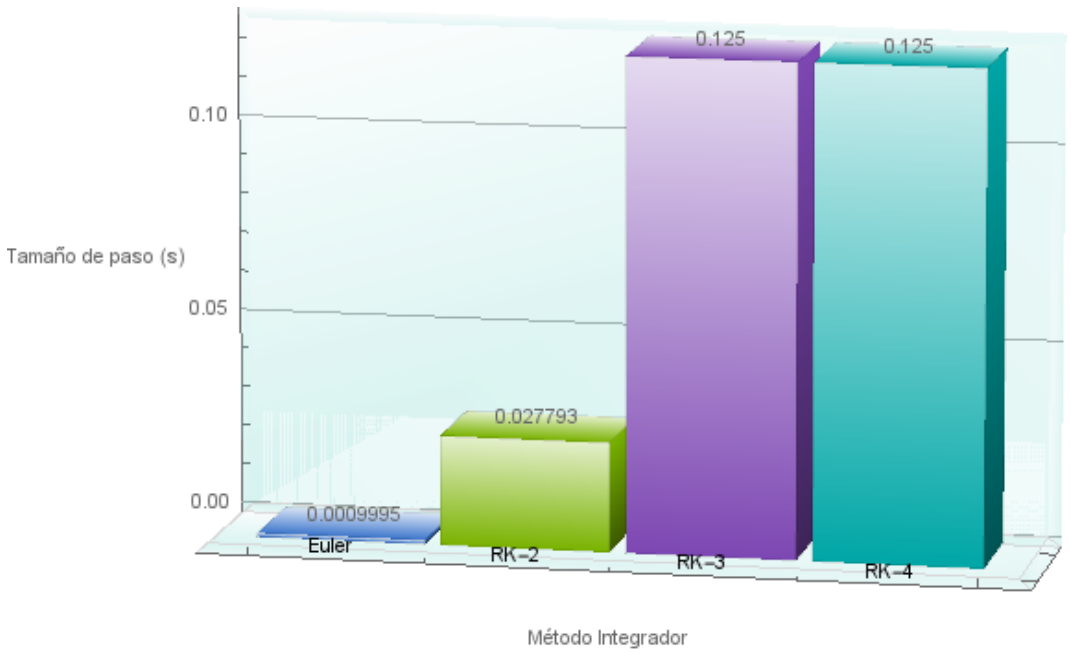

Figura 4.10: Gráfica de barras con el paso de integración recomendado para cada método integrador con estructura correcta del modelo de los resortes.

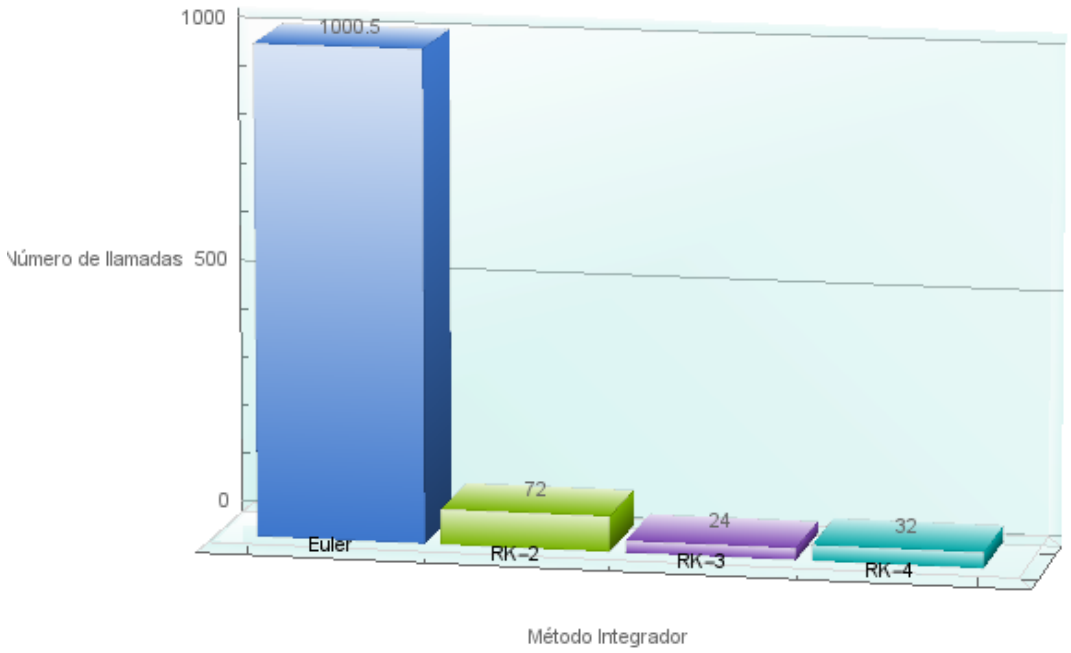

Figura 4.11: Gráfica de barras del costo computacional para cada método de integración con estructura correcta del modelo de los resortes.

4.3. Modelo no lineal de un *quadrotor*

En este experimento consideramos el modelo no lineal de un *quadrotor*, el cual fue tomado de Zhu et al. [40].

El *quadrotor* es un tipo vehículo aéreo que difiere del helícoptero convencional en su dinámica y control, ver Zhue et al. [40]. Pero los estados que describen la posición del vehículo aéreo siguen siendo los mismos: la velocidad correspondiente a los ejes cuerpo (u, v, w), la velocidad angular al rededor de los ejes cuerpo (p, q, r), la posición con respecto a los ejes terrestres (x, y, z), y los ánglulos de Euler: alabeo, cabeceo y guiñada (ϕ, θ, ψ).

La cinemática y dinámica en el eje z está dada por las siguientes ecuaciones, de acuerdo con Zhu et al [40].

$$\begin{cases} \dot{z} &= w, \\ \dot{w} &= (U_1 \cdot \cos\phi \cdot \cos\theta - m \cdot g)/m. \end{cases} \tag{4.3.1}$$

Las derivadas de los ángulos de Euler son funciones de ϕ, θ, ψ y la velocidad angular p, q, r, las cuales mostramos a continuación.

$$\begin{cases} \dot{\phi} &= (p \cdot \cos\theta + q \cdot \operatorname{sen}\phi \cdot \operatorname{sen}\theta + r \cdot \cos\phi \cdot \operatorname{sen}\theta)/\cos\theta, \\ \dot{\theta} &= q \cdot \cos\phi + r \cdot \operatorname{sen}\phi, \\ \dot{\psi} &= (q \cdot \operatorname{sen}\phi + r \cdot \cos\phi)/\cos\theta. \end{cases} \tag{4.3.2}$$

Las derivadas de la velocidad angular son

$$\begin{cases} \dot{p} &= [\sqrt{2} \cdot l \cdot U_2 + q \cdot r \cdot (I_y - I_z) - J_{TP} \cdot q \cdot \Omega]/I_x, \\ \dot{q} &= [\sqrt{2} \cdot l \cdot U_3 + p \cdot r \cdot (I_z - I_x) - J_{TP} \cdot q \cdot \Omega]/I_y, \\ \dot{r} &= [\sqrt{2} \cdot U_4 + q \cdot p \cdot (I_x - I_y)]/I_z. \end{cases} \tag{4.3.3}$$

Los controles proporcionales derivativos (PD) están dados por

$$\begin{cases} U_1 &= k_{p1} \cdot (z_t - z(t)) - k_{d1} \cdot \dot{z}, \\ U_2 &= k_{p2} \cdot (\phi_t - \phi(t)) - k_{d2} \cdot \dot{\phi}, \\ U_3 &= k_{p3} \cdot (\theta_t - \theta(t)) - k_{d2} \cdot \dot{\theta}, \\ U_4 &= k_{p4} \cdot (\psi_t - \psi(t)) - k_{d4} \cdot \dot{\psi}. \end{cases} \tag{4.3.4}$$

Donde k_{p1}, k_{p2}, $k_{p3,}$ k_{p4} son parámetros de ganancias del control proporcional y k_{d1}, k_{d2}, $k_{d3,}$ k_{d4} son constantes del control derivativo. En la tabla 4.5 mostramos los valores de tales constantes.

La tabla 4.3 muestra una breve descripción de las variables presentes en este modelo, mientras que la tabla 4.4 proporciona una descripción de las constantes.

Parámetro	Unidad	Descripción
z	m	altura
w	m/s	velocidad en altura
ϕ	rad	ángulo de alabeo
θ	rad	ángulo de cabeceo
ψ	rad	ángulo de guiñada
p	rad/s	velocidad angular en el eje x
q	rad/s	velocidad angular en el eje y
r	rad/s	velocidad angular en el eje z
U_1	N	salida del control en altura
U_2	N	salida del control en alabeo
U_3	N	salida del control en cabeceo
U_4	$N \cdot m$	salida del control en guiñada

Tabla 4.3: Descripción de variables del modelo del *quadrotor*.

Parámetro	Valor	Descripción
$I_{x,y,z}$	$0.08N \cdot m$	inercia del cuerpo
m	$1kg$	masa del *quadrotor*
g	$9.81m/s^2$	aceleración gravitacional
p_t	$5.42 \times 10^{-5}N \cdot s^2$	coeficiente de fuerza de elevación de las hélices
p_d	$1.1 \times 10^{-6}N \cdot m \cdot s^2$	momento del coeficiente de resistencia
l	$0.24m$	distancia entre el motor y el CG
μ	1	coeficiente de resistencia (del momento)
J_{TP}	$1.04 \times 10^{-4}N \cdot m \cdot s^2$	momento de inercia de las hélices

Tabla 4.4: Descripción de parámetros del modelo del *quadrotor*.

Constantes	Valor	Descripción
k_{p1}	48.270047298107329	constantes del control proporcional
k_{p2}	13.074313157278580	
k_{p3}	11.589080252102653	
k_{p4}	6.183970583567265	
k_{d1}	11.098764953224586	constantes del control derivativo
k_{d2}	1.436416916730880	
k_{d3}	2.737102082342955	
k_{d4}	2.910128388274638	

Tabla 4.5: Valores que toman las constantes del control PI del *quadrotor*.

Tiempo	Ecuación	Resultados de la prueba estructural
0	2	-24
0.01	2	-23
0.5	2	-4.8
1	2	-9.8
4	2	-9.8
4.5	2	-9.8

Tabla 4.6: Resultados de la *Prueba estructural mejorada* del modelo del *quadrotor*.

4.3.1. Resultados de la aplicación de la metodología propuesta

A continuación presentamos los resultados de la aplicación de la metodología propuesta (ver sección 3.2) al modelo no lineal del *quadrotor*, que se realizaron de forma análoga al experimento presentado en 4.1.

Primero realizamos la *prueba estructural mejorada*, para analizar si el orden de ejecución de las ecuaciones del modelo tenía la forma correcta, en los tiempos $t = \{0, 0.01, 0.5, 1, 4, 4.5\}$, con la codificación de las ecuaciones que componen el modelo tal como se presenta en este trabajo. Sin embargo, la prueba no fue superada. Desde el tiempo $t = 0$, la prueba mostró resultados diferentes de cero, tal como se observa en la tabla 4.6. En consecuencia, el sistema tiene una estructura incorrecta con la codificación dada.

Entonces, buscamos el orden de ejecución de las ecuaciones que proporcionara una estructura correcta al modelo. Después de aplicar la *prueba estructural mejorada* encontramos que el orden correcto de ejecución de las ecuaciones se presenta al

codificarlas en el siguiente orden, a saber de la ecuación 4.3.5 a la 4.3.16.

$$\dot{z} = w, \tag{4.3.5}$$
$$U_1 = k_{p1} \cdot (z_t - z(t)) - k_{d1} \cdot \dot{z}, \tag{4.3.6}$$
$$\dot{w} = (U_1 \cdot \cos\phi \cdot \cos\theta - m \cdot g)/m, \tag{4.3.7}$$
$$\dot{\phi} = (p \cdot \cos\theta + q \cdot \operatorname{sen}\phi \cdot \operatorname{sen}\theta + r \cdot \cos\phi \cdot \operatorname{sen}\theta)/\cos\theta, \tag{4.3.8}$$
$$U_2 = k_{p2} \cdot (\phi_t - \phi(t)) - k_{d2} \cdot \dot{\phi}, \tag{4.3.9}$$
$$\dot{\theta} = q \cdot \cos\phi + r \cdot \operatorname{sen}\phi, \tag{4.3.10}$$
$$U_3 = k_{p3} \cdot (\theta_t - \theta(t)) - k_{d2} \cdot \dot{\theta}, \tag{4.3.11}$$
$$\dot{\psi} = (q \cdot \operatorname{sen}\phi + r \cdot \cos\phi)/\cos\theta, \tag{4.3.12}$$
$$U_4 = k_{p4} \cdot (\psi_t - \psi(t)) - k_{d4} \cdot \dot{\psi}, \tag{4.3.13}$$
$$\dot{p} = [\sqrt{2} \cdot l \cdot U_2 + q \cdot r \cdot (I_y - I_z) - J_{TP} \cdot q \cdot \Omega]/I_x, \tag{4.3.14}$$
$$\dot{q} = [\sqrt{2} \cdot l \cdot U_3 + p \cdot r \cdot (I_z - I_x) - J_{TP} \cdot q \cdot \Omega]/I_y, \tag{4.3.15}$$
$$\dot{r} = [\sqrt{2} \cdot U_4 + q \cdot p \cdot (I_x - I_y)]/I_z. \tag{4.3.16}$$

Después de encontrar la estructura correcta, siguiendo la metodología, revisamos la composición del modelo. De los resultados obtenidos concluimos que el modelo no cuenta con **EAs**. Además, la no linealidad del modelo del *quadrotor* se presenta en las **EDOs**, en este caso también utilizamos el método integrador **ODE** de Shampine [35].

