UNIVERSIDAD MICHOACANA DE SAN NICOLÁS DE HIDALGO

FACULTAD DE INGENIERÍA ELÉCTRICA

DIVISIÓN DE ESTUDIOS DE POSGRADO

DYNAMICAM: SOFTWARE PARA ANÁLISIS DE SISTEMAS DINÁMICOS

TESIS

Que para obtener el grado de

MAESTRO EN CIENCIAS EN INGENIERÍA ELÉCTRICA

Presenta

Iván Jonathan Aceves Adán

Dr. Juan José Flores Romero

Director de Tesis

Dr. Claudio Rubén Fuerte Esquivel

Co-Director de Tesis

Morelia, Michoacán Julio 2016

DYNAMICAM: SOFTWARE PARA ANÁLISIS DE SISTEMAS DINÁMICOS

TESIS

Que para obtener el grado de

MAESTRO EN CIENCIAS EN INGENIERÍA ELÉCTRICA

presenta

Iván Jonathan Aceves Adán

Dr. Juan José Flores Romero

Director de Tesis

Dr. Claudio Rubén Fuerte Esquivel

Co-Director de Tesis

Universidad Michoacana de San Nicolás de Hidalgo

Julio 2016

DYNAMICAM: Software para Análisis de Sistemas Dinámicos

Los Miembros del Jurado de Examen de Grado aprueban la **Tesis de Maestría en Ciencias en Ingeniería Eléctrica** de ***Iván Jonathan Aceves Adán***

Dr. Félix Calderón Solorio
Presidente del Jurado

Dr. Juan José Flores Romero
Director de Tesis

Dr. Claudio Rubén Fuerte Esquivel
Co-director de Tesis

Dr. Jaime Cerda Jacobo
Vocal

Dr. Francisco Javier Domínguez Mota
Revisor Externo
Fac. Cs. Físico Matemáticas UMSNH

Dr. Félix Calderón Solorio
Jefe de la División de Estudios de Posgrado de la Facultad de Ingeniería Eléctrica. UMSNH
(Por reconocimiento de firmas).

UNIVERSIDAD MICHOACANA DE SAN NICOLÁS DE HIDALGO
Junio 2016

Resumen

En diversas ocasiones es posible modelar mediante sistemas de ecuaciones diferenciales ordinarias el comportamiento que presentan una gran cantidad de fenómenos físicos. Por tal motivo el estudio de los sistemas dinámicos es de gran importancia en la actualidad, debido a que se encuentran relacionados al mundo real. Existen diversos métodos para coadyuvar el análisis de los sistemas dinámicos, sin embargo se cuenta con una cantidad limitada de software que proporcione de manera simple estos métodos. En esta tesis se presenta la realización de un software intuitivo para el análisis de sistemas dinámicos, el cual cuenta con una interfaz gráfica de usuario y un interfaz de programación de aplicaciones. Este software incluye los métodos de generación de diagramas de bifurcación, mapas de Poincaré, campos vectoriales, simulaciones en el dominio del tiempo y diagramas de fase. Este sistema permite producir este tipo de análisis basado en una definición única del sistema, el cual utiliza una sintaxis simple e intuitiva para el usuario. Se utiliza como base de desarrollo el software de Mathematica, gracias a que proporciona un único sistema integrado, basado en re-escritura de términos (computación simbólica). DynamicaM esta enfocado a la comunidad estudiantil y científica del área de Ingeniería Eléctrica. Además se presentan una serie de ejemplos para ilustrar y validar el uso del software.

Palabras claves: Sistemas-dinámicos, software, bifurcaciones, diagrama-de-bifurcación, ODEs.

Abstract

The behavior of physical phenomena can be modeled by systems of ordinary differential equations. Therefore the study of dynamic systems is very important today, because are related to the real world. There are several methods to the analysis of dynamic systems, however in the actuality there are a limited amount of software that provides a simple way these methods. This thesis presents the making of a intuitive software package for the analysis of dynamical systems, its which have a graphical user interface and an application programming interface. This software includes methods for the generating of bifurcation diagrams, Poincare maps, vector fields, simulations in the time domain and phase diagrams. This system can produce this type of analysis based on a single definition of the system, which uses a simple and intuitive syntax for the user. It is used as a basis for developing software Mathematica, by providing a single integrated system, based on re-writing of terms (symbolic computation). This software is focused for student and scientific community. In addition this thesis present a number of examples to illustrate and validate the use of the software. Also in this thesis present a number of examples to illustrate and validate the use of the software.

Contenido

Lista de Figuras

Lista de Tablas

Lista de Símbolos

$\dot{\mathbf{x}}$	Primer derivada respecto al tiempo t de la variable x
$\mathbf{x}$	Vector de variables de estado de un sistema dinámico
$\boldsymbol{\mu}$	Vector de parámetros de bifurcación de un sistema dinámico
$\mathbf{f}, f$	Conjunto de ecuaciones diferenciales ordinarias
A	Matriz Jacobiana
$Re()$	Parte real de un numero
$\lambda(\boldsymbol{\mu})$	Valor característico crítico
I	Matriz identidad
g	Función de mapeo
$det(A)$	Determinante de una matriz
$\mathbb{R}$	Números reales
S	Plano transversal al flujo de las trayectorias del plano de fase
p	Sección de Poincaré
t	Tiempo
P	Potencia activa
X_i	Vector posición de una partícula
$X_{fitness}$	Fitness de la posición actual
V_i	Vector velocidad de una partícula
V_{New}	Nuevo vector velocidad de una partícula
$r_1,\ r_2$	Números aleatorios generados con una distribución uniforme u(0,1)
$C_1,\ C_2$	Constantes de operación de PSO
P_{Besti}	Mejor solución encontrada por una partícula
$P_{fitness}$	Fitness de la mejor posición de las partícula
Q_1	Potencia reactiva
P_m	Potencia mecánica de entrada al generador
D_m	Par de fricción
ω	Velocidad angular del rotor
θ	Ángulo de voltaje en los nodos de la red de transmisión
θ_m	Ángulo mecánico del rotor respecto al eje de referencia del estator
θ_0	Valor inicial de θ_m
$P_0,\ K_{pw}$	Coeficientes del modelo de carga dinámica tipo II

$K_{pv},\ T,\ Q_0$	Coeficientes del modelo de carga dinámica tipo II
$K_{qw},\ K_{qv},\ K_{qv2}$	Coeficientes del modelo de carga dinámica tipo II
P_1	Potencia constante real del modelo de carga dinámica tipo II
E'	Voltaje interno de cuadratura del generador
δ	Ángulo de voltaje en el nodo interno del generador
M	Momento de inercia del generador
V	Magnitud de voltaje del nodo del sistema de transmisión

Lista de Acrónimos

API: Interfaz de programación de aplicaciones (del inglés: Application Programming Interface)

DB: Diagrama de bifurcación

DynamicaM: Software para análisis de sistemas dinámicos (del latín *dynamicam* 'dinámica', resaltando la 'M' por el entorno de Mathematica)

GIO: Optimización por interacciones gravitacionales (del inglés: Optimization by gravitational interactions)

GUI: Interfaz gráfica de usuario (del inglés: Graphical User Interface)

MATLAB: MATrix LABoratory

PSO: Optimización por enjambre de partículas (del inglés: Particle Swarm Optimization)

PSONichos: Optimización por enjambre de partículas con nichos (del inglés: Niching Particle Swarm Optimization)

XPP/XPPAUT: X-Windows Phase Plane plus Auto

Capítulo 1

Introducción

Hoy en día el estudio de los sistemas dinámicos es de gran importancia debido a que están relacionados con comportamientos que se observan en el mundo real. Estos comportamientos o fenómenos físicos pueden ser descritos por modelos matemáticos conformados por ecuaciones diferenciales ordinarias. Estos sistemas se encuentran conformados por parámetros y variables, los cuales determinan el comportamiento de los mismos [Kuznetsov98].

Dentro del campo de la investigación científica existen diversas disciplinas de la ingeniería, en particular la Ingeniería Eléctrica y Electrónica donde es necesario el estudio y análisis de sistemas dinámicos, con el fin de conocer el comportamiento que exhibe determinado sistema asociado a cierto parámetro de operación. Existen diversos tipos de software que proporcionan una gran cantidad de herramientas que permiten realizar análisis a sistemas dinámicos. Un software de los más utilizados XPP/XPPAUT [XPP03] es una herramienta numérica para simulaciones, animaciones y análisis de sistemas dinámicos, donde los modelos pueden ser discretos de estados finitos, modelos estocásticos de Markov, ecuaciones diferenciales parciales y ecuaciones integro-diferenciales. Una de las grandes desventajas de XPPAUT para generar diagramas de bifurcación es que requiere declarar un punto inicial de las variables de estado. Esto es necesario para que XPP/XPPAUT converja al punto fijo más cercano. XPP requiere de más parámetros donde el valor de estos influye en una correcta generación de los diagramas.

En esta tesis se presenta la implementación de una biblioteca de herramientas desarrolladas para el entorno de Mathematica. Esta tesis presenta a DynamicaM (del latín *dynamicam* 'dinámica', resaltando la 'M' por el entorno de Mathematica), una herramienta que ofrece una interfaz de programación de aplicaciones (API por sus siglas en inglés) y una interfaz gráfica de usuario (GUI por sus siglas en inglés) con el fin de coadyuvar el estudio y análisis de sistemas dinámicos. Algunas de las herramientas incluidas en DynamicaM son, campos vectoriales, secciones de Poincaré, simulaciones en el tiempo y diagramas de bifurcación, entre otros. DynamicaM tiene como ventajas sobre XPPAUT la visualización dinámica de diagramas de bifurcación, facilidad de uso, la sintaxis natural de los sistemas de ecuaciones diferenciales ordinarias, es decir, tal y como se definen estos sistemas matemáticamente en los libros de texto, entre otras.

1.1. Planteamiento del problema

En el ámbito de la investigación científica contamos con múltiples herramientas o software, que nos permiten resolver distintos problemas. Uno de estos problemas es el análisis de sistemas dinámicos, para el cuál contamos con diversas herramientas como XPP/XPPAUT. XPP se utiliza para la simulación numérica de las ecuaciones diferenciales, trazar diagramas de bifurcación, etc. [Doedel94]. Desafortunadamente, el uso de este software no es intuitivo, puesto que es necesario conocer a fondo su funcionamiento y en muchos casos no existe una documentación adecuada.