Posteriormente, simulamos un estado estacionario en donde la variables controladas z, ϕ, θ y ψ ya llegaron a sus valores deseados (*set point*), $1, 0, 0$, y $\pi/6$ respectivamente. Observamos que el estado estacionario se mantuvo, por lo que pasamos a la siguiente etapa de la metodología.

Después, elegimos un transitorio en el cual las variables controladas $\phi = \pi/4$, $\theta = \pi/4$, $\psi = \pi/4$ y $z = 0$ se encuentran fuera de su punto de referencia. Mediante la simulación se llevará al *quadrotor* al estado en el que $(z, \phi, \theta, \psi) = (1, 0, 0, \pi/6)$. Con el transitorio seleccionado realizamos una simulación de $5s$, tomando muestras de los datos cada $0.025s$. Este estado de la simulación también fue superado y proseguimos con la siguiente etapa de la metodología.

Luego, para calcular el paso de integración más adecuado para cada método integrador de paso fijo, Euler, **R–K** 2, **R–K** 3 y **R–K** 4, dispusimos de la matriz jacobiana del modelo en diferentes tiempos a lo largo de la simulación. Los tiempos que elegimos fueron $t = \{0, 0.01, 0.5, 1, 4, 4.5\}$ (ver sección 1.8).

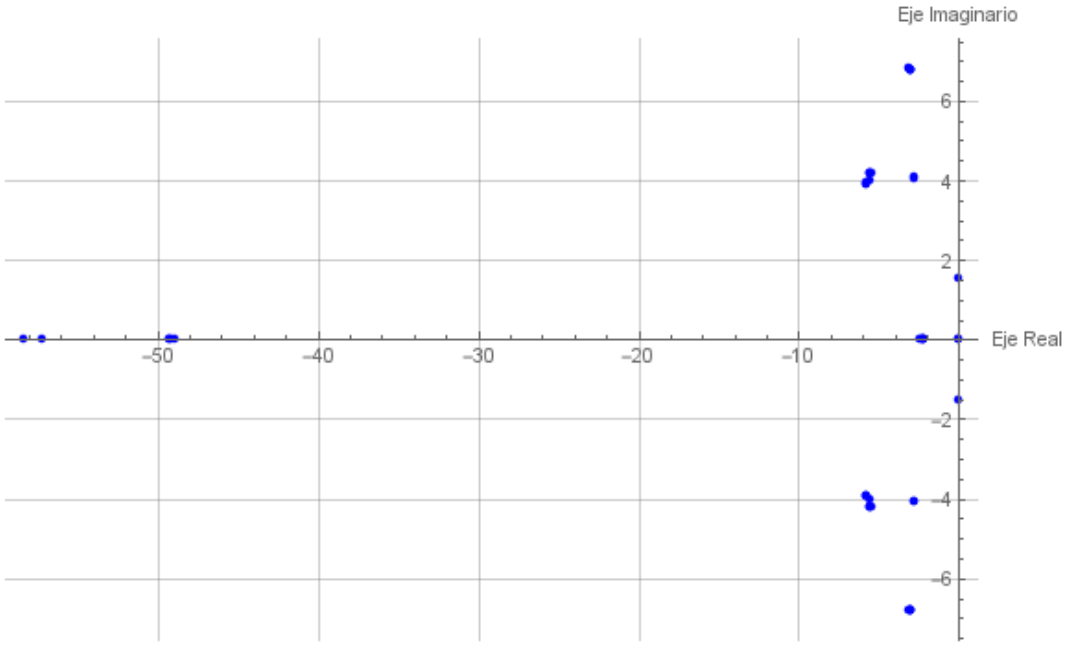

Figura 4.12: Gráfica de la distribución los valores característicos con estructura correcta del modelo del *quadrotor*.

A partir de las matrices jacobianas aproximadas obtuvimos el conjunto de valores característicos para cada una de ellas. Estos los mostramos graficados en la figura 4.12. Al igual que en la sección 4.1, todos los valores característicos encontrados tienen parte real negativa y aunque existen algunos muy cercanos a cero, el modelo aún es estable y se puede encontrar al menos un paso de integración h que satisfaga las condiciones de las regiones de estabilidad absoluta de los métodos numéricos integrales.

La gráfica de barras de la figura 4.13 muestra el paso de integración recomendado por cada método integrador.

También, ilustramos el costo computacional de cada uno de estos métodos cada $0.025s$ mediante la gráfica de barras de la figura 4.14.

Después de aplicar la metodología, el método y paso de integración recomendados son Euler y paso de integración $0.03428s$, porque proporcionan el menor costo computacional.

Además, analizamos cómo la estructura incorrecta afecta la eficiencia computacional de la simulación. Por lo que hemos permitido que el modelo no lineal del *quadrotor* posea una estructura incorrecta y utilizamos el método integrador, **ODE** de Shampine [35], con los mismos parámetros que cuando el modelo tenía una estructura correcta.

Simulamos el mismo estado estacionario: $(z, \phi, \theta, \psi) = (1, 0, 0, \pi/6)$, el cual fue superado a pesar de que el modelo tenía una estructura incorrecta.

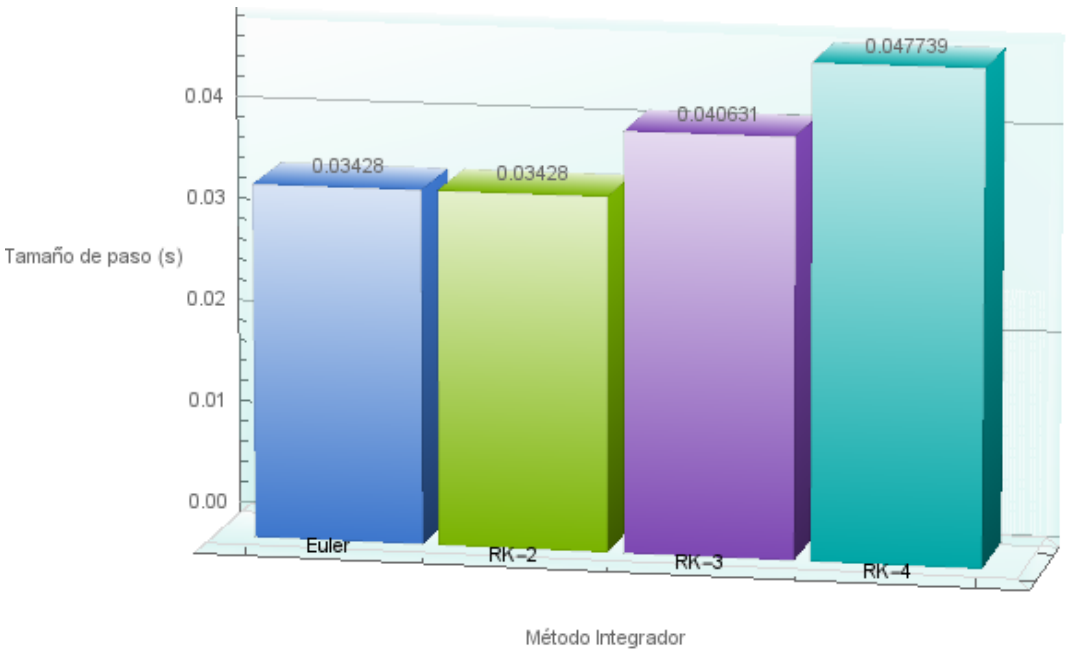

Figura 4.13: Gráfica de barras con el paso de integración recomendado para cada método integrador con estructura correcta del modelo del *quadrotor*.

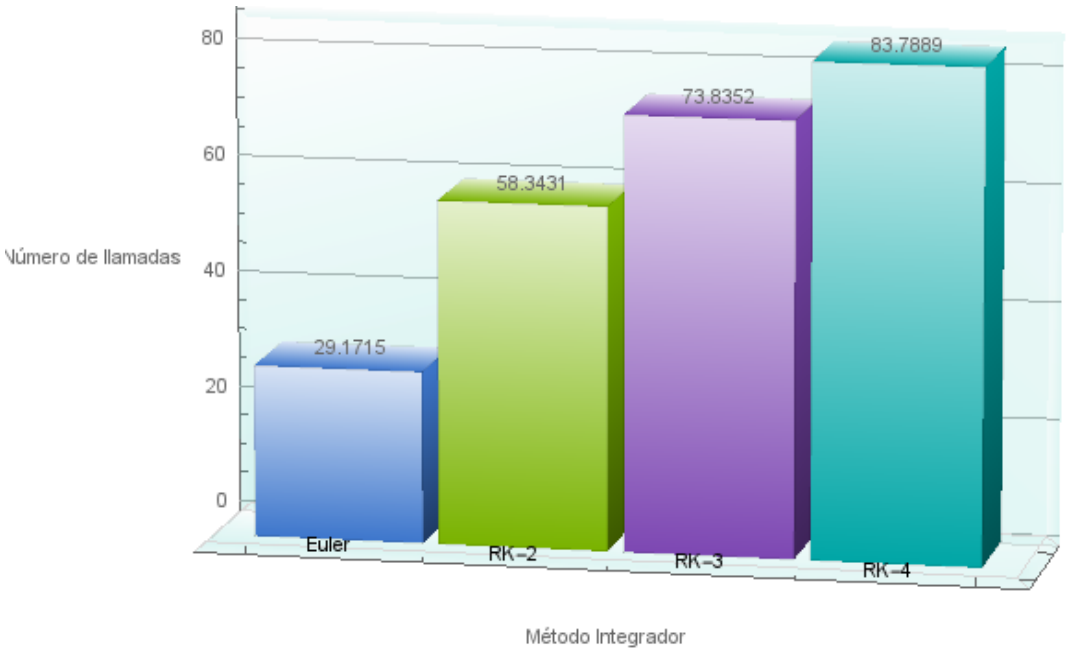

Figura 4.14: Gráfica de barras del costo computacional para cada método de integración con estructura correcta del modelo del *quadrotor*.

Posteriormente, realizamos simulaciones con el mismo transitorio: $(z, \phi, \theta, \psi) = (\pi/4, \pi/4, \pi/4, 0)$, el cual también fue superado. Luego, al aproximar las matrices jacobianas en los mismos tiempos que antes ($t = \{0, 0.01, 0.5, 1, 4, 4.5\}$), obtuvimos los valores característicos graficados en 4.15. Notamos que existen valores con parte real positiva muy grande e imaginarios puros, tales valores son los que afectan el tamaño del paso de integración porque no se encuentran dentro de las **REAs** de los métodos integradores candidatos, ver sección 3.4.1.

La gráfica de barras de figura 4.16 muestra el paso de integración recomendado para cada método integrador. Observamos que los pasos de integración coinciden con los que obtuvimos con la estructura correcta, esto sucede porque al calcular los pasos de integración recomendados, los valores característicos con parte real positiva y muy grande se omiten. En consecuencia, los valores característicos seleccionados coinciden con los que utilizamos en la estructura correcta. Por lo que el costo computacional, mostrado en la figura 4.17 mediante un gráfica de barras, también coincide con el de la estructura correcta.