Debido a esto y para cubrir las necesidades de los investigadores, surge la necesidad de crear un software funcional con un diseño simple y un enfoque reduccionista (i.e. tener secciones bien definidas para cada una de las funciones, esto para facilitar al usuario el entendimiento de éstas), que tenga una estructura definida y fácil de usar. Este software debe de tener tener un número razonable de funciones que permitan realizar diferentes análisis a un mismo sistema, como son: secciones de Poicaré, diagramas de fase, campos vectoriales, diagramas de bifurcación, etc. Por lo tanto a partir de un sistema dinámico de la forma $f(x, \mu)$,

DynamicaM debe de ser capaz de generar una serie de gráficos que coadyuven en el análisis del sistema. Se considera que este aporte será de gran apoyo, debido a que en DynamicaM se reúnen diferentes métodos de análisis para los sistemas dinámicos en un mismo software.

1.2. Antecedentes

Hasta el día de hoy existe una cantidad limitada de software que se especialice en el análisis de sistemas dinámicos. Uno de los más usados para el estudio de sistemas dinámicos es el software de XPP/XPPAUT [Doedel94]. XPP/XPPAUT es una herramienta para la simulación, animación y análisis de sistemas dinámicos [Ermentrout03], que generalmente se utiliza para la simulación numérica de ecuaciones diferenciales. XPP es capaz de trazar diagramas de fase de sistemas no lineales y encontrar puntos fijos para generar diagramas de bifurcación (DB) mediante un método de continuación basado en predictor-corrector, indicando las soluciones estables e inestables de los puntos de equilibrio. Desafortunadamente esta herramienta presenta grandes retos para el usuario algunos de estos son, tener un amplio conocimiento para instalar el software correctamente en diferentes plataformas, tener conocimientos previos de Fortran/C para describir un sistema dinámico y poseer conocimientos previos de la nomenclatura usada, entre otros. Sin embargo, en los trabajos de investigación de Raúl García Kasusky [Kasusky02] y de Héctor Ramiro Carvajal Pérez [Pérez07], se recurre al uso de esta herramienta para el análisis de bifurcaciones en sistemas eléctricos y análisis de inestabilidades en sistemas eléctricos de potencia por medio de la teoría de bifurcación. Una herramienta alternativa a XPPAUT es un paquete para el análisis de sistemas dinámicos llamado pplane8.m [Polking09] desarrollado en el entorno de MATLAB [Villanueva12], el cual tiene la capacidad de encontrar puntos críticos de un sistema de ecuaciones diferenciales ordinarias y generar diagramas de fase para cada punto encontrado tal como muestra Alejandra Méndez [Méndez13].

Así mismo existen distintos trabajos de investigación en los cuales se generan DB con distintos métodos, como es el tema de investigación presentado por Julio Barrera [Barrera07] [Barrera08], en el cual se propone la generación de diagramas de bifurcación completos

mediante algoritmos de optimización inteligentes basados en optimización con enjambre de partículas (PSO), así como un algoritmo determinista llamado "el algoritmo del explorador". Similar a este trabajo se Rodrigo López [López10], presento la generación de los mismos atacando los problemas de búsqueda de raíces en sistemas de ecuaciones diferenciales trascendentales y ecuaciones que presentan discontinuidades en sistemas, de igual manera Mauricio López [Villanueva10], presenta la implementación de una herramienta para auxiliar en el trazado de DB a través del uso de técnicas metaheurísticas para la búsqueda de raíces. Aí mismo Oscar Vargas presenta una librería para la generación de DB, ésta implementada para en el lenguaje de programación de Java. Existen otros trabajos en donde se presenta la cualitativización de un DB. Jose Ortiz [Bejar08] amplia la representación cualitativa presentada por Juan Flores [Flores06], realizando predicciones o simulaciones del comportamiento que presenta el sistema. Por otra parte Héctor Rodríguez [Rangel09] se centra en generar la representación cualitativa de diagramas de bifurcación, partiendo de los datos cuantitativos.

1.3. Objetivos de la tesis

En este apartado se presentan los objetivos generales y particulares a los que se desea llegar en este trabajo de investigación.

1.3.1. Objetivo general

Desarrollar una herramienta intuitiva y funcional de nombre "DynamicaM", utilizando un sistema único integrado llamado Mathematica como base de desarrollo para la implementación de esta herramienta, con el fin de facilitar y simplificar el estudio y análisis de sistemas dinámicos representados por medio de ecuaciones diferenciales ordinarias.

1.3.2. Objetivos particulares

- Llevar acabo la codificación de los diferentes métodos o herramientas utilizadas para el análisis de sistemas dinámicos (secciones de Poincaré, campos vectoriales, diagramas de fase, simulación en el dominio del tiempo y diagramas de bifurcación).

- Diseñar y codificar la interfaz de aplicaciones e interfaz gráfica de una manera que sea comprensible y accesible para los usuarios.
- Integrar a DynamicaM herramientas para la búsqueda de raíces en el plano real y complejo presentadas presentados por Rodrigo lópez [López10] y Ana Ochoa [Ochoa16], respectivamente.
- Realizar un método para graficar los distintos diagramas de bifurcación obtenidos, permitiendo visualizar de manera dinámica un mismo resultado.
- Calificar los puntos de bifurcación según su tipo, en pitchfork, saddle-Node y Hopf.

1.4. Descripción de capítulos

Esta tesis contiene un total de cinco capítulos, cuyos contenidos son los siguientes:

- En el segundo capítulo se desarrolla una descripción de los conceptos fundamentales que son indispensables para tener una mayor comprensión de los resultados que son obtenidos mediante las herramientas implementadas. Estos conceptos son campos vectoriales, secciones de Poincaré, diagramas de fase, entre otros.
- En el tercer capítulo se da una reseña del desarrollo del software DinamicaM, se presenta una descripción del desarrollo de cada una de las funciones que se integraron al software. Así mismo se presentan ejemplos de como utilizar el software.
- En el cuarto capítulo se presentan los resultados de aplicar a tres diferentes casos de estudio las distintas herramientas con las que cuenta DynamicaM.
- En el quinto capítulo se presentan las conclusiones generales a las que se han llegado en este trabajo, los aportes realizados con este proyecto y los trabajos futuros que derivan del mismo.

1.5. Conclusiones del capítulo

En el presente capitulo se realizó una introducción al tema a tratar, presentando algunas de las herramientas existentes para el análisis a sistemas dinámicos y realizando una breve comparación entre éstas. Se planteó el objetivo general y los objetivos particulares que se desprenden del mismo. Por último se describe la estructura general de la tesis.

Capítulo 2

Sistemas dinámicos

El presente capítulo aborda una breve descripción de sistemas dinámicos, el significado de los puntos fijos y la estabilidad de los mismos según Lyapunov. Así mismo se describen métodos para el análisis de sistemas dinámicos como son campos vectoriales, secciones de Poincaré, diagramas de fase y la obtención de puntos fijos para la generación de DB.

2.1. Sistemas dinámicos

En la ciencia a menudo la descripción matemática de los procesos físicos o fenómenos nos conduce a sistemas de ecuaciones diferenciales ordinarias. Estos sistemas describen el comportamiento que presenta el proceso. La evolución de un sistema dinámico significa un cambio en el estado del sistema en el tiempo, es decir, su estado evoluciona conforme el tiempo transcurre. Esta evolución se encuentra gobernada por un conjunto de reglas que definen la dinámica del sistema [Nayfeh95]. En general un sistema dinámico puede ser descrito por el conjunto dado por (2.1).

$$\begin{aligned} \dot{x}_1 &= f_1(x_1, x_2, ..., x_n, \mu_1, \mu_2, ..., \mu_m) \\ \dot{x}_2 &= f_2(x_1, x_2, ..., x_n, \mu_1, \mu_2, ..., \mu_m) \\ &\cdot \\ \dot{x}_n &= f_n(x_1, x_2, ..., x_n, \mu_1, \mu_2, ..., \mu_m) \end{aligned} \tag{2.1}$$

donde $x_1, x_2, ...x_n$ son las variables de estado, las cuales son capaces (en conjunto) de determinar las condiciones de la dinámica del sistema para todo tiempo y $\mu_1, \mu_2, ...\mu_m$ son los parámetros de control (parámetros de bifurcación), los cuales son variados cuasiestáticamente para construir un DB. El conjunto de Ecuaciones (2.1) puede ser reescrito de forma reducida como se muestra en la Ecuación (2.2).

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \boldsymbol{\mu}),\ \mathbf{x} \in \mathbb{R}^n,\ \boldsymbol{\mu} \in \mathbb{R}^m \tag{2.2}$$

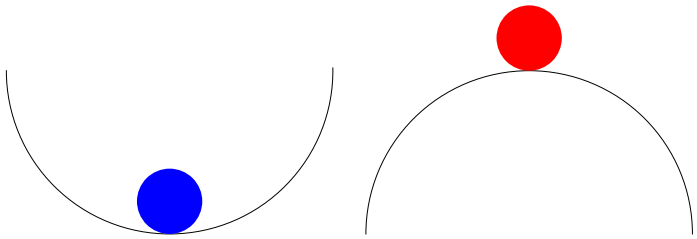
Existen diversos métodos para realizar análisis a sistemas dinámicos, algunos de estos son los métodos analíticos, graficos y numéricos, los cuales permiten tener una idea generar del comportamiento que exhibirá el sistema en un determinado tiempo.

2.2. Puntos fijos y estabilidad

Los punto fijos representan puntos de equilibrio de un sistema, estos puntos son considerados soluciones del sistema dinámico, las cuales deben de satisfacer (2.3).

$$\mathbf{f}(\mathbf{x}, \boldsymbol{\mu}) = 0 \tag{2.3}$$

Los puntos fijos o de equilibrio son estables si todas las soluciones que se encuentran en sus cercanías permanecen cerca del punto, mientras que se consideran asintóticamente estable si las soluciones además de permanecer en sus cercanías tienden hacia éste a medida que el tiempo se tiende a infinito. De otro modo, el punto de equilibrio será inestable. Es decir, un sistema es estable si al tener una entrada éste produce salidas que no divergen, por el contrario, un sistema es inestable si al tener una entrada éste produce salidas que divergen. Por ejemplo, en la Figura 2.2 se observar dos círculos, el primero se encuentra en un tazón cóncavo y el segundo en un tazón convexo, si aplicamos una pequeña perturbación al círculo azul (Figura 2.1(a)), éste regresará al mismo punto de partida (sistema estable). Por otra parte si aplicamos una fuerza al círculo rojo (Figura 2.1(b)) éste no regresará al punto de partida (sistema inestable).



(a) *Punto fijo estable* **(b) *Punto fijo inestable***

Figura 2.1: Representación de la estabilidad.

Al determinar los puntos fijos de un sistema de ecuaciones diferenciales, estos no contienen información de la estabilidad que presentan. Por lo tanto para determinar su estabilidad es imprescindible observar los valores característicos que presenta el Jacobiano en cada punto. Por lo tanto una vez que se encuentran las soluciones de $\mathbf{x}^*$ que satisfagan la Ecuación (2.3), estos son evaluados en el Jacobiano A Ecuación (2.4), se obtienen los valores característicos de la misma mediante la Ecuación (2.5). En esta ecuación A es la matriz Jacobiana evaluada en $\mathbf{x}^*$ e I es la matriz identidad de tamaño n x n. La estabilidad del punto fijo se verifica por medio del Teorema de Lyapunov (Teorema 2.1) [Kuznetsov98].

$$A = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \tag{2.4}$$

$$det(A - \lambda I) = 0 \tag{2.5}$$

Teorema 2.1 (Lyapunov)

Considere un sistema dinámico definido por:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \boldsymbol{\mu}), \;\; \mathbf{x} \in \mathbb{R}^n, \;\; \boldsymbol{\mu} \in \mathbb{R}^m \tag{2.6}$$

donde $\mathbf{f}$ *es una función continua cuya derivada existe. Supongamos que* $\mathbf{f}$ *tiene un punto fijo* $(\mathbf{x}_0, \boldsymbol{\mu}_c)$ *que satisface* $\mathbf{f}(\mathbf{x}_0, \boldsymbol{\mu}_c) = 0$*, y sea* A *la matriz del Jacobiano de* $\mathbf{f}(\mathbf{x}, \boldsymbol{\mu})$ *evaluada en el punto fijo,* $A|_{(\mathbf{x}_0, \boldsymbol{\mu}_c)}$*. Entonces el punto* $(\mathbf{x}_0, \boldsymbol{\mu}_c)$ *es estable si todos los valores propios de*

$\lambda_1, \lambda_2, ...\lambda_l$ de A tienen parte real negativa ($Re(\lambda_l) < 0$). Por el contrario un punto $(\mathbf{x}_0, \boldsymbol{\mu}_c)$ es inestable si al menos un valor propio de $\lambda_1, \lambda_2, ...\lambda_l$ de A tienen parte real positiva ($Re(\lambda_l) > 0$).

2.3. Tipos de Bifurcaciones

La estructura que puede presentar un DB varía respecto a los cambios realizados al parámetro de bifurcación μ, mientras que los cambios cualitativos que se presentan en la dinámica son denominados bifurcaciones. Los valores de los parámetros en los que se produce una bifurcación son llamados puntos de bifurcación. Las bifurcaciones son importantes científicamente debido a que proporcionan modelos de transiciones e inestabilidades mediante la variación de algunos parámetros de control [Strogatz94].

Las bifurcaciones pueden ser clasificadas en bifurcaciones locales y globales. Las bifurcaciones locales se presentan cuando ocurre un cambio cualitativo (cambios en las propiedades y características de los diagramas de bifurcación) en el entorno del punto fijo o una solución periódica del sistema. Mientras que las bifurcaciones globales involucran regiones grandes del espacio de fase, no solamente la vecindad de un punto fijo. Generalmente estas bifurcaciones involucran ciclos (ciclos nodo silla, bifurcaciones de doble periodo, bifurcaciones homoclínicas) [Strogatz94].

Las bifurcaciones locales pueden ser analizadas mediante cambios que ocurren en las propiedades de la estabilidad local. Las bifurcaciones locales mas típicas son: Nodo-Silla, tridente y Hopf.

En esta tesis únicamente se abordan los tres tipos de bifurcaciones locales, describiendo a continuación cada una de éstas.

2.3.1. Bifurcación Nodo-Silla

La bifurcación de Nodo-Silla es el tipo más simple de bifurcación; es un mecanismo por el cual los puntos fijos pueden ser creados o destruidos.

Considerando un sistema que depende de un parámetro $\boldsymbol{\mu}$:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \boldsymbol{\mu}),\ \mathbf{x} \in \mathbb{R}^n,\ \boldsymbol{\mu} \in \mathbb{R}^m \tag{2.7}$$

Se dice que una bifurcación saddle-node ocurre cuando el punto fijo $(\mathbf{x_0}, \boldsymbol{\mu_c})$ del Sistema (2.7), con un cierto valor del parámetro de bifurcación $\boldsymbol{\mu_c}$, cumple las siguientes condiciones:

- $\mathbf{f}(\mathbf{x_0}, \boldsymbol{\mu_c}) = 0$
- $D_{\mathbf{x}}\mathbf{f}$ cuenta con un valor característico con parte real cero mientras que todos los demás valores tienen parte real diferente de cero en $(\mathbf{x_0}, \boldsymbol{\mu_c})$.
- $\frac{\partial \mathbf{f}}{\partial \mu}|_{\mathbf{x_0}, \boldsymbol{\mu_c}} \neq 0$

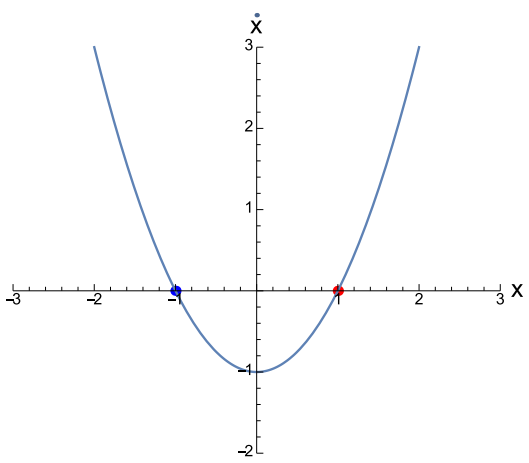
La primera condición asegura que la solución considerada sea un punto de equilibrio del Sistema (2.7). La segunda condición implica que la solución es un punto de equilibrio no hiperbólico y la tercera condición implica condiciones de transversalidad, es decir, existe una función continua diferenciable $\lambda = h(x)$ en el punto de bifurcación (x_0, μ_c). En términos geométricos significa que el punto de equilibrio intersecta la línea $x = x^*$ transversalmente [Nayfeh95].

El ejemplo más común de una bifurcación saddle-node está dado por el sistema de primer orden, dado por la Ecuación (2.8) [Strogatz94].

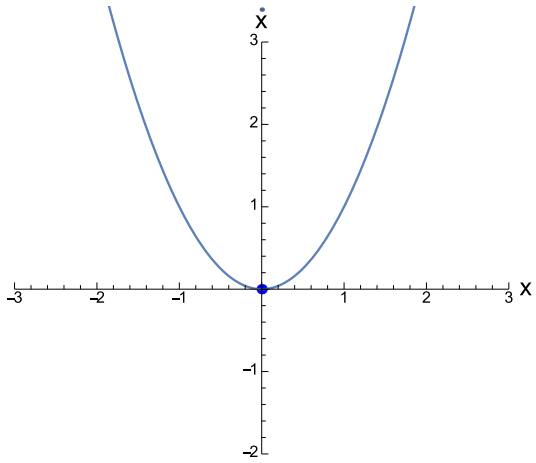
$$\dot{x} = r + x^2 \tag{2.8}$$

En donde r es el parámetro de bifurcación, el cual puede tomar valores positivos, negativos y cero, cuando $r < 0$ (Figura 2.2(a)), existen dos puntos fijos, uno estable y el otro inestable.

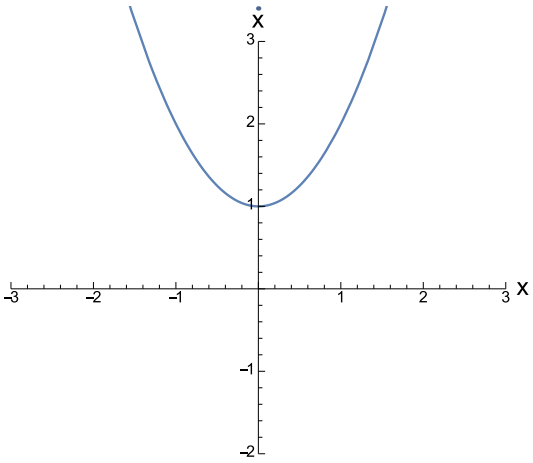
A medida que el parámetro r va aumentando los puntos fijos se acercan entre sí. Cuando $r = 0$ (Figura 2.2(b)) los dos puntos fijos se convierten en un punto fijo único que sólo es estable por la izquierda, mientras que para valores positivos de r (Figura 2.2(c)) no existen puntos fijos. El diagrama de bifurcación de este sistema se observa en la Figura 2.3.



(a) $r < 0$



(b) $r = 0$



(c) $r > 0$

Figura 2.2: Posibles puntos fijos que pueden presentarse en el Sistema (2.8)

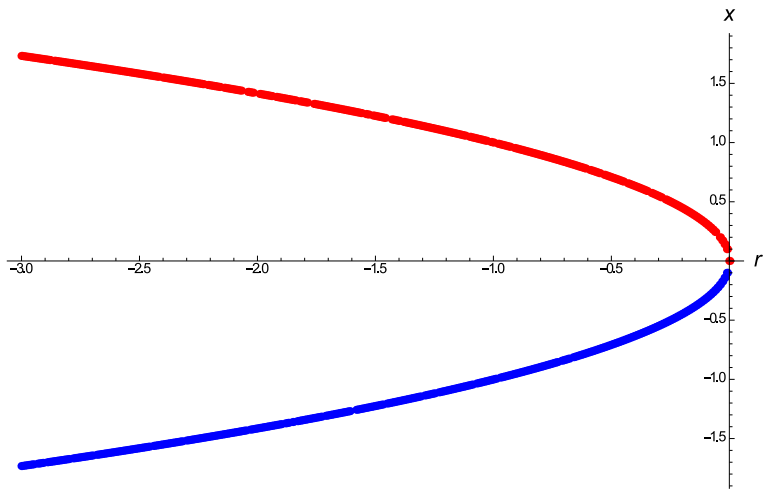


Figura 2.3: Diagrama de bifurcación de la Ecuación (2.8).