Analicemos, por un momento, lo que sucede con la REAs del método integrador Euler, que está dada por la inecuación $|1 + \lambda h| < 1$, ver sección 3.4.2 , cuando el valor característico $\lambda = x + iy$ tiene parte real positiva. Por simplicidad, supongamos que $y = 0$. Como $h > 0$, entonces $|1 + \lambda h| \geq 1$. Por lo tanto, con tal valor característico estamos fuera de la región de estabilidad del método de Euler.

Observemos que los métodos de integración **R–K 3** y **R–K 4** logran capturar valores característicos con parte real positiva, pero cercanos al origen, ver gráfica 3.13. En general, si la parte real es positiva el modelo es inestable.

Como mencionamos, el modelo no lineal del *quadrotor* tenía una estructura incorrecta y aún así era capaz de simular un transitorio con un método de integración de paso variable con control de error. A continuación, para observar la mejora que se obtiene al utilizar la estructura correcta en lugar de la incorrecta, analizamos el costo computacional y el comportamiento de la variable z (porque nos interesa que alcance el valor de 1) realizando simulaciones con cada una de las estructuras.

La gráfica de la figura 4.18 presenta los resultados de la simulación del transitorio elegido con la estructura correcta. En el eje primario graficamos la variable z la cual parte del valor 0 y alcanza el 0.8, el cual está próximo al valor deseado 1. En el eje secundario se grafica el costo computacional por cada $0.025s$. El rango de valores del costo computacional varía entre 0 y 56 llamadas, la parte más costosa se da al inicio de la simulación con 56 llamadas, después de que transcurre el primer segundo de simulación, el número de llamadas varía entre 2 y 0.

Por otro lado, la gráfica de la figura 4.19 muestra los resultados obtenidos al simular el mismo transitorio pero con la estructura incorrecta. Observamos que el comportamiento de la variable z es el mismo que en la gráfica anterior 4.18. Sin embargo,

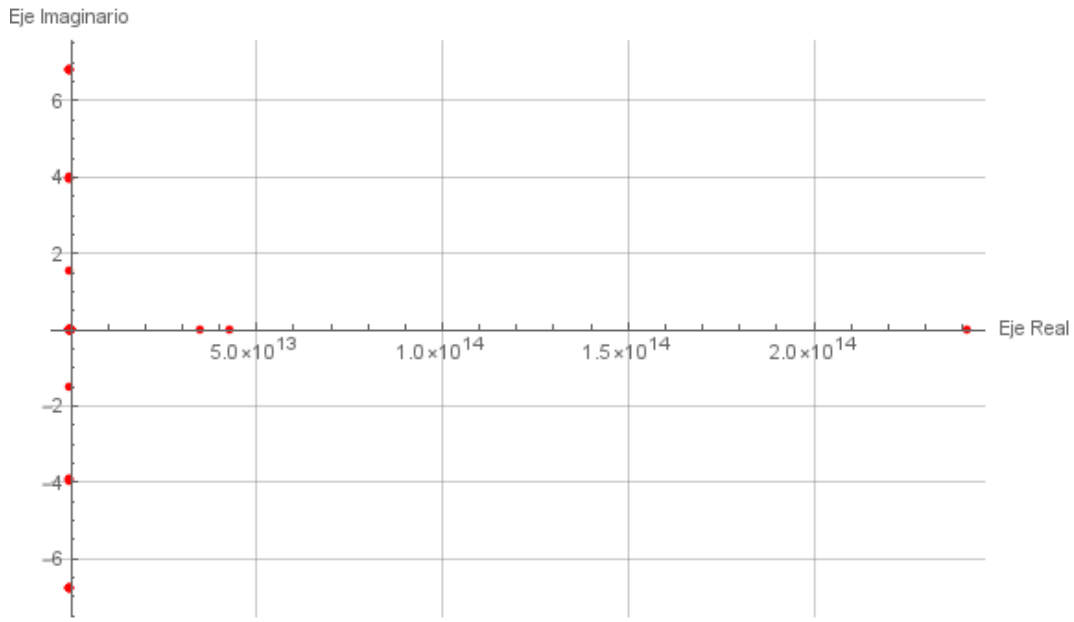

Figura 4.15: Gráfica de la distribución de los valores característicos con estructura incorrecta del modelo del *quadrotor*.

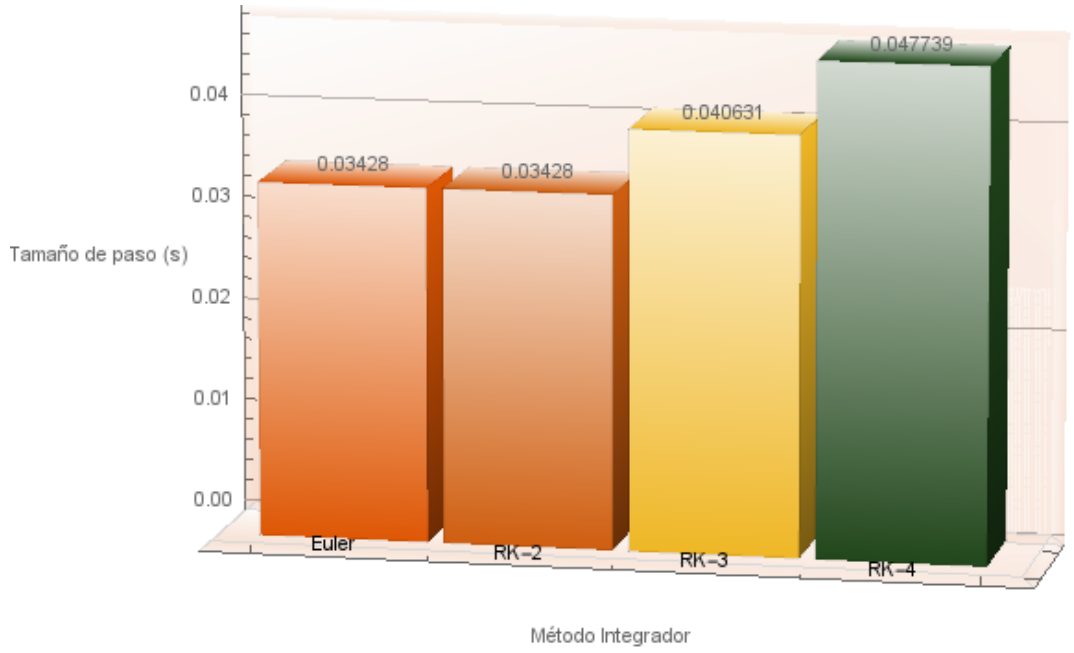

Figura 4.16: Gráfica de barras con el paso de integración recomendado para cada método integrador con estructura incorrecta del modelo del *quadrotor*.

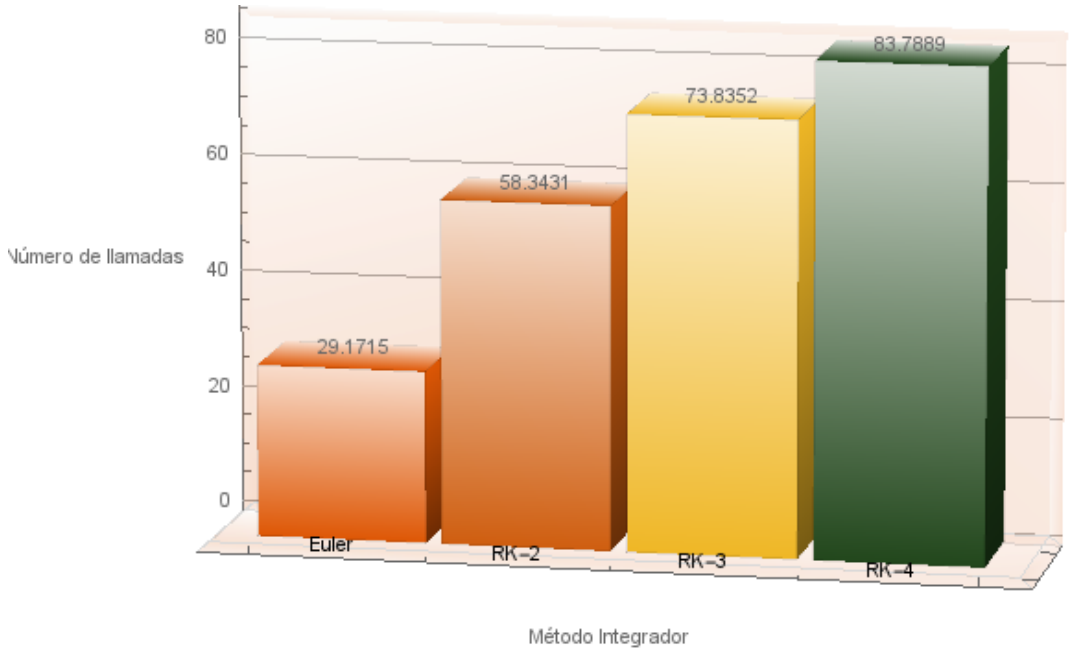

Figura 4.17: Gráfica de barras del costo computacional para cada método de integración con estructura incorrecta del modelo del *quadrotor*.

el costo computacional es muy distinto. Ahora varía entre 0 y 420. El mayor número es 420 llamadas y se da al principio, después a medida que se supera el transitorio y se alcanza el nuevo estado estacionario el número de llamadas desciende hasta oscilar entre 5 y 0.

Finalmente, para comparar el ahorro en el costo computacional que obtenemos al utilizar la estructura correcta del modelo en lugar de la incorrecta, calculamos el número total de llamadas para cada una de las simulaciones realizadas. Con la estructura correcta fueron 606 llamadas, mientras que con la incorrecta fueron $9,130$. Observemos que 601 representa el $6.58\,\%$ de $9,130$, por lo que tenemos un ahorro del $93.42\,\%$ si utilizamos la estructura correcta del modelo en lugar de la incorrecta.

4.4. Modelo de una planta termoeléctrica

Presentamos un modelo desarrollado por Usoro [39] que describe el comportamiento dinámico de una planta termoeléctrica, la cual está compuesta por unidades de generación termoélectrica. Una **Unidad de generación Termoélectrica (UT)** es capaz de transformar la energía química, contenida en combustibles fósiles, en energía eléctrica.

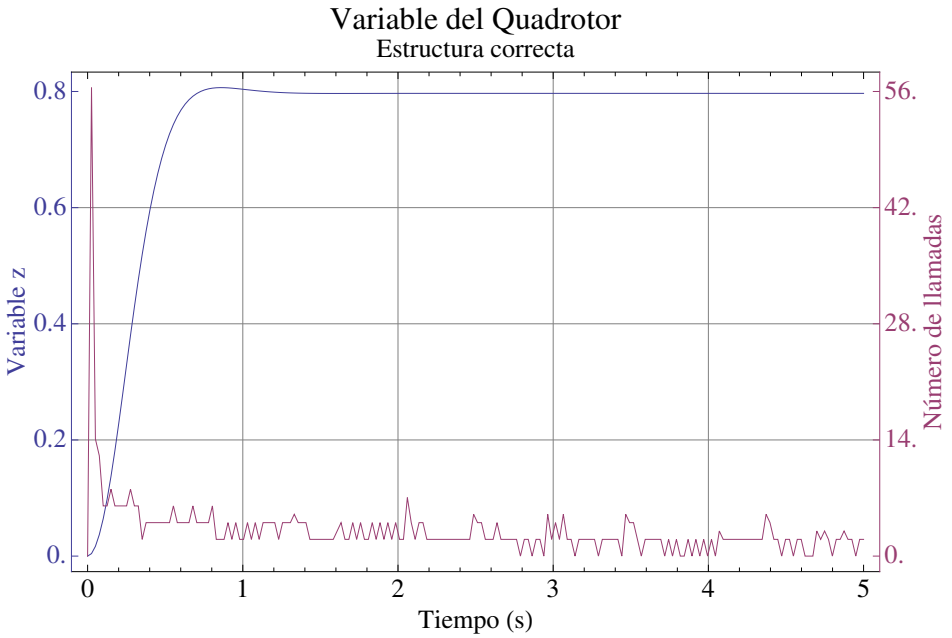

Figura 4.18: Gráfica de la variable z y el costo computacional cuando la simulación posee una estructura correcta del modelo del *quadrotor*.

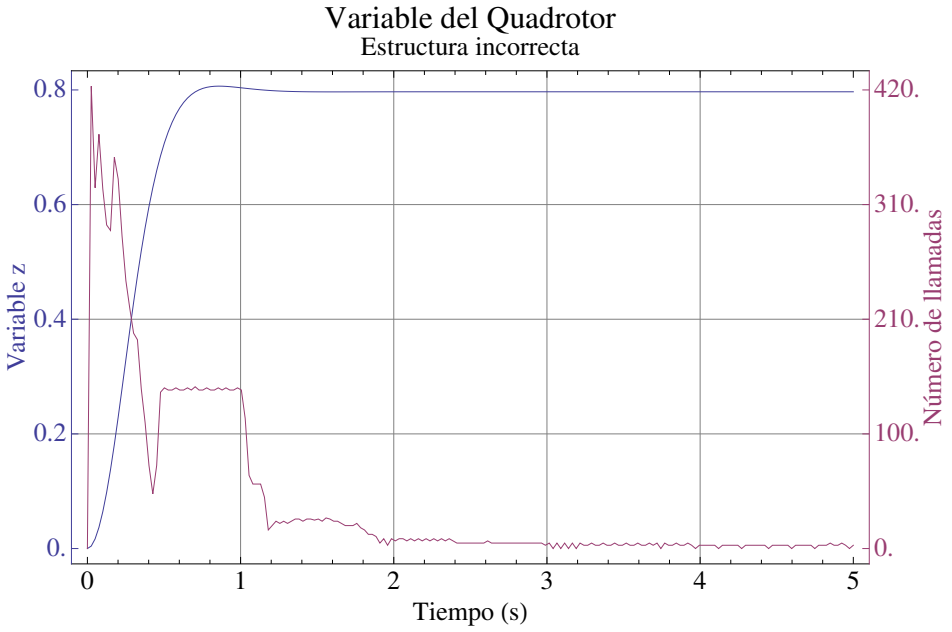

Figura 4.19: Gráfica de la variable z y el costo computacional cuando la simulación posee una estructura incorrecta del modelo del *quadrotor*.

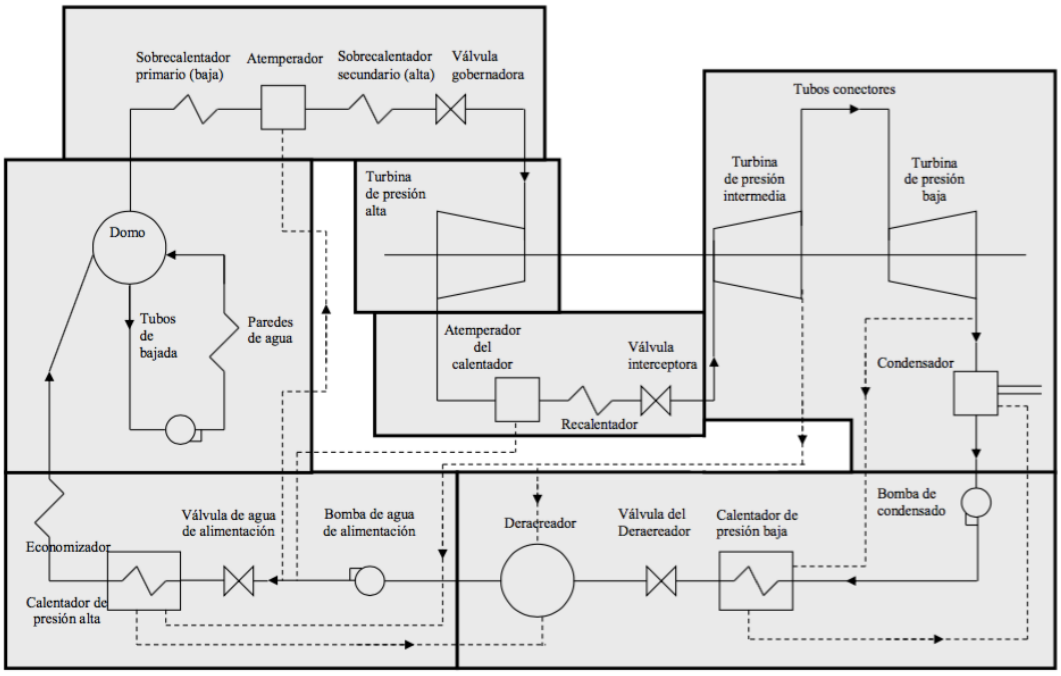

Figura 4.20: Módulos de la Unidad Termoeléctrica, imagen tomada de [10]

Este modelo contiene veintisiete variables de estado, doce de ellas describen los procesos físicos y las restantes el sistema de control. Debido al gran número de ecuaciones presentes en el modelo de la planta termoeléctrica, en este escrito no presentamos de manera explícita el modelo y, remitimos al lector interesado a las siguientes referencias [39] y [18].

El modelo matemático se encuentra dividido en partes llamadas módulos, donde cada uno representa una función específica. En la figura 4.20 se muestra un diagrama con los módulos identificados para esta **UT**. Cada módulo describe mediante un sistema de **EADs** una parte del modelo y, físicamente representa una sección de la planta como: la caldera, la turbina, el turbogenerador, el circuito de agua condensado, etc. Cabe resaltar que, el modelo matemático de la planta termoeléctrica que usamos en nuestros experimentos es lineal.

4.4.1. Resultados de la aplicación de la metodología propuesta

Presentamos los resultados de la aplicación de la metodología propuesta (ver sección 3.2) al modelo de la planta termoeléctrica de forma análoga al experimento realizado en la sección 4.1.

Primero realizamos la *prueba estructural mejorada*, para analizar si el orden de ejecución de las ecuaciones del modelo está en forma correcta, en los tiempos $t = \{0, 5, 50, 150, 200, 250\}$ pero la prueba no fue superada. Fue necesario corregir el orden de ejecución de las ecuaciones del modelo. Dada la complejidad del modelo fue necesario utilizar el algoritmo de Tarjan [37] implementado por Cruz-Tavira y Rodríguez-Gómez [10] para que la prueba estructural fuera superada satisfactoriamente. Ver el apéndice C para una explicación más detallada.

Luego, siguiendo nuestra metodología, revisamos la composición del modelo. De los resultados obtenidos concluimos que el modelo contiene 154 **EAs** lineales, además observamos que tiene 27 **EDOs** de primer orden, no lineales con valores iniciales. Por lo que, utilizamos el método integrador **ODE** de Shampine [35] que integra un sistema de ecuaciones diferenciales ordinarias a valores iniciales por medio de las fórmulas *Adams-Bashforth* y *Adams-Moulton* con control de error local, paso y orden variable.

Posteriormente, simulamos un estado estacionario,este consiste en simular $400s$ iniciando la planta al 100 % de la demanda de carga, observamos que el estacionario se mantenía, por lo que pasamos a la siguiente etapa de la metodología.

Después, elegimos un transitorio en el cual se inicia la planta al 100 % de la demanda de carga y después de un tiempo se tiene una transición de 100 % a 77.5 %, estabilizándose. Con el transitorio seleccionado, realizamos una simulación de $400(s)$, tomando muestras de los datos cada segundo. El transitorio fue superado y con ello esta etapa de la metodología.

Luego, para calcular el paso de integración más adecuado para cada método integrador de paso fijo, Euler, **R–K** 2, **R–K**3 y **R–K** 4, aproximamos la matriz jacobiana del modelo en diferentes tiempos a lo largo de la simulación. Los tiempos que elegimos fueron $t = \{0, 5, 50, 150, 200, 250\}$ (ver sección 1.8).

En seguida, a partir de tales matrices jacobianas obtuvimos el conjunto de valores característicos para cada una de ellas. Estos los mostramos graficados en la figura 4.21. En este caso, la mayoría de los valores característicos encontrados tienen parte real negativa. Excepto uno que es positivo, lo cual sucede por los errores de redondeo.

La gráfica de barras de la figura 4.22 muestra el paso de integración recomendado por cada método integrador.

El costo computacional de cada uno de estos métodos lo calculamos cada segundo y lo mostramos mediante la gráfica de barras en la figura 4.23.

De acuerdo con la metodología propuesta, el método y paso de integración recomendados son **R–K 3** y paso de integración $0.5s$, puesto que proporcionan el menor costo computacional.

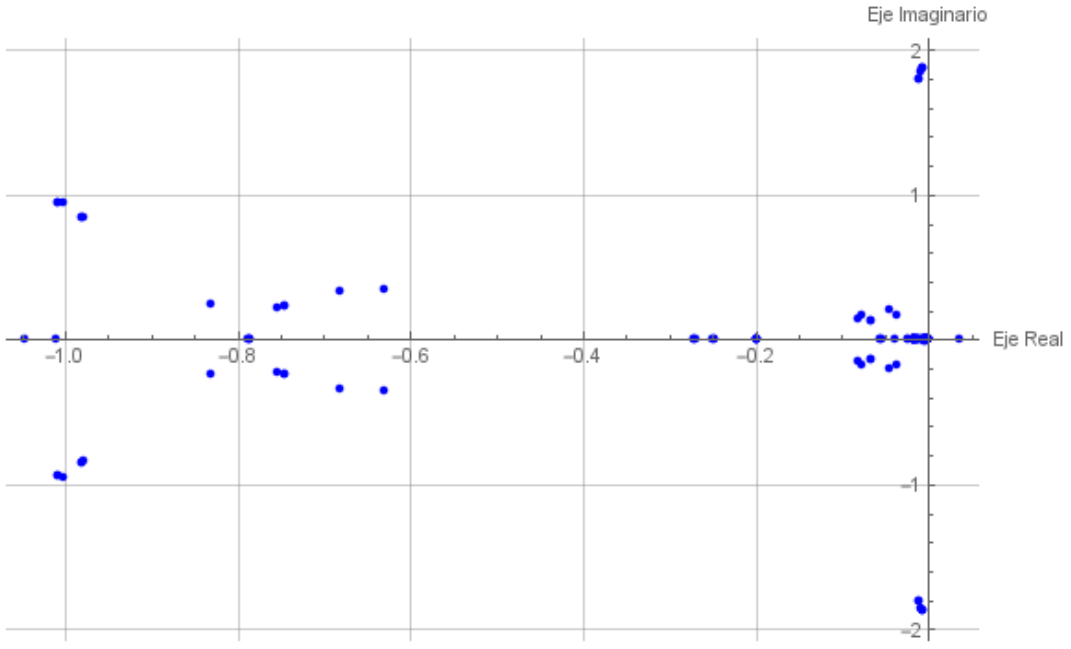

Figura 4.21: Gráfica de la distribución los valores característicos con estructura correcta del modelo de la planta termoeléctrica.

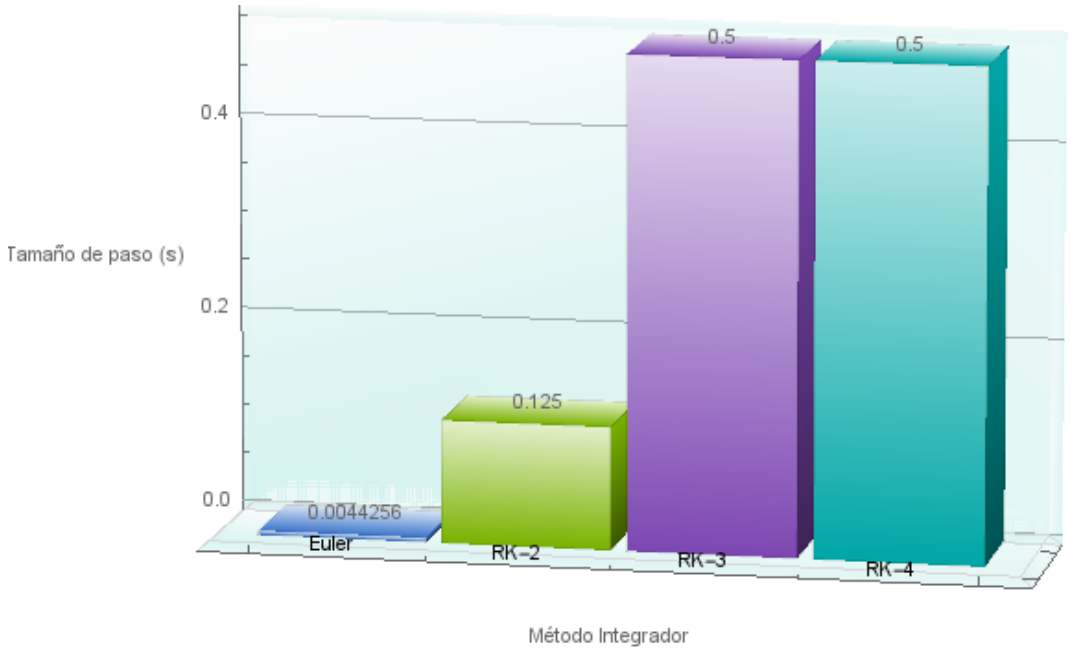

Figura 4.22: Gráfica de barras con el paso de integración recomendado para cada método integrador con estructura correcta del modelo de la planta termoeléctrica.