En este ejemplo, la bifurcación se produce cuando $r = 0$, debido a que los campos vectoriales para $r < 0$ y $r > 0$ son cualitativamente diferentes [Strogatz94].

2.3.2. Bifurcación Tridente

Este tipo de bifurcaciones son comunes en problemas físicos que cuentan con simetría física o espacial de izquierda a la derecha. Existen dos tipos diferentes de bifurcación tridente la primera llamada supercrítica y la segunda subcrítica.

Se dice que una bifurcación tridente ocurre cuando el punto fijo $(\mathbf{x_0}, \boldsymbol{\mu_c})$ del Sistema (2.7), con un cierto valor del parámetro de bifurcación $\boldsymbol{\mu_c}$, cumple las siguientes condiciones: [Rasband90]

- $\mathbf{f}(-\mathbf{x_0}, \boldsymbol{\mu_c}) = -\mathbf{f}(\mathbf{x_0}, \boldsymbol{\mu_c})$
- $\left.\frac{\partial \mathbf{f}}{\partial x}\right|_{(\mathbf{x_0}, \boldsymbol{\mu_c})} = 0$
- $\left.\frac{\partial^2 \mathbf{f}}{\partial x \partial \mu}\right|_{(\mathbf{x_0}, \boldsymbol{\mu_c})} > 0$
- $\left.\frac{\partial^3 \mathbf{f}}{\partial x^3}\right|_{(\mathbf{x_0}, \boldsymbol{\mu_c})} \neq 0$

Una bifurcación tridente supercrítica ocurre si:

$$\left.\frac{\partial^3 \mathbf{f}}{\partial x^3}\right|_{(\mathbf{x_0},\boldsymbol{\mu_c})} > 0$$

Una bifurcación tridente subcrítica ocurre cuando:

$$\left.\frac{\partial^3 \mathbf{f}}{\partial x^3}\right|_{(\mathbf{x_0},\boldsymbol{\mu_c})} < 0.$$

Un ejemplo de un sistema que presenta una bifurcación tridente supercrítica es representado por el Sistema (2.9):

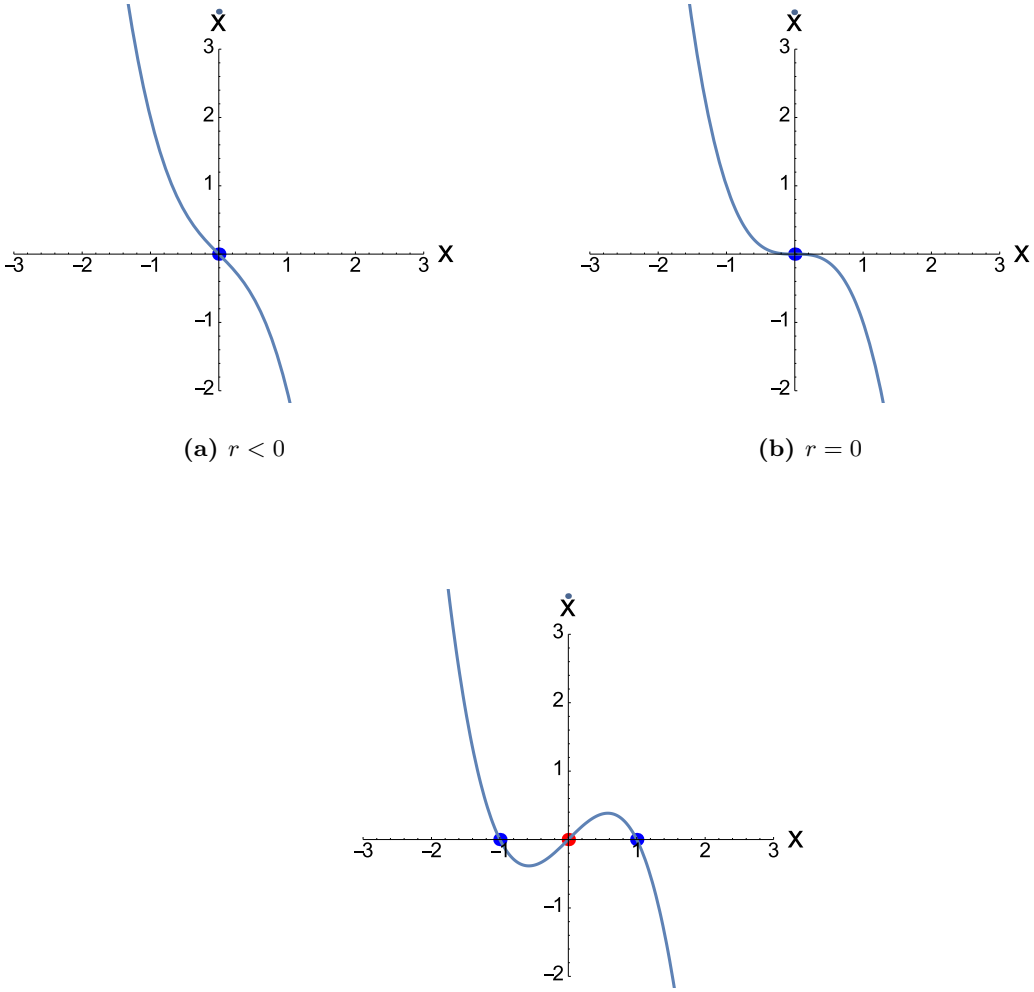
$$\dot{x} = rx - x^3 \tag{2.9}$$

Este sistema tiene sus puntos fijos en $x^* = 0$ y en $x^* = \pm\sqrt{r}$. La Figura 2.4 muestra que para el valor de $r < 0$ (Figura 2.4(a)) existe solo un punto de equilibrio estable. Cuando el parámetro $r = 0$ (Figura 2.4(b)) este punto es aún estable pero más débil, puesto que la linealización desaparece. Cuando de $r > 0$ (Figura 2.4(c)), este punto de equilibrio se vuelve inestable y da origen a dos puntos fijos estables en forma simétrica [Strogatz94].
El diagrama de bifurcación asociado al Sistema (2.9) se muestra en la Figura 2.5, comprobando mediante éste lo expuesto anteriormente.

El Sistema (2.10) presenta una bifurcación tridente subcrítica.

$$\dot{x} = rx + x^3 \tag{2.10}$$

En la Figura 2.6 se observa el diagrama de bifurcación asociado al Sistema (2.10), en la cual la bifurcación tridente se encuentran invertida en comparación con el diagrama de la Figura 2.5. Los puntos fijos distintos de cero $x^* = \pm\sqrt{-r}$ son inestables y existen únicamente del lado derecho de la bifurcación ($r < 0$), lo que motiva el término "subcrítica". Más importante, el origen es estable para $r < 0$ e inestable para $r > 0$, como en el caso supercrítico, pero ahora la inestabilidad para $r > 0$ no depende el término cúbico.



(a) $r < 0$ **(b)** $r = 0$

(c) $r > 0$

Figura 2.4: Posibles puntos fijos que pueden presentarse en el Sistema (2.9)

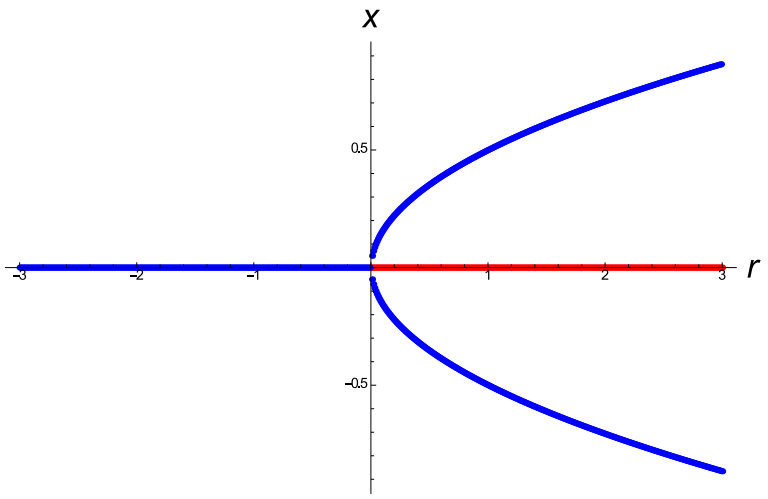


Figura 2.5: Bifurcación tridente supercrítica asociado al Sistema(2.9)

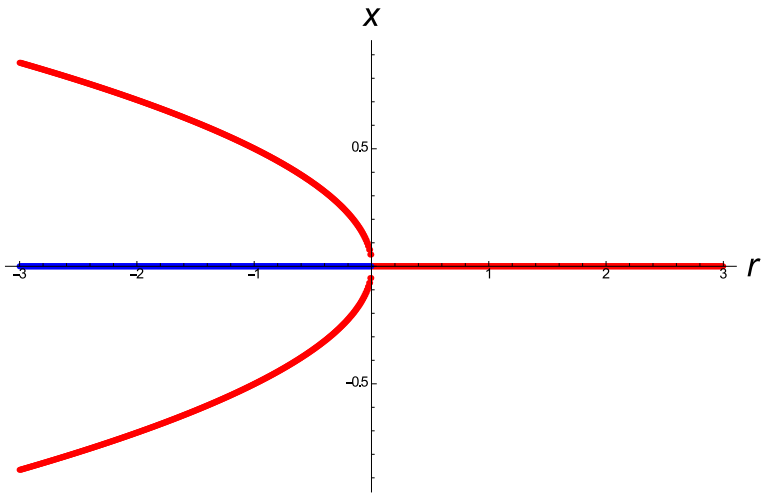


Figura 2.6: Bifurcación tridente subcrítica asociado al Sistema (2.10)

2.3.3. Bifurcación Hopf

La bifurcación Hopf, también llamada bifurcación Poincaré Andronov Hopf, se produce normalmente cuando un par conjugado de eigenvalores de puntos fijo son puramente imaginarios. Ésta sólo puede ocurrir en los sistemas que cuentan con mas de una dimensión. Para que un sistema sea estable los eigenvalores de este deben de ser complejos conjugados.

En cambio, para que un sistema sea inestable, un eigenvalor real o dos eigenvalor complejos conjugados deben de cruzar el eje imaginario por cero y volverse positivos cuando el parámetro de bifurcación es variado. Cuando sucede esto se da origen a una bifurcación Hopf.

De manera mas formal una bifurcación Hopf de un punto fijo $(\mathbf{x_0}, \boldsymbol{\mu_c})$ se produce cuando las siguientes condiciones son satisfechas [Mithulananthan02]:

- $\mathbf{f}(\mathbf{x_0}, \boldsymbol{\mu_c}) = 0$
- El Jacobiano evaluado en el punto $\frac{\partial \mathbf{f}}{\partial x}|_{\mathbf{x_0}, \boldsymbol{\mu_0}}$ deberá tener solamente un par de valores característicos puramente imaginarios $\lambda = \pm j\beta$, mientras que todos los demás valores característicos tienen parte real diferente de cero.
- La relación de cambio de la parte real del valor característico crítico $\Re(\lambda(\boldsymbol{\mu}))$ con respecto al parámetro de bifurcación $\boldsymbol{\mu}$ deberá ser diferente de cero, es decir:
$$\frac{d\Re(\lambda(\boldsymbol{\mu}))}{d\boldsymbol{\mu}}|_{\boldsymbol{\mu}\neq 0}$$

La primera condición asegura que el punto fijo sea una solución del sistema, la segunda condición asegura que el punto de fijo sea no hiperbólico y la tercera implica el cruce de los eigenvalores complejos conjugados hacia el plano derecho del plano complejo, mejor conocida como condición de transversalidad.

2.4. Campos vectoriales

Un campo vectorial es una expresión de calculo vectorial que asigna un vector a cada punto en el espacio, de la forma $F : \mathbb{R}^n \to \mathbb{R}^n$ [Galbis12]. Son utilizados a menudo para modelar el comportamiento que presenta algún sistema. Por ejemplo en el campo de la física, son ocupados para representar la velocidad y dirección de algún fluido, o bien la intensidad y dirección que presentan las fuerzas gravitatoria o electromagnética.

2.4.1. Campos vectoriales en 2D

Una función vectorial $\mathbf{F} : \mathbb{R}^2 \to \mathbb{R}^2$ puede ser visualizada como un campo vectorial. Es decir, sea D un subconjunto de $\mathbb{R}^2$, un campo vectorial sobre $\mathbb{R}^2$ es una función $\mathbf{F}$ que asigna, a cada punto (x, y) de D un vector de dos dimensiones $\mathbf{F}(x, y)$. Tomando un punto (x, y), se traza el valor de $\mathbf{F}(x, y)$ como un vector, el cual se encuentra anclado en (x, y) como se aprecia en la Figura 2.7.

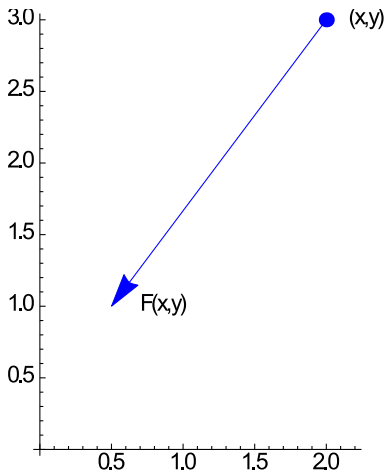


Figura 2.7: Como se representa un campo vectorial.

Repitiendo este paso para un conjunto de puntos (x, y), se logra visualizar un campo vectorial correspondiente a la función $\mathbf{F}(x, y)$. Por ejemplo, para obtener el campo vectorial del Sistema (2.11), se sigue lo descrito anteriormente, evaluando la función en los intervalos $x \in (-3, 3)$ y $y \in (-3, 3)$. Como resultado se produce el campo vectorial mostrado en la Figura 2.8. Como se puede observar el campo vectorial del Sistema (2.11) parece girar en sentido horario, lo que nos lleva a que si colocamos un punto en la circunferencia de color negro, este punto se mantendrá girando sobre ésta. Por lo tanto se puede decir que ésta es una zona estable del sistema.

$$\mathbf{F}(x, y) = \begin{bmatrix} f_1(x, y) \\ f_2(x, y) \end{bmatrix} = \begin{bmatrix} -y \\ x \end{bmatrix} \tag{2.11}$$

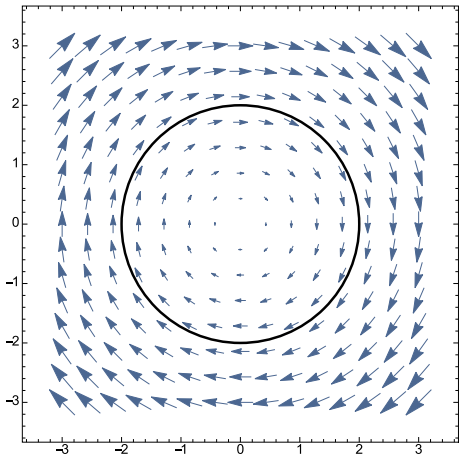


Figura 2.8: Campo vectorial del Sistema (2.11).

2.4.2. Campos vectoriales en 3D

Se puede visualizar un campo vectorial en tres dimensiones, para funciones vectoriales del tipo $\mathbf{F} : \mathbb{R}^3 \to \mathbb{R}^3$. Es decir, sea E un subconjunto de $\mathbb{R}^3$, un campo vectorial sobre $\mathbb{R}^3$ es una función $\mathbf{F}$ que asigna, a cada punto (x, y, z) de E un vector en tres dimensiones $\mathbf{F}(x, y, z)$. Tomando un punto (x, y, z), se traza el valor de $\mathbf{F}(x, y, x)$ como un vector, el cual se encuentra anclado en el punto (x, y, z).

Por ejemplo el campo vectorial del Sistema (2.12), se obtiene aplicando lo descrito anteriormente, esto es, evaluando dicha función en los intervalos de $x \in (-0.5, 0.5), y \in (-0.5, 0.5)$ y $z \in (-0.5, 0.5)$ tomando las anclas correspondientes a cada combinación de valores, se obtiene el campo vectorial mostrado en la Figura 2.9. El campo vectorial del Sistema (2.12) corresponde a una rotación en tres dimensiones, sobre el eje z, presentando el mismo comportamiento que el campo vectorial mostrado en la Figura 2.8, el cual corresponde al Sistema (2.11).

$$\mathbf{F}(x, y) = \begin{bmatrix} f_1(x, y) \\ f_2(x, y) \\ f_3(x, y) \end{bmatrix} = \begin{bmatrix} \frac{y}{z} \\ \frac{-x}{z} \\ 0 \end{bmatrix} \tag{2.12}$$

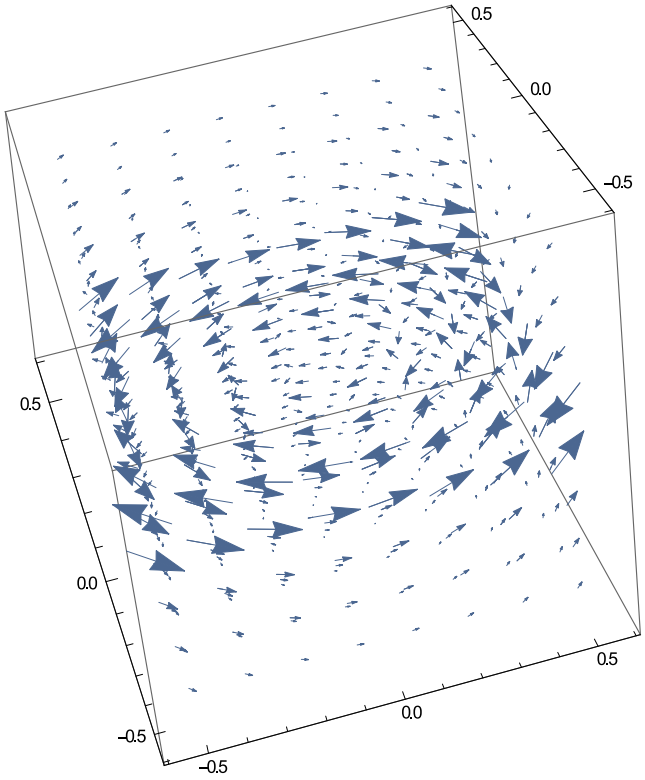


Figura 2.9: Campo vectorial de la Función (2.12).

2.5. Diagramas de fase

El diagrama de fase es una técnica muy usada para entender el comportamiento asintótico o dinámico de sistemas de ecuaciones diferenciales. Un diagrama de fase nos ofrece información cualitativa acerca de la estabilidad de los sistemas; es útil para conocer si éste converge o no a un estado de equilibrio intertemporal estático. En palabras mas formales, un diagrama de fase es una colección de trayectorias que representan las soluciones de ecuaciones en el espacio de fase, proporcionando información sobre los comportamientos transitorios y asintóticos de las soluciones del sistema [Lerm03].

La forma general de un campo vectorial en el espacio de fase está dada por la Ecuación (2.13)

$$\begin{aligned} \dot{x_1} &= f_1(x_1, x_2) \\ \dot{x_2} &= f_2(x_1, x_2) \end{aligned} \tag{2.13}$$

donde f_1 y f_2 son funciones. Este sistema puede ser reescrito de manera mas compacta mediante notación vectorial tal como se observa en la Ecuación (2.14)

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) \tag{2.14}$$

donde $\mathbf{x} = (x_1, x_2)$ y $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}))$. $\mathbf{x}$ representa un punto en el plano de fase, y $\dot{\mathbf{x}}$ es el vector de velocidad de éste punto. Para seguir el flujo del campo vectorial, un punto de fase traza una solución de $\mathbf{x}(t)$, correspondiente a una trayectoria del plano de fase [Strogatz94].