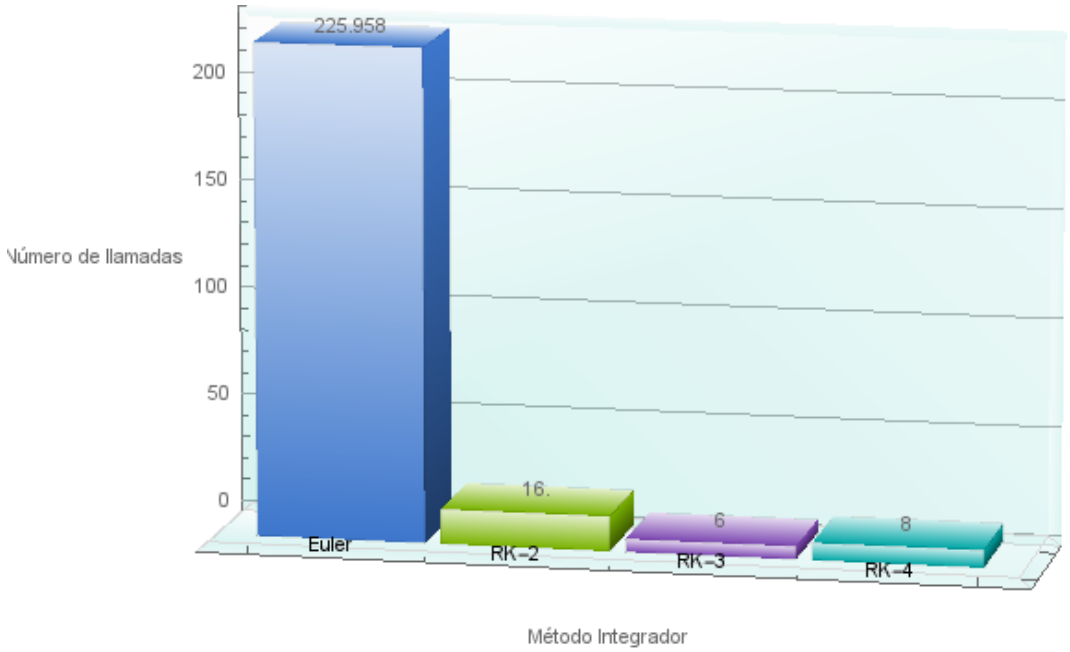

Figura 4.23: Gráfica de barras del costo computacional para cada método de integración con estructura correcta del modelo de la planta termoeléctrica.

Además, analizamos cómo la estructura incorrecta afecta la eficiencia computacional de la simulación. Para ello, hemos permitido que el *modelo de la termoeléctrica* posea una estructura incorrecta y utilizamos el método integrador, **ODE** de Shampine, con los mismos parámetros que cuando el modelo tenía una estructura correcta.

La *prueba estructural mejorada* nos arrojó los resultados esperados porque al aplicarla en los mismos tiempos que antes ($t = \{0, 5, 50, 150, 200, 250\}$), desde el tiempo $t = 0$, la prueba mostró resultados diferentes de cero, tal como a observamos en la tabla 4.7, ir el apéndice D para ver la tabla completa.

Después, simulamos el mismo estado estacionario: iniciando la planta al 100 % de la demanda de carga, el cual fue superado a pesar de que el modelo mantenía una estructura incorrecta.

Posteriormente, realizamos simulaciones con el mismo transitorio: se inicia la planta al 100 % de la demanda de carga y después de un tiempo se tiene una transición de 100 % a 77.5 %, el cual también fue superado. Sin embargo, al calcular los pasos de integración para cada método integrador obtuvimos resultados muy diferentes a los antes presentados.

Al aproximar las matrices jacobianas en los mismos tiempos que antes ($t = \{0, 5, 50, 150, 200, 250\}$), obtuvimos los valores característicos graficados en 4.24. Notamos que existen valores con parte real positiva, tales valores son los que afectan

Tiempo	Ecuación	Resultados de la prueba estructural
0	1	0.5
0	2	0.8
0	3	0.52
0	4	1.2
0	6	1.1
0	7	0.041
0	8	0.17
0	9	0.000012
0	10	0.00015
0	13	−0.74

Tabla 4.7: Resultados de la *Prueba estructural mejorada* para el tiempo t = 0 s del modelo de la planta termoeléctrica.

el tamaño del paso de integración porque no se encuentran dentro de las **REAs** de los métodos integradores candidatos, tal como lo mostramos en la sección 3.4.1.

La gráfica de barras de figura 4.25 muestra el paso de integración recomendado por cada método integrador. Observamos que los pasos de integración son más pequeños en comparación con los que obtuvimos con la estructura correcta. El costo computacional lo mostramos en la figura 4.26 mediante un gráfica de barras, observamos que los resultados obtenidos aumentan en comparación con los anteriores (ver figura 4.23).

Como mencionamos antes, el modelo de la planta poseía una estructura incorrecta y aún así era capaz de simular un transitorio con un método de integración de paso variable con control de error. A continuación, comparamos el comportamiento de la variable $QWWGM$ y el costo computacional cuando el sistema contaba con una estructura correcta y después con una estructura incorrecta.

La gráfica de la figura 4.27 presenta los resultados de la simulación del transitorio elegido con la estructura correcta. En el eje primario graficamos la variable $QWWGM$ la cual oscila entre 5×10^4 y 3×10^5, esta oscilación se amortigua y tiende al valor de 1.4×10^5 que es el valor deseado, en el eje secundario se grafica el costo computacional por cada segundo. El rango de valores del costo computacional varía entre 0 y 20 llamadas, la parte más costosa se da al inicio de la simulación con 20 llamadas, después de que transcurren los primeros 100s de simulación, el número de llamadas varía entre 7 y 2.

Por otro lado, la gráfica de la figura 4.28 muestra los resultados obtenidos al

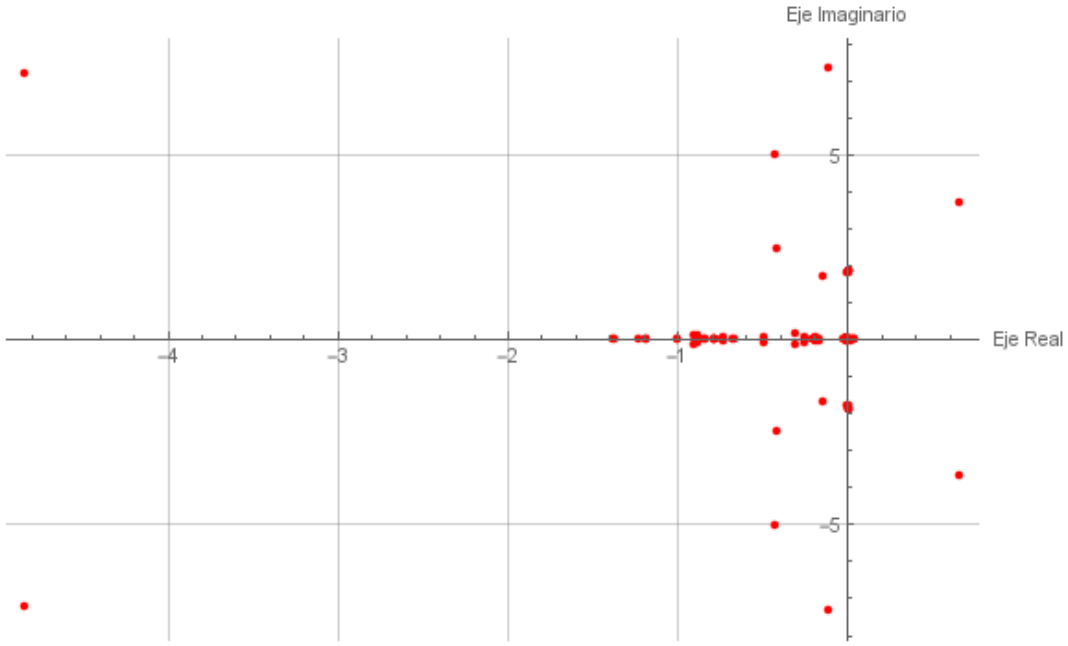

Figura 4.24: Gráfica de la distribución de los valores característicos con estructura incorrecta del modelo de la planta termoeléctrica.

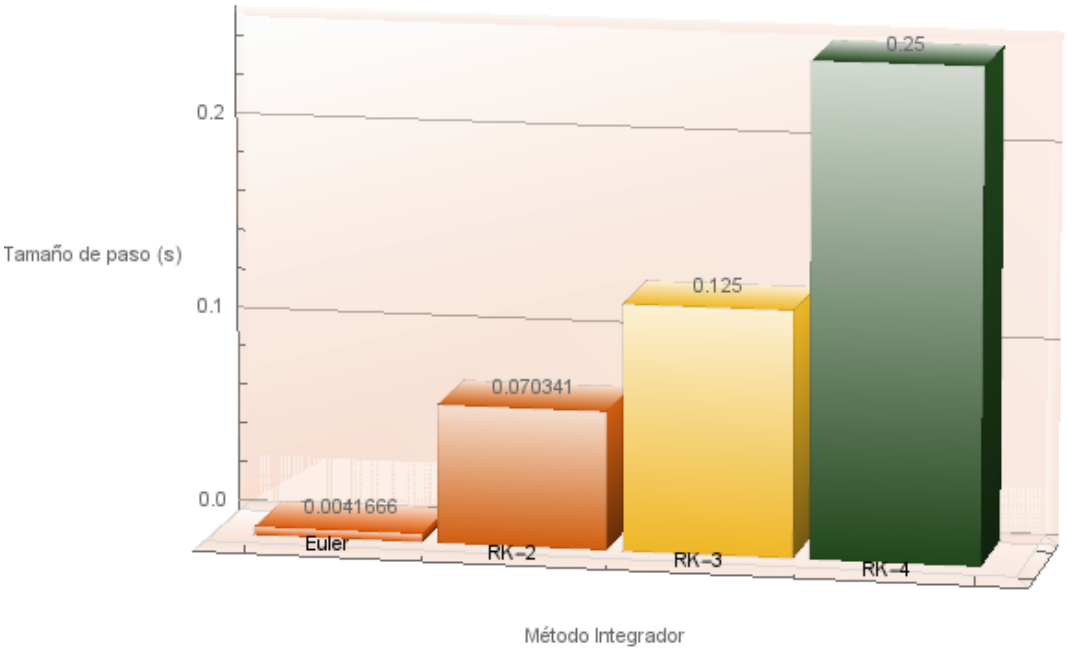

Figura 4.25: Gráfica de barras con el paso de integración recomendado para cada método integrador con estructura incorrecta del modelo de la planta termoeléctrica.

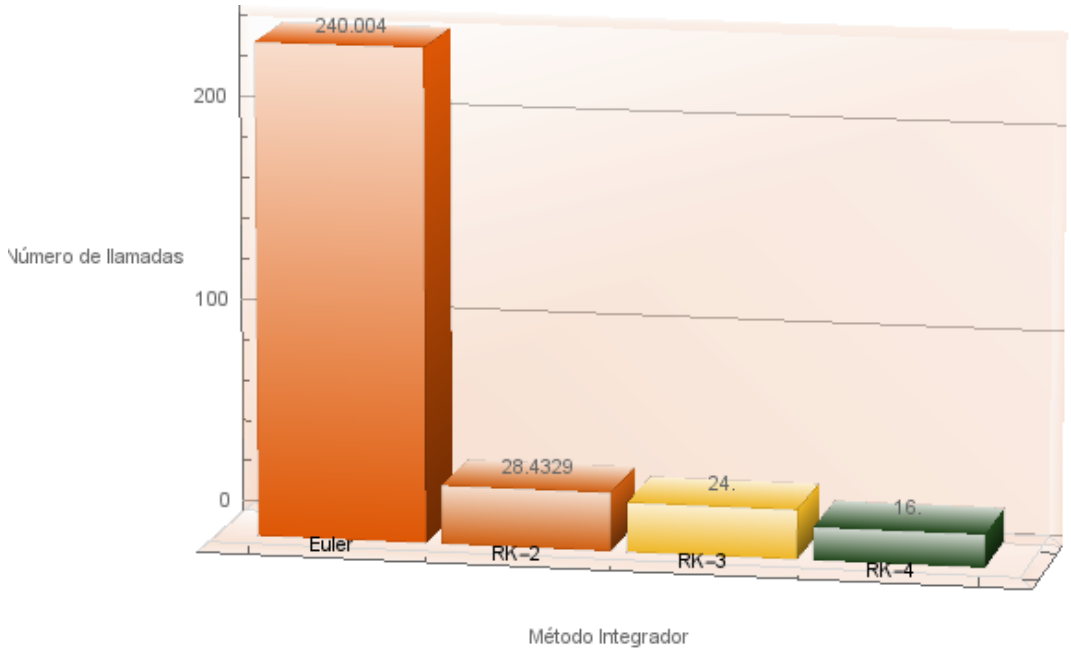

Figura 4.26: Gráfica de barras del costo computacional para cada método de integración con estructura incorrecta del modelo de la planta termoeléctrica.

simular el mismo transitorio pero con la estructura incorrecta. Observamos que el comportamiento de la variable $QWWGM$ es el mismo que en la gráfica anterior 4.27; sin embargo, el costo computacional es muy distinto. Ahora varía entre 0 y 100. El mayor número de llamadas es 100 y se da al principio después a medida que se supera el transitorio y se alcanza el nuevo estado estacionario el número de llamadas oscila entre 8 y 2.