Por ejemplo el plano de fase de un sistema Masa-Resorte es visualizado en dos dimensiones. El estado del sistema es determinado por la posición x y la velocidad v. El diagrama de fase captura todos los posibles estados del sistema (x, v). Por lo tanto para un sistema Masa-Resorte:

$$\begin{aligned} \dot{x}[t] &= v[t] \\ \dot{v}[t] &= -(bv[t] + kx[t])/m \end{aligned} \tag{2.15}$$

Donde m es la masa del cuerpo, b es el coeficiente de fricción y k la constante del resorte. Asumiendo que los parámetros toman los valores numéricos de $b = 2$, $k = 3$ y $m = 4$, entonces tenemos que el sistema resultante se muestra en la Ecuación (2.16).

$$\begin{aligned} \dot{x}[t] &= v[t] \\ \dot{v}[t] &= -(2v[t] + 3x[t])/4 \end{aligned} \tag{2.16}$$

Una vez calculadas las trayectorias del sistema, para el conjunto de estados iniciales $(x_0, v_0) \in ((1, 2), (-3, 1), (-1, -2), (2, 2))$, se obtiene como resultado el diagrama de fase que se observa en la Figura 2.10. Donde cada linea es una solución del sistema, diferenciando a cada solución mediante un color diferente.

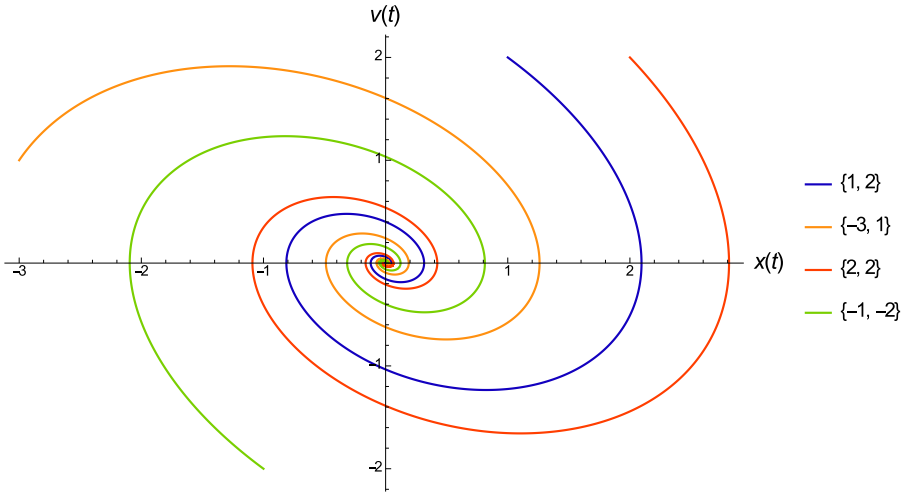


Figura 2.10: Diagrama de fase del sistema Masa-Resorte

Diagramas de fase en 3D

Un ejemplo de un diagrama de fase en 3D esta dado por la ecuación de Lorenz [Seydel10], el cual es un sistema dinámico no lineal derivado de las ecuaciones de NavierStokes de mecánica de fluidos. Mediante este sistema Lorenz trató de encontrar un modelo matemático que permitiera predecir el comportamiento de grandes masas de aire. Lorentz consiguió ajustar el modelo a sólo tres variables (P, R, b), las cuales indican como será el cambio de la velocidad y la temperatura del aire a lo largo del tiempo. Este sistema está descrito por el conjunto de ecuaciones (2.17).

$$\begin{aligned} \dot{y_1} &= P(y_2 - y_1) \\ \dot{y_2} &= -y_1 y_3 + R y_1 - y_2 \\ \dot{y_3} &= y_1 y_2 - b y_3 \end{aligned} \tag{2.17}$$

Asumiendo que los parámetros toman los valores numéricos de $P = 16$, $R = 40$ y $b = 4$, entonces el sistema resultante se muestra en la Ecuación (2.18).

$$\begin{aligned} \dot{y_1} &= 16(y_2 - y_1) \\ \dot{y_2} &= -y_1 y_3 + 40 y_1 - y_2 \\ \dot{y_3} &= y_1 y_2 - 4 y_3 \end{aligned} \tag{2.18}$$

Debido a que el sistema es tridimensional el diagrama de fase estará en una región $\mathbb{R}^3$. Por lo tanto se toma en cuenta el punto inicial situado en $y_1 = 1.051$, $y_2 = -0.30654$ y $y_3 = 10$. Se calcula la trayectoria que genera éste y se tiene como resultado el diagrama de fase que se muestra en la Figura 2.11.

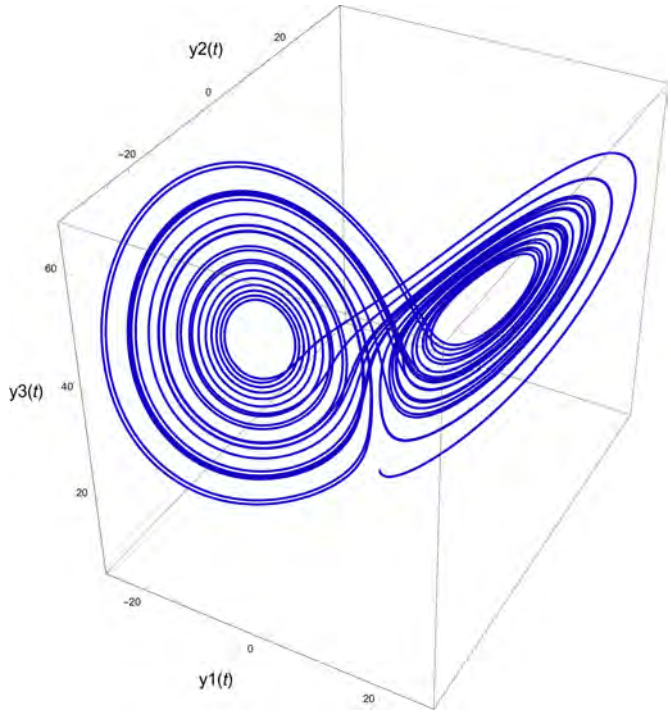


Figura 2.11: Diagrama de fase del Sistema (2.18)

Como se observa en las Figuras 2.10 y 2.11 los diagramas de fase son una representación de un conjunto de soluciones en los cuales se crean curvas paramétricas dentro del plano cartesiano. Cada linea de flujo representa una solución particular de los sistemas.

2.6. Secciones de Poincaré

Dentro de los sistemas dinámicos, la sección de Poincaré o mapas de Poincaré, llamada así por Henri Poincaré, es la intersección de una órbita periódica en el espacio del sistema dinámico con un cierto subespacio de menor dimensión.

Los secciones de Poincaré usualmente son usadas para estudiar flujos en forma de remolino, tal es el caso del flujo cercano a una órbita periódica. Considerando un sistema n-dimensional $\dot{x} = f(x)$. S es una sección de la superficie de dimensión $n-1$, en donde S debe de ser transversal al flujo de todas las trayectorias, las cuales tienen que pasar a través de S y no paralelamente.

Una mapa de Poincaré p es obtenido de S, debido a que en S se encuentran las intersecciones de las trayectorias. Si $x_k \epsilon S$ entonces la k-ésima intersección, en el mapa de Poincaré es definida como:

$$x_{k+1} = P(x_k). \tag{2.19}$$

Suponiendo que x^* es un punto fijo de p, $p(x^*) = x^*$. Entonces una trayectoria que comienza en x^* y después de un tiempo T retorna a x^*, se dice que es una órbita cerrada del sistema original $\dot{x} = f(x)$. Por otra parte, al observar el comportamiento que tiene p al estar cerca del punto fijo, podemos determinar la estabilidad de la órbita cerrada [Strogatz94].

Esta técnica ofrece varias ventajas en el estudio de las ecuaciones diferenciales ordinarias, algunas de ellas son las siguientes:

1. Reducción dimensional: La construcción de las secciones de Poincaré implica la eliminación de al menos una de las variables del problema resultantes en el estudio de un problema de menor dimensión.

2. Dinámica Global: En problemas de menor dimensión (dimensión <= 4) las secciones de Poincaré calculadas numéricamente ofrecen un mapa intuitivo y sorprendente del sistema dinámico global.

Desafortunadamente, no existen métodos generales aplicables a las ecuaciones diferenciales ordinarias, debido a que la construcción de las secciones de Poincaré requiere un poco de conocimiento de la estructura geométrica del espacio de la ecuación diferencial ordinaria [Wiggins03].

Por ejemplo para el sistema Masa-Resorte gobernado por el sistema de Ecuaciones (2.16), es posible obtener su mapa de Poincaré, debido a que conocemos la estructura geométrica que presenta en el espacio de fase Figura 2.10. Por lo tanto si se define una recta que se encuentre situada en $v[t] = 0.5$, se podrán proyectar las secciones de Poincaré sobre esta, generando entonces el mapa de Poincaré que se observa en la Figura 2.12.

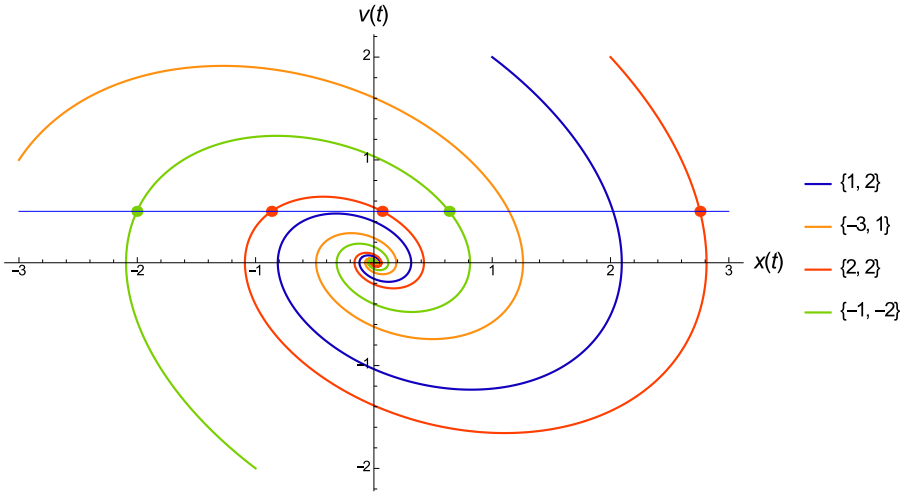


Figura 2.12: Mapa de Poincaré del Sistema (2.16)

2.7. Trazado de diagramas de bifurcación método tradicional

En particular en sistemas dinámicos, un diagrama de bifurcación muestra los valores asintóticos (puntos fijos, órbitas periódicas, o atractores caóticos) de un sistema en función de un parámetro de operación μ. En diversas ocasiones es habitual representar las soluciones estables e inestables como líneas solidas y punteadas, respectivamente, en esta tesis los puntos fijos estables e inestables se representan mediante el color azul y rojo, respectivamente. Existen diversos métodos para la generación de DB, comúnmente estos se encuentran basados en métodos de continuación.

Los métodos de continuación son utilizados para la generación de diagramas de bifurcación. Estos métodos consisten en tomar un punto inicial (x^1, μ_1) para calcular el resto de los puntos fijos deseados $(x^2, \mu_2), (x^3, \mu_3), ..., (x^j, \mu_j)$, la mayoría de estos métodos se encuentran basados en métodos de predictor-corrector, por lo tanto para calcular el (x^{j+1}, μ_{j+1}) se inicia en la solución (x^j, μ_j), ilustrando el proceso en la Ecuación (2.20).

$$(x^j, \mu_j) \xrightarrow{predictor} (\overline{x}^{j+1}, \overline{\mu}_{j+1}) \xrightarrow{corrector} (x^{j+1}, \mu_{j+1}) \tag{2.20}$$

Por lo general, el predictor $(\overline{x}^{j+1}, \overline{\mu}_{j+1})$ no es una solución, este provee una aproximación inicial para que al iterar el corrector encuentre la solución que satisfaga la Ecuación (2.3). En la Figura 2.13 las iteraciones del corrector están denotadas por puntos, la distancia entre dos soluciones calculadas consecutivamente (x^j, μ_j) y (x^{j+1}, μ_{j+1}) es llamada longitud de paso o tamaño de paso.

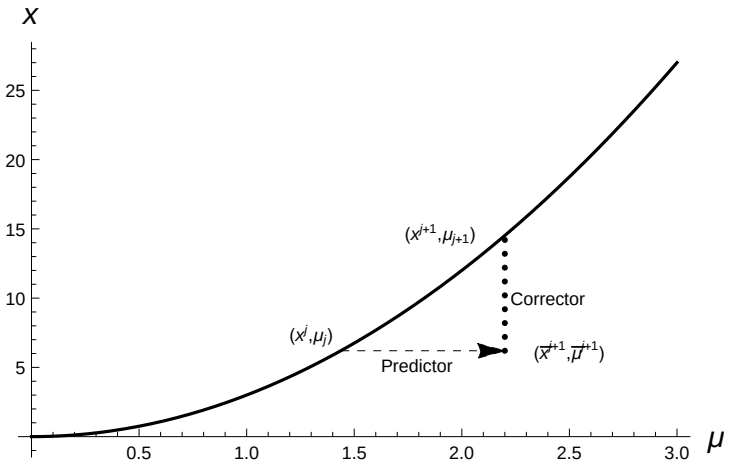


Figura 2.13: Método de continuación Predictor-Corrector.

La mayoría de los métodos de continuación usados en el análisis de bifurcación implementan el método descrito anteriormente, el cual incluye tres pasos fundamentales que se realizan de forma repetitiva:

1. Predicción del próximo punto.

2. Corrección.

3. Control de la longitud del paso.

Estos métodos se pueden dividir en dos grupos: métodos para ecuaciones diferenciales ordinarias y métodos de extrapolación polinomial. Los primeros se encuentran basados en la función $f(x, \mu)$ y sus derivadas, mientras que los segundos solo se encuentran basados en las soluciones (x, μ).

El control del paso es un factor importante en el desempeño del algoritmo de continuación, debido a que afecta directamente la convergencia del mismo. Por ejemplo, si se selecciona un paso demasiado pequeño, el proceso de continuación puede demorar demasiado tiempo, o bien si el paso es muy grande puede generar que el corrector no converja.

2.8. Obtención de diagramas de bifurcación mediante PSO

Dentro de Inteligencia Artificial existen diversos métodos para el trazado de diagramas de bifurcación. Esta tesis se centra en generación de DB mediante optimización por enjambre de partículas con nichos (PSONichos). PSO es una metaheurística poblacional propuesta por Kennedy y Eberhard en 1995 [Kennedy95], inspirada en el comportamiento social del vuelo de las bandadas de aves. Existen diversas variante del algoritmo de PSO, siendo PSONichos unos de ellas. PSONichos es un algoritmo de optimización multimodal, eficiente para localizar con éxito múltiples soluciones en problemas de optimización [Brits02].

2.8.1. Función de aptitud

Dentro de los algoritmos de evolución inteligente es imprescindible contar con una función que guíe la evolución de las particular a través de las generaciones. Dicha función es llamada función de aptitud. Esta función tiene como objetivo principal la de medir la calidad de las soluciones generadas por el algoritmo evolutivo.

Para el caso de los sistemas dinámicos el diseño de la función objetivo (función que se desea optimizar) será el conjunto de ecuaciones diferenciales ordinarias asociadas al sistema f. Al ser evaluada esta función tendrá como salida un vector de dimensión n, debido a que se cuentan con n ecuaciones diferenciales. Como la solución obtenida por la función de objetivo es un vector del tamaño del número de ecuaciones diferenciales ordinarias, es indispensable mapear la solución a un escalar. Para que ésta funja como función de aptitud y mediante ésta guiar la evolución del algoritmo de PSO. Esto se logra mediante alguna función de mapeo que satisfaga la Ecuación (2.21).

$$g : \mathbb{R}^n \to \mathbb{R} \tag{2.21}$$

Función de mapeo

La función de mapeo utilizada para este caso es la propuesta por Varum Aggarwal [Aggarwal00], la cual a sido utilizada en diversos trabajos de investigación para la búsqueda de raíces en sistemas dinámicos (v.g. Julio Barrera [Barrera08], Rodrigo López [López10], Ana Ochoa [Ochoa16]). Esta función transforma un problema de búsqueda de raíces, a un problema de búsqueda de máximos. La cual está descrita por la Ecuación (2.22).

$$g(\mathbf{x}) = \frac{1}{1 + ||\mathbf{f}(\mathbf{x}, \boldsymbol{\mu})||} \tag{2.22}$$

donde $|\mathbf{f}(\mathbf{x}, \boldsymbol{\mu})|$ corresponde a la norma euclídea de un vector, por lo tanto se puede observar que cuando $|\mathbf{f}(\mathbf{x}, \boldsymbol{\mu})| \neq 0$ entonces $g(\mathbf{x}) < 1$, cuando $|\mathbf{f}(\mathbf{x}, \boldsymbol{\mu})| = 0$ entonces $g(\mathbf{x}) = 1$. Esta función se mantiene en un rango de $(0, 1)$. Por lo tanto se convierte de esta manera un problema de búsqueda de ceros para la función $\mathbf{f}$ a un problema de optimización de g.

2.8.2. Funcionamiento básico del algoritmo de enjambre de partículas (PSO)

El algoritmo de PSO comienza iniciando una nube de partículas o enjambre. Las partículas son las encargadas de explorar el espacio de búsqueda. Cada partícula es una so-

lución que tiene la posibilidad de llegar a ser óptima a través del paso de las iteraciones. A cada partícula se le asocian diversos componentes como son el aptitud, posición, velocidad, memoria, etc. Estos atributos son los encargados de guiar la evolución del enjambre.

Cada partícula se encuentra conformada por un vector $X \in \Re^d$, el cual almacena la posición actual de la partícula en el espacio de búsqueda, un $X_{fitness}$ que almacena la aptitud de la solución actual (X), un vector P_{Besti}, el cual almacena la mejor solución encontrada hasta el momento por la partícula y el $P_{fitness}$ almacena la aptitud de la mejor posición obtenida hasta el momento (P_{Besti}). El vector $V \in \Re^n$ almacena la velocidad de la partícula.

Para que una partícula sea desplazada en el espacio de búsqueda primero se calcula el nuevo vector de velocidad V_i mediante la Ecuación (2.23).

$$V_{New} \longleftarrow V_i + C_1 r_1 (P_{Besti} - X_i) + C_2 r_2 (G - X_i) \tag{2.23}$$

donde $C_1 r_1 (P_{Besti} - X_i)$ y $C_2 r_2 (G - X_i)$ son los componentes cognitivo y social, respectivamente. P_{Besti} es la mejor solución encontrada por la partícula, C_1, C_2 representan las rapideces de aprendizaje que controlan los componentes, G es la partícula con el mejor $P_{fitness}$, r_1 y r_2 son números aleatorios que siguen una distribución uniforme $U(0, 1)$.

Una vez calculada la velocidad, simplemente se añade esta al vector posición X_i para obtener un nuevo vector de posición (Ecuación (2.24)).

$$X_{New} \longleftarrow X_i + V_i \tag{2.24}$$

Una vez calculada el nuevo vector de posición este es evaluado. Si la aptitud de éste es mejor que el $P_{fitness}$ entonces se realizan los cambios mostrados en la Ecuación (2.25).

$$P_{Besti} \longleftarrow X_i \;\; ; \;\; P_{fitness} \longleftarrow X_{fitness} \tag{2.25}$$

2.8.3. Optimización mediante enjambre de partículas con nichos

Existen funciones que cuentan con múltiples soluciones o raíces, conocidas como funciones multimodales. Para este tipo de funciones es común tratar de encontrar todas las soluciones locales mediante métodos clásicos. Estos métodos se ejecutan exhaustivamente con distintos puntos iniciales, esperando de esta manera localizar todas las soluciones que presenta el problema. Existen alternativas para encontrar las soluciones para este tipo de problemas. Una de estas alternativas es la optimización multimodal la cual trata problemas de funciones que contengan múltiples soluciones; dentro de la optimización multimodal existen a su vez diversos métodos, algunos de estos son PSONichos (R. Brits [Brits02]) y la optimización por interacciones gravitacionales (Rodrigo López [López10]) (GIO, por sus siglas en inglés).

La optimización mediante enjambre de partículas con nichos es una modificación al algoritmo de PSO, el cual tiene como ventaja la optimización multimodal o multiobjetivo. La idea general de esta modificación se basa en la manera en que algunas especies crean nichos, para explorar diferentes sub-regiones de su hábitat.