Finalmente, para comparar el ahorro en el costo computacional que obtenemos al utilizar la estructura correcta del modelo en lugar de la incorrecta, calculamos el número total de llamadas para cada una de las simulaciones realizadas. Con la estructura correcta fueron $1,884$ llamadas, mientras que con la incorrecta fueron $4,915$. Observemos que $1,884$ representa el $38.33\,\%$ de $4,915$, por lo que tenemos un ahorro del $61.67\,\%$ si utilizamos la estructura correcta del modelo en lugar de la incorrecta.

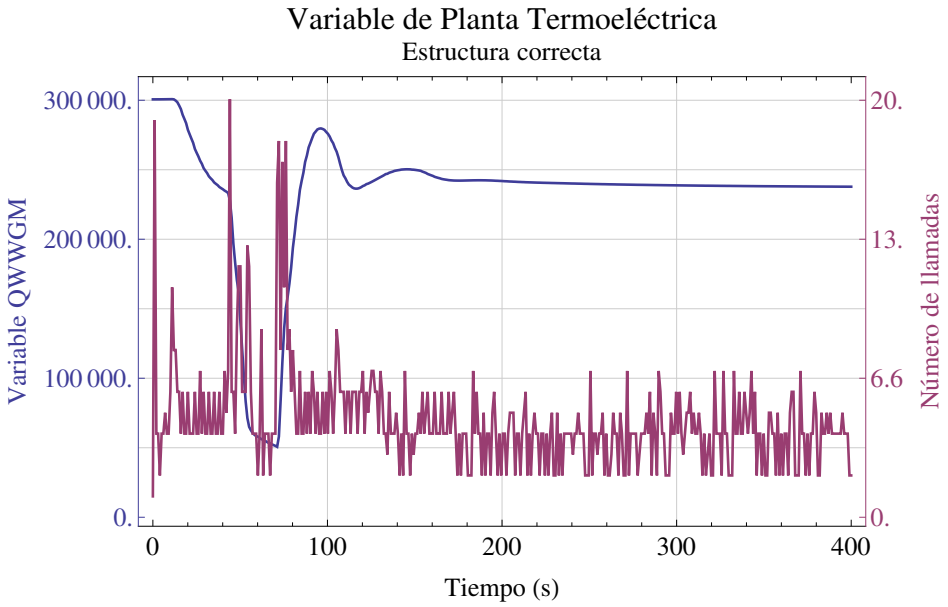

Figura 4.27: Gráfica de la variable $QWWGM$ y el costo computacional cuando la simulación posee una estructura correcta del modelo de la planta termoeléctrica.

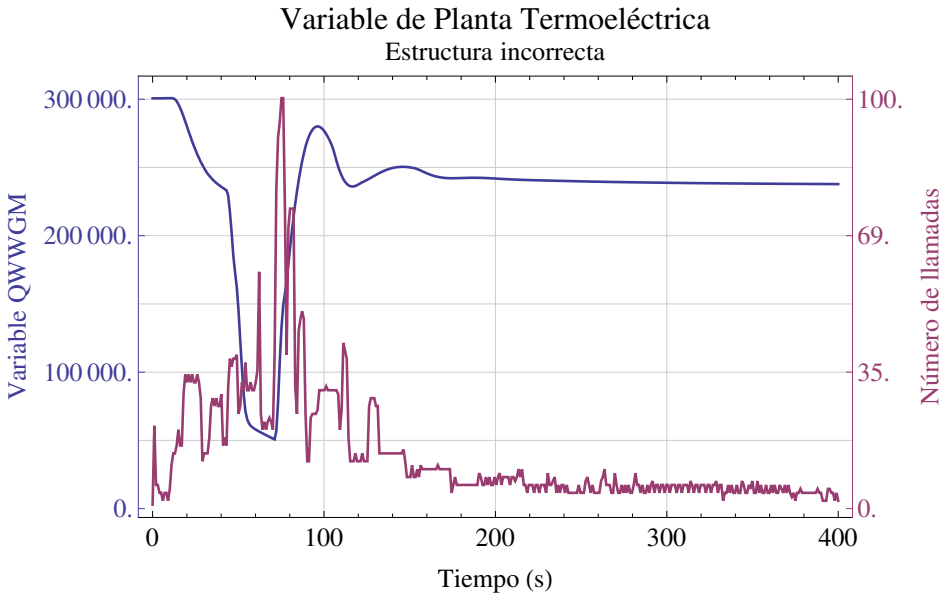

Figura 4.28: Gráfica de la variable $QWWGM$ y el costo computacional cuando la simulación posee una estructura incorrecta del modelo de la planta termoeléctrica.

Capítulo 5

Conclusiones

En este proyecto propusimos una metodología para la validación y selección de algoritmos numéricos para modelos de simulación de procesos dinámicos. Además, verificamos la simulación con la correcta estructura del modelo computacional.

También, proporcionamos el *algoritmo estructural mejorado* que indica si el orden de las ecuaciones del sistema es el correcto, la cual no es computacionalmente costosa. Además, proporcionamos un planteamiento teórico y una solución al problema de las **EDOs** que se encuentran limitadas debido a las restricciones físicas del proceso real.

El objetivo presentado la introducción se cumplió satisfactoriamente porque la metodología propuesta permite realizar simulaciones más eficientes puesto que seleccionamos el método con el paso de integración más grande con un error aceptable en cada uno de los casos analizados. Además, en éstos últimos cuando existe una estructura incorrecta, se puede observar que el costo computacional es mucho mayor que con la estructura correcta.

Otro hecho a tomar en cuenta es que con una estructura incorrecta los valores característicos no reflejan la dinámica del sistema, puesto que se rompe tal como observamos en los experimentos realizados.

En cuanto al trabajo futuro tenemos dos posible direcciones. La primera es analizar el problema de los limitadores y la solución planteada en este escrito. Y la segunda toma en cuenta que en los casos estudiados pudimos advertir que los valores característicos se agrupan en conjuntos disjuntos, esto sucede principalmente cuando

existe una estructura correcta. Lo anterior significa que la planta física tiene módulos con dinámicas diferentes. Así que los métodos de integración multitasa podrían explotar esta situación. Los métodos de integración multitasa son técnicas que utilizan diferentes tamaños de paso con esquemas de integración diferentes o iguales para aproximar los componentes del sistema de acuerdo a su dinámica, para más detalles ver [14, 29].

Apéndice A

Cota superior para los métodos R–K

En este apéndice se presenta el desarrollo completo de la estimación de una cota superior para el paso de integración de los métodos Runge–Kutta, tal como se presentó en la sección 3.4.2 para el método de integración de Euler.

A.1. Cota superior para los métodos R–K de orden dos

Consideremos el polinomio de estabilidad para los métodos Runge–Kutta de orden dos

$$p_2(\overline{h}) = 1 + \overline{h} + \frac{1}{2}\overline{h}^2,$$

donde $\overline{h} = \lambda h$, y $\lambda = x + iy$, con $i = \sqrt{-1}$, $x < 0$ y $h > 0$. El polinomo de estabilidad está definido de la siguiente forma, de acuerdo con Lambert [21]:

$$p_2(\overline{h}) = 1 + h(x + iy) + \frac{1}{2}h^2(x + iy)^2.$$

La región de estabilidad absoluta del método numérico se encuentra en el plano complejo $\mathbb{C}$ y está dada por

$$\mathcal{R}_2 = \left\{ \overline{h} = \lambda h : p(\overline{h}) < 1 \right\}.$$

La región de estabilidad absoluta se determina por medio de su frontera, esto es, determinamos las $\overline{h}$ que satisfacen

$$p_2(\overline{h})^2 - 1 = 0. \tag{A.1.1}$$

Recordemos que $|z|^2 = z \cdot \overline{z}$. Entonces desarrollando A.1.1 se tiene que

$$h\left(2x + 2hx^2 + h^2x(x^2+y^2) + \frac{1}{4}h^3(x^2+y^2)^2\right) = 0.$$

Observamos que el polinomio tiene una raíz en $h = 0$, por lo cual el polinomio para encontrar H, la cota superior, se simplifica a la siguiente expresión

$$p_2(x,y) = 2x + 2hx^2 + h^2x(x^2+y^2) + \frac{1}{4}h^3(x^2+y^2)^2.$$

Por medio del polinomio $p_2(x,y)$ dado $\lambda = x + iy$ se determina la cota superior H para seleccionar el paso de integración para que el paso de integración cumpla $0 < h < H$ y garantizar que el polinomio de estabilidad del método de Euler satisface

$$\left|1 + \overline{h} + \frac{1}{2}\overline{h}^2\right| < 1.$$

A.2. Cota superior para los métodos R–K de orden tres

Consideremos el polinomio de estabilidad para los métodos Runge–Kutta de orden tres

$$p_3(\overline{h}) = 1 + \overline{h} + \frac{1}{2}\overline{h}^2 + \frac{1}{6}\overline{h}^3,$$

donde $\overline{h} = \lambda h$, y $\lambda = x + iy$, con $i = \sqrt{-1}$, $x < 0$ y $h > 0$. El polinomo de estabilidad está definido de la siguiente forma, ver Lambert [21]

$$p_3(\overline{h}) = 1 + h(x+iy) + \frac{1}{2}h^2(x+iy)^2 + \frac{1}{6}h^3(x+iy)^3.$$

La región de estabilidad absoluta del método numérico se encuentra en el plano complejo $\mathbb{C}$ y está dada por

$$\mathcal{R}_3 = \left\{ \overline{h} = \lambda h : p(\overline{h}) < 1 \right\}.$$

La región de estabilidad absoluta se determina por medio de su frontera, esto es, determinamos las $\overline{h}$ que satisfacen

$$p_3(\overline{h})^2 - 1 = 0. \tag{A.2.1}$$

Recordemos que $|z|^2 = z \cdot \overline{z}$. Entonces desarrollando A.2.1 tenemos que

$$\begin{aligned} &h^6 \left(\frac{x^6}{36} + \frac{x^4y^2}{12} + \frac{x^2y^4}{12} + \frac{y^6}{36} \right) + h^5 \left(\frac{x^5}{6} + \frac{x^3y^2}{3} + \frac{xy^4}{6} \right) + \\ &h^4 \left(\frac{7x^4}{12} + \frac{x^2y^2}{2} - \frac{y^4}{12} \right) + \frac{4h^3x^3}{3} + 2h^2x^2 + 2hx \\ &= 0. \end{aligned}$$

Observamos que el polinomio tiene una raíz en $h = 0$, por lo cual el polinomio para encontrar H, la cota superior, se simplifica a la siguiente expresión

$$\begin{aligned} p_3(x,y) = h^5 &\left(\frac{x^6}{36} + \frac{x^4y^2}{12} + \frac{x^2y^4}{12} + \frac{y^6}{36} \right) + h^4 \left(\frac{x^5}{6} + \frac{x^3y^2}{3} + \frac{xy^4}{6} \right) \\ &+ h^3 \left(\frac{7x^4}{12} + \frac{x^2y^2}{2} - \frac{y^4}{12} \right) + \frac{4h^2x^3}{3} + 2hx^2 + 2x. \end{aligned}$$

Por medio del polinomio $p_3(x,y)$ dado $\lambda = x + iy$ se determina la cota superior H para seleccionar el paso de integración para que el paso de integración cumpla $0 < h < H$ y garantizar que el polinomio de estabilidad del método de Euler satisface

$$\left| 1 + \overline{h} + \frac{1}{2}\overline{h}^2 + \frac{1}{6}\overline{h}^3 \right| < 1.$$

A.3. Cota superior para los métodos R–K de orden cuatro

Consideremos el polinomio de estabilidad para los métodos Runge–Kutta de orden cuatro

$$p_4(\overline{h}) = 1 + \overline{h} + \frac{1}{2}\overline{h}^2 + \frac{1}{6}\overline{h}^3 + \frac{1}{24}\overline{h}^4,$$

donde $\overline{h} = \lambda h$, y $\lambda = x + iy$, con $i = \sqrt{-1}$, $x < 0$ y $h > 0$. El polinomo de estabilidad está definido de la siguiente forma, ver Lambert [21]:

$$p_4(\overline{h}) = 1 + h(x+iy) + \frac{1}{2}h^2(x+iy)^2 + \frac{1}{6}h^3(x+iy)^3 + \frac{1}{24}h^4(x+iy)^4.$$

La región de estabilidad absoluta del método numérico se encuentra en el plano complejo $\mathbb{C}$ y está dada por

$$\mathcal{R}_4 = \left\{\overline{h} = \lambda h : p(\overline{h}) < 1\right\}.$$

La región de estabilidad absoluta se determina por medio de su frontera, esto es, determinamos las $\overline{h}$ que satisfacen

$$p_4(\overline{h})^2 - 1 = 0. \tag{A.3.1}$$

Recordemos que $|z|^2 = z \cdot \overline{z}$. Entonces desarrollando A.3.1 tenemos que

$$\begin{aligned}
&h^8\left(\frac{x^8}{576} + \frac{x^6y^2}{144} + \frac{x^4y^4}{96} + \frac{x^2y^6}{144} + \frac{y^8}{576}\right) + h^7\left(\frac{x^7}{72} + \frac{x^5y^2}{24} + \frac{x^3y^4}{24} + \frac{xy^6}{72}\right) + \\
&h^6\left(\frac{5x^6}{72} + \frac{x^4y^2}{8} + \frac{x^2y^4}{24} - \frac{y^6}{72}\right) + h^5\left(\frac{x^5}{4} + \frac{x^3y^2}{6} - \frac{xy^4}{12}\right) + \\
&\frac{2h^4x^4}{3} + \frac{4h^3x^3}{3} + 2h^2x^2 + 2hx \\
&= 0.
\end{aligned}$$

Observamos que el polinomio tiene una raíz en $h = 0$, por lo cual el polinomio para encontrar H, la cota superior, se simplifica a la siguiente expresión

$$\begin{aligned}
p_4(x,y) = {} & h^7\left(\frac{x^8}{576} + \frac{x^6y^2}{144} + \frac{x^4y^4}{96} + \frac{x^2y^6}{144} + \frac{y^8}{576}\right) + h^6\left(\frac{x^7}{72} + \frac{x^5y^2}{24} + \frac{x^3y^4}{24} + \frac{xy^6}{72}\right) \\
& + h^5\left(\frac{5x^6}{72} + \frac{x^4y^2}{8} + \frac{x^2y^4}{24} - \frac{y^6}{72}\right) + h^4\left(\frac{x^5}{4} + \frac{x^3y^2}{6} - \frac{xy^4}{12}\right) + \frac{2h^3x^4}{3} + \frac{4h^2x^3}{3} \\
& + 2hx^2 + 2x.
\end{aligned}$$

Por medio del polinomio $p_4(x,y)$ dado $\lambda = x + iy$ se determina la cota superior H para seleccionar el paso de integración para que el paso de integración cumpla $0 < h < H$ y garantizar que el polinomio de estabilidad del método de Runge–Kutta 4 satisface

$$\left|1 + \overline{h} + \frac{1}{2}\overline{h}^2 + \frac{1}{6}\overline{h}^3 + \frac{1}{24}\overline{h}^4\right| < 1.$$

Apéndice B

Condiciones de simulación

A continuación presentamos las características de *hardware* y *software* que utilizamos para llevar a cabo los experimentos en este trabajo.

Las pruebas las realizamos en una computadora personal de 1.6 GHz con 3.18 GB de RAM con un procesador Intel Core i5.

El *software* que utilizamos fue *Code::Blocks* versión 16.01, revisión 10702. El compilador fue *GNU Fortran Compiler* de *GCC* versión 6.2.0.

Apéndice C

Funcionamiento del algoritmo y software de ordenamiento

Tal como mencionamos en la sección 4.4.1 del modelo de la termoeléctrica, fue necesario utilizar el algoritmo de Tarjan [37] para que la prueba estructural fuese superada satisfactoriamente. En este apéndice mostramos un ejemplo de la entrada y salida de la implementación de tal algoritmo presentado por Cruz-Tavira y Rodríguez-Gómez en [10].

Para utilizar el software de [10] se introduce una matriz de dependencia de variables, se crea una matriz de incidencia interna, y se obtiene de salida una matriz de incidencia, la cual muestra la estructura correcta de las ecuaciones.

La figura C.1 muestra la matriz de incidencia que se forma internamente. Observemos como en la parte superior de la matriz encontramos elementos (números *uno*) en repetidas ocasiones, esto indica que el modelo tiene una estructura incorrecta. Un ejemplo de ecuación mal ordenada es la ecuación 3 que depende la variable 81, cuyo valor en este caso aun no está calculado. Otros ejemplos son la ecuación 5, la no. 12, etc.

La figura C.2 muestra la matriz de incidencia de salida del algoritmo, observemos como ya no se encuentran elementos por arriba de la diagonal principal. Puesto que algoritmo de ordenamiento mediante permutaciones de la matriz de incidencia interna encontró la estructura correcta del modelo de la termoeléctrica.

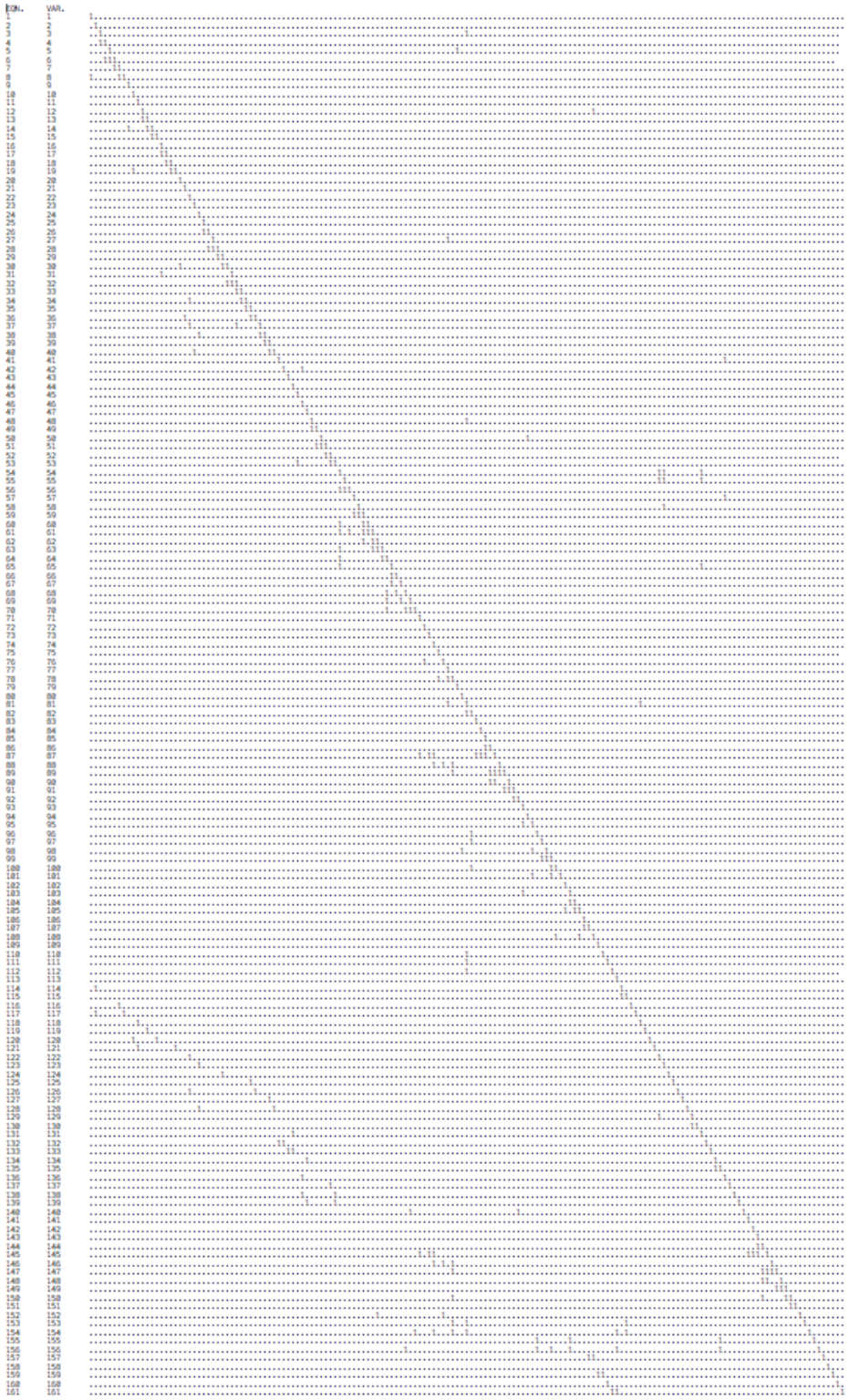

Figura C.1: Matriz de incidencia formada por el algoritmo de ordenamiento de la Unidad Termoeléctrica.

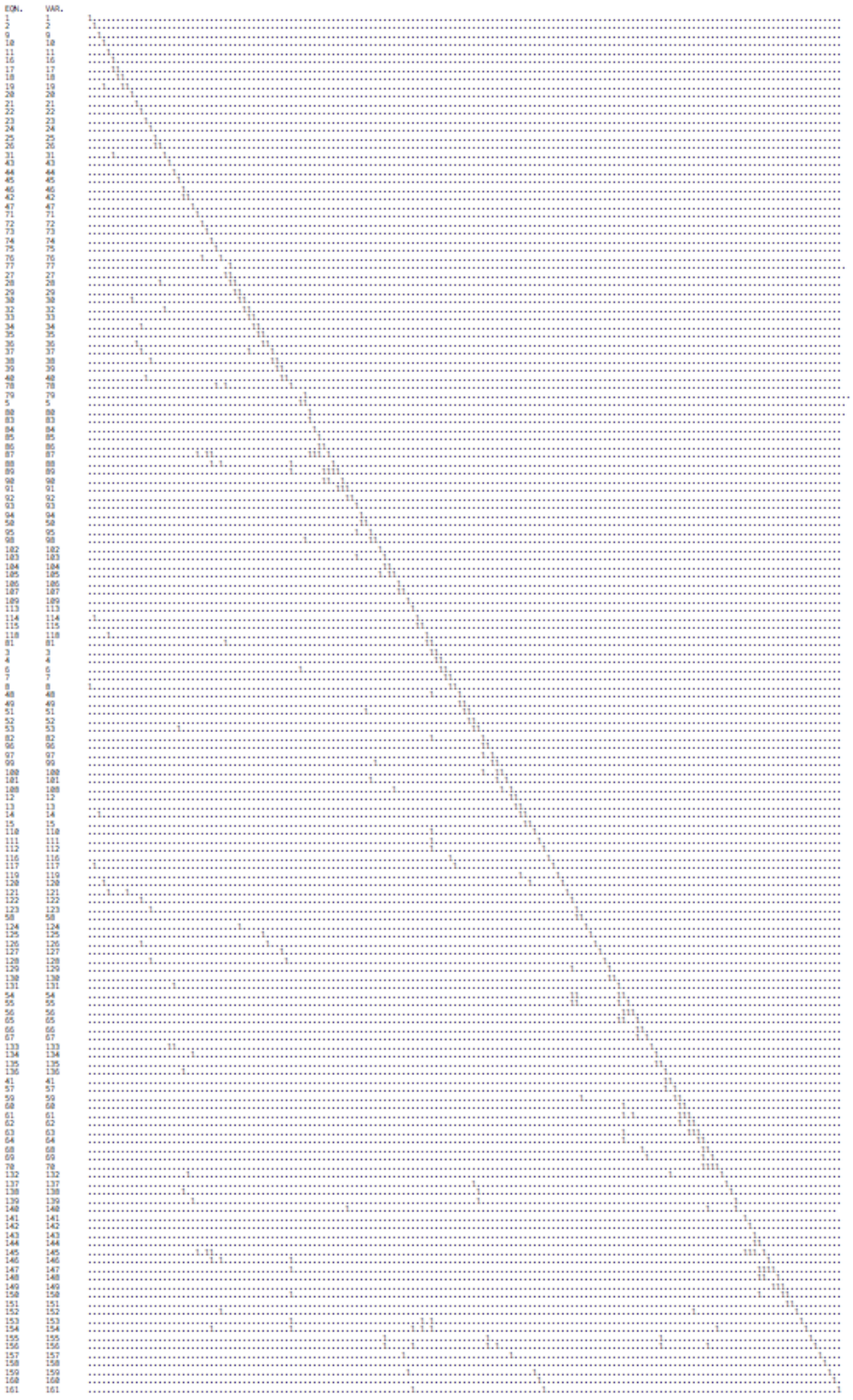

Figura C.2: Matriz de incidencia de salida del algoritmo de ordenamiento de la Unidad Termoeléctrica.

Apéndice D

Tabla de la prueba de orden del modelo de la planta termoeléctrica

A continuación se presenta la tabla completa de los resultados de la prueba de estructural del experimento 4.4.

Tiempo	Ecuación	Resultados de la prueba estructural
0	1	0.5
0	2	0.8
0	3	0.52
0	4	1.2
0	6	1.1
0	7	0.041
0	8	0.17
0	9	0.000012
0	10	0.00015
0	13	-0.74
5	3	0.52×10^{-6}
5	4	0.52×10^{-5}
5	6	0.48×10^{-6}
5	7	0.14×10^{-5}
5	8	0.17×10^{-4}
5	13	-0.19×10^{-6}
5	16	0.11×10^{-4}
5	18	0.52×10^{-5}

Continua en la página siguiente.

Tiempo	Ecuación	Resultados de la prueba estructural
5	19	0.61×10^{-7}
5	20	-0.19×10^{-6}
5	21	-0.51×10^{-7}
5	22	0.82×10^{-5}
5	23	-0.54×10^{-6}
5	25	-0.25×10^{-6}
5	26	-0.14×10^{-7}
5	27	-0.25×10^{-6}
50	3	0.23×10^{-5}
50	4	0.23×10^{-4}
50	6	0.21×10^{-4}
50	9	0.62×10^{-5}
50	10	0.74×10^{-4}
50	13	0.22×10^{-5}
50	16	-0.78×10^{-4}
50	18	-0.14×10^{-3}
50	19	0.48×10^{-6}
50	20	-0.99×10^{-4}
50	21	-0.35×10^{-6}
50	22	-0.13×10^{-3}
50	23	-0.37×10^{-5}
50	25	-0.13×10^{-5}
50	26	-0.12×10^{-6}
50	27	-0.23×10^{-5}
150	1	-0.19×10^{-4}
150	3	-0.74×10^{-5}
150	4	-0.74×10^{-4}
150	6	-0.77×10^{-5}
150	7	-0.23×10^{-4}
150	8	-0.28×10^{-3}
150	13	-0.63×10^{-5}
150	16	-0.12×10^{-4}
150	18	-0.46×10^{-4}
150	19	0.42×10^{-6}
150	20	-0.19×10^{-3}
150	21	-0.31×10^{-6}
150	22	-0.11×10^{-3}
150	23	-0.30×10^{-5}
150	25	-0.10×10^{-5}
150	26	-0.11×10^{-6}
150	27	-0.20×10^{-5}
200	1	-0.30×10^{-4}
200	3	-0.50×10^{-5}

Continua en la página siguiente.

Tiempo	Ecuación	Resultados de la prueba estructural
200	4	-0.51×10^{-4}
200	6	-0.49×10^{-5}
200	7	-0.15×10^{-4}
200	8	-0.18×10^{-3}
200	9	-0.15×10^{-4}
200	10	-0.18×10^{-3}
200	13	-0.12×10^{-4}
200	16	-0.23×10^{-4}
200	18	-0.11×10^{-3}
200	19	0.35×10^{-5}
200	20	-0.15×10^{-3}
200	21	-0.26×10^{-5}
200	22	-0.30×10^{-3}
200	23	-0.25×10^{-4}
200	25	-0.84×10^{-5}
200	26	-0.93×10^{-6}
200	27	-0.16×10^{-4}
250	1	-0.69×10^{-5}
250	3	-0.18×10^{-5}
250	4	-0.18×10^{-4}
250	6	-0.79×10^{-7}
250	7	-0.24×10^{-6}
250	8	-0.29×10^{-5}
250	9	-0.24×10^{-6}
250	10	-0.29×10^{-5}
250	13	-0.55×10^{-5}
250	16	-0.50×10^{-5}
250	18	-0.32×10^{-4}
250	19	0.10×10^{-5}
250	20	-0.44×10^{-4}
250	21	-0.77×10^{-6}
250	22	-0.89×10^{-4}
250	23	-0.73×10^{-5}
250	25	-0.25×10^{-5}
250	26	-0.27×10^{-6}
250	27	-0.48×10^{-5}

Tabla D.1: Resultados de la *Prueba estructural mejorada* del modelo de la planta termoeléctrica.

Bibliografía

[1] C. Addison, W. H. Enright, P. Gaffney, I. Gladwell, and P. Hanson. Algorithm 687: a decision tree for the numerical solution of initial value ordinary differential equations. *ACM Transactions on Mathematical Software (TOMS)*, 17(1):1–10, 1991.

[2] C. Andersson, C. Führer, and J. Åkesson. Assimulo: A unified framework for ode solvers. *Mathematics and Computers in Simulation*, 116:26–43, 2015.

[3] J. H. Avila. Multi-rate predictor-corrector methods. 1981.

[4] K. E. Brenan, S. L. Campbell, and L. R. Petzold. *The Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations.* SIAM Series, Classics In Applied Mathematics., 1996.

[5] S. Browne, J. Dongarra, E. Grosse, and T. Rowan. The netlib mathematical software repository. *D-lib Magazine*, 1(9), 1995.

[6] P. Bunus. A simulation and decision framework for selection of numerical solvers in scientific computing. In *Simulation Symposium, 2006. 39th Annual*, pages 10–pp. IEEE, 2006.

[7] P. Bunus and P. Fritzson. Automated static analysis of equation-based components. *Simulation*, 80(7-8):321–345, 2004.

[8] E. Carpanzano and C. Maffezzoni. Symbolic manipulation techniques for model simplification in object-oriented modelling of large scale continuous systems. *Mathematics and Computers in Simulation*, 48(2):133–150, 1998.

[9] J. A. Cruz-Tavira and G. Rodríguez-Gómez. Ordenamiento orientado objetos de ecuaciones algebraico-diferenciales. *Cuarto Encuentro de Investigación, INAOE 2003.*, 2003.

[10] J. A. Cruz-Tavira and G. Rodríguez-Gómez. ¿es posible mejorar la eficiencia de cómputo de un modelo ordenando sus ecuaciones? *XII International Congress on Computing, CIC, IPN*, 2004.

[11] I. S. Duff and J. K. Reid. An implementation of tarjan's algorithm for the block triangularization of a matrix. *ACM Transactions on Mathematical Software (TOMS)*, 4(2):137–147, 1978.

[12] P. Fritzson and V. Engelson. Modelica—a unified object-oriented language for system modeling and simulation. In *European Conference on Object-Oriented Programming*, pages 67–90. Springer, 1998.

[13] F. V. Gallardo. Implantacion de la formula bdf de orden 1, a paso fijo sin control de error, para simulacion de modelos matematicos de procesos de plantas de potencia en tiempo real., 2002.

[14] C. Gear. Simulations: Conflicts between real-time and software. *Mathematical Software III. John R. Rice. Academic Press*, pages 121–138, 1977.

[15] E. Grosse. Repository mirroring. *ACM Transactions on Mathematical Software (TOMS)*, 21(1):89–97, 1995.

[16] F. Harary. A graph theoretic method for the complete reduction of a matrix with a view toward finding its eigenvalues. *Studies in Applied Mathematics*, 38(1-4):104–111, 1959.

[17] P. Henrici. *Discrete Variable Methods in Ordinary Differential Equations*. John Wiley and Sons, 1962.

[18] E. Jimenez-Magallon. Simulación de una planta termoeléctrica bao un sistema operativo unix. Master's thesis, Instituto Tecnológico y de Estudios Superiores de Monterrey, 1995.

[19] M. S. Kamel and K. S. Ma. Odexpert, an expert system to select numerical solvers for initial value ode systems. *ACM Transactions on Mathematical Software*, 19(1):44–62, March 1993.

[20] A. Kecskemethy and M. Hiller. Object-oriented programming techniques in vehicle dynamics simulation. *Mathematics and Computers in Simulation*, 39:549–558, 1995.

[21] J. D. Lambert. *Computational Methods in Ordinary Differential Equations*. John Wiley and Sons, 1973.

[22] M. Lucks and I. Gladwell. Automated selection of mathematical. *ACM Transactions on Mathematical Software*, 18(1), March 1992.

[23] M. M. Martín. *Introduction to software for chemical engineers*. CRC Press., 2014.

[24] S. E. Mattsson. Simulation of object-oriented continuous time models. *Mathematics and Computers in Simulation*, 39:513–518, 1995.

[25] M. Metcalt. *Fortran 90/95 Explained*. Oxford University Press, 1996.

[26] J. A. B. Montevechi, T. F. Pereira, C. E. S. da Silva, R. d. C. Miranda, and A. P. G. Scheidegger. IDENTIFICATION OF THE MAIN METHODS USED IN SIMULATION PROJECTS. In *2015 WINTER SIMULATION CONFERENCE (WSC)*, Winter Simulation Conference Proceedings, pages 3469–3480, 2015. Winter Simulation Conference (WSC), Huntington Beach, CA, DEC 06-09, 2015.

[27] D. Murray-Smith. Enhanced environments for the development and validation of dynamic system models. *Mathematics and Computers in Simulation*, 39:459–464, 1995.

[28] S. Robinson. *Simulation: The Practice of Model Development and Use.* John Wiley and Sons Ltd., England, 2004.

[29] G. Rodríguez-Gómez, P. González-Casanova, and J. Martínez-Carballido. Computing general companion matrices and stability regions of multirate methods. *International journal for numerical methods in engineering*, 61(2):255–273, 2004.

[30] G. Rodríguez-Gómez. Análisis de técnicas numéricas para tiempo-real. Technical report, Instituto de Investigaciones Eléctricas, 1995.

[31] G. Rodríguez-Gómez. Implantación y validación del método resolvedor de sistemas de ecuaciones diferenciales algebraicas (edas) a paso fijo para simulación en tiempo real. Technical report, Instituto de Investigaciones Eléctricas, Septiembre 1995.

[32] G. Rodríguez-Gómez. Métodos de integración multitasa para simulación de procesos en tiempo real. Master's thesis, UNAM, Facultad de Ciencias., México, D.F, 1998.

[33] A. Rukgauer and W. Schiehlen. Simulation of modular dynamic systems. *Mathematics and Computers in Simulation*, 46:535–542, 1998.

[34] R. W. H. Sargent and A. W. Westerberg. "speed-upïn chemical engineering design. *Trans. Inst. Chem. Engrs.*, (42):190–197, 1964.

[35] L. F. Shampine, R. C. Allen, and S. Pruess. *Fundamentals of numerical computing (Vol. 1).* New York: Wiley, 1997.

[36] D. V. Steward. Partitioning and tearing systems of equatios. *J. SIAM Numer Anal.*, 2(2):345–365, 1965.

[37] R. E. Tarjan. Depth first search and linear graph algorithms. *SIAM J.*, (Comptg. 1):146–160, 1972.

[38] R. E. Tarjan. Efficiency of a good but not linear set union algorithm. *J. ACM.*, 2(22):215–225, 1975.

[39] P. B. Usoro. Modelling and simulation of a drum boiler-turbine power plant under emergency state control. Master's thesis, Massachusetts Institute of Technology, 1977.

[40] J. Zhu, E. Liu, S. Guo, and C. Xu. A gradient optimization based pid tuning approach on quadrotor. In *Control and Decision Conference (CCDC), 2015 27th Chinese*, pages 1588–1593. IEEE, 2015.