Como se describe en el trabajo de investigación de Rodrigo López [López10], al algoritmo de PSO se le realizan algunas modificaciones. Una vez creado el enjambre, en cada iteración se ordenan las partículas de mayor a menor aptitud (en el caso de maximizar); las partículas ordenadas se colocan en una pila donde el último elemento es la partícula con la mayor aptitud, después se toma la partícula que encuentre en la cima de la pila y si existen nichos se revisa si pertenece a algún nicho. De ser así se agrega al nicho y se incrementa el contador de partículas de cada nicho. Si no existe o no pertenece a algún nicho, esta partícula forma un nuevo nicho y se incrementa el contador de nichos. Se sacan todas las partículas de la pila repitiendo este procedimiento, se hace esto hasta que la pila quede vacía.

Como se presenta en el tema de investigación de Rodrigo López [López10], el algoritmo de PSONichos es una opción viable para el trazado de DB, debido a que mediante éste se

han podido trazar diagramas de bifurcación completos variando uno o mas parámetros de bifurcación de manera no supervisada, en espacios muy grandes de búsqueda.

2.9. Conclusiones del capítulo

En el presente capítulo se presentaron los conceptos fundamentales para la correcta interpretación de los resultados obtenidos en este trabajo. Así mismo se mostraron ejemplos de cada uno de éstos, esto para tener una idea clara y concisa. El siguiente capítulo describe la implementación en DynamicaM de los conceptos que se presentaron anteriormente.

Capítulo 3

Diseño de DynamicaM

3.1. Introducción

En este capítulo se presentan las funciones implementadas que conforman el software de DynamicaM. Se describen las dos secciones principales que lo conforman, la interfaz gráfica de usuario y la interfaz de programación de aplicaciones. Se explica cada uno de los apartados con los que cuenta cada interfaz y se muestran ejemplos del funcionamiento de estas. Todos los resultados mostrados en este trabajo de investigación han sido obtenidos mediante DynamicaM.

3.2. Definición de sistemas dinámicos

La naturaleza simbólica con la que cuenta Mathematica en el lenguaje de Wolfram permite que su código efectue cálculos simbólicos a ecuaciones diferenciales ordinarias. Esta propiedad constituye un factor importante para definir de manera característica los sistemas de ecuaciones diferenciales.

Con el propósito de definir el conjunto de ecuaciones diferenciales, constantes y ecuaciones algebraicas que definen al sistema dinámico, se crearon tres funciones que se encuentran incluidas en el API: DefineConstants, DefineAlgebraicEquation y DefineODE. Las cuales en conjunto son utilizadas para definir el sistema dinámico mediante la función DefineSystem.

El código correspondiente a estas funciones se encuentra disponible en el Apéndice A.

Cada una de estas funciones implementan reglas de sustitución. Por ejemplo, la función $DefineSystem$, la cual tiene como parámetros el sistema de ecuaciones diferenciales ordinarias, las constantes, las ecuaciones algebraicas, variables de estado y los parámetros de bifurcación. Una vez que se ejecuta esta función se extraen las ecuaciones algebraicas y constantes. Posteriormente mediante reglas de sustitución, los valores de las ecuaciones y contantes son sustituidos en el sistema dinámico. A continuación se presenta un ejemplo del uso de cada una de estas funciones:

La función $DefineConstants$ es empleada cuando el sistema cuenta con constantes. Un ejemplo de su uso es el siguiente:

```
1 C = DefineConstants[a -> 2, b -> 4 ,c -> 1, d -> 0]
```

La función $DefineAlgebraicEquation$, permite especificar las ecuaciones algebraicas que llegan a formar parte de un sistema de ecuaciones diferenciales ordinarias. Un ejemplo de esto es el siguiente:

```
1 Ae = DefineAlgebraicEquation[P ->  m x + n, Q -> r + 5 g]
```

Como se observa, los valores correspondientes a las contantes y las ecuaciones algebraicas son asignados a estas mediante una flecha $(->)$. Esto se debe a que posteriormente DynamicaM realiza una sustitución en el sistema de las ecuaciones algebraicas y todos los valores de las constantes. Se evita de esta forma el uso de variables globales en el sistema.

Por último la función $DefineODE$ permite al usuario definir de manera característica una ecuación diferencial ordinaria. Por ejemplo, el sistema $x^{'}[t] = rx[t]^2$ en DynamicaM se define como:

```
1 Ode = DefineODE[x'[t] == r  x[t]^2]
```

Dentro de la función $DefineODE$ en ecuaciones diferenciales se debe usar un doble igual

(==), esto es debido a que una ecuación diferencial ordinaria representa una igualdad o una restricción y no una asignación.

El conjunto de estas funciones permiten definir un sistema de ecuaciones diferenciales ordinarias mediante la función $DefineSystem$, teniendo dos variantes:

Para un sistema que cuenta con un número considerable de ecuaciones algebraicas o diferenciales y constantes, estas pueden ser definidas mediante las funciones descritas anteriormente y posteriormente conformar un sistema. Un ejemplo del uso de esta función seria:

```
1 DefineSystem[
2   AlgebraicEquation -> {AE1,AE2,AE3},
3   OrdinaryDifferentialEquations -> {ODE1,ODE2},
4   Constants -> {constants},
5   StateVariables->{SV1,SV2,Sv3,SV4},
6   BifurcationVariables -> {BV1,BV2}
7 ]
```

Un sistema pequeño, con pocas ecuaciones diferenciales, algebraicas y constantes, puede ser definido directamente en la función $DefineSystem$, por ejemplo:

```
1 DefineSystem[
2   OrdinaryDifferentialEquations -> {x'[t] == r x[t]^2 + a},
3   Constants -> {a -> 3},
4   StateVariables -> {x},
5   BifurcationVariables -> {r}
6 ]
```

Como se observa en la función no se ha incluido la opción de $AlgebraicEquation$, esto es debido el sistema no cuenta con ecuaciones algebraicas. La implementación de DynamicaM se diseño para proporcionar la flexibilidad de incluir estos elementos.

3.3. Generación de diagramas de bifurcación mediante DynamicaM

En DymanicaM se ha integrado la producción de DB con puntos fijos reales y complejos, mediante técnicas propuestas en los trabajos de investigación de Rodrigo López [López10] y Ana Ochoa [Ochoa16], respectivamente. En ambos trabajos de investigación se propone la producción de DB mediante un algoritmo de optimización inteligente, basados en optimización de enjambre de partículas con nichos, teniendo la capacidad de identificar la estabilidad de los puntos de equilibrio o puntos fijos.

Para integrar dichos trabajos se codificó una API que ofrece acceso a un conjunto de bibliotecas, que tienen como fin la producción de diagramas de bifurcación. La Tabla 3.1 muestra las funciones indispensables para este propósito, incluyendo los parámetros obligatorios y opcionales de cada función.

Tabla 3.1: Funciones indispensables para la creación de diagramas de bifurcación.

Función	Parámetros obligatorios	Parámetros opcionales
DefineODE	ODE	Null
DefineConstants	Constant	Null
DefineAlgebraicEquations	AE	Null
DefineSystem	OrdinaryDifferentialEquations StateVariables BifurcationVariables	Constant AlgebraicEquation
DefineExperiment	System BifRanges	Dir Arguments FileName MetaHeuristic
CreateBifurcationDiagram	Experiment	FixedPointsComplex
PlotBD	BD PlotVariables	Null

Como se describió anteriormente las funciones *DefineOde*, *DefineAlgebraicEquations*, *DefineConstants* y *DefineSystem* se utilizan para definir el sistema de ecuaciones diferenciales, mientras que *DefineExperiment* y *CreateBifurcationDiagram* son para generar los diagramas. Por ultimo *PlotBD* es utilizada para visualizar el DB. El código correspon-

diente a estas funciones se puede observar el el Apéndice A.

Al igual que las funciones descritas en la sección anterior, la función $DefineExperiment$ ocupa reglas de sustitución. Esto para construir una estructura definida, la cual contendrá toda la información indispensable para generar un diagrama de bifurcación. Esta Función se usa para definir un experimento, contando con parámetros opcionales y obligatorios. Como se muestra en la Tabla 3.1 los parámetros opcionales son Dir, $FileName$, $MetaHeuristic$ y $Arguments$. El primero de estos contiene por default el directorio en donde se encuentra guardado el software, $FileName$ contiene el nombre de los archivos que genera el proceso de búsqueda de PSONichos teniendo por default el nombre de "Diagram", $MetaHeuristic$ contiene por default la meta heurística de PSONichos, siendo esta la única por el momento. Por ultimo $Arguments$ que contiene los parámetros indispensables de PSONichos teniendo por default los valores de: NumberOfParticles $\rightarrow$ 100, NumberOfIterations $\rightarrow$ 100, RadiusSize $\rightarrow$ 0.1, SizeNiche $\rightarrow$ 20, C0 $\rightarrow$ 10, C1 $\rightarrow$ 2.1, C2 $\rightarrow$ 2.1, TS $\rightarrow$ 0.9. Los parámetros obligatorios son el sistema definido mediante la función $DefineSystem$ y los rangos de las parámetros de bifurcación.

La función $CreateBifurcationDiagram$ es utilizada para generar el DB, contando con un parámetro obligatorio y uno opcional. El opcional es $FixedPointsComplex$, indicando mediante este si se desea obtener un DB en el plano complejo o no, teniendo por default el valor de "False". Por otro lado la opción obligatoria $Experiment$, contendrá el experimento definido mediante la función $DefineExperiment$. Al igual que las demás funciones ésta ocupa reglas de sustitución en los parámetros que recibe, estas reglas permiten el manejo de parámetros opcionales en cada función, dando de esta forma la flexibilidad de incluir o no algunos parámetros. Cuando la función recibe los parámetros correspondientes, ésta extrae el sistema dinámico y calcula las soluciones o puntos fijos de éste mediante el algoritmo de PSONichos, una vez que termina este algoritmo se filtran los nichos para obtener las partículas que cuenten con una aptitud mayor a un delta (0.9999), considerando a éstas como soluciones o puntos fijos del sistema, posteriormente se calcula la estabilidad de cada punto fijo y se genera un archivo el cual contendrá el DB.

La función $PlotBD$ que cuenta con dos parámetros obligatorios BD, $PlotVaribles$ y un parámetro opcional $Color$. Siendo BD el diagrama de bifurcación generado por $CreateBifurcationDiagram$, $PlotVariables$ las variables que se desean visualizar el en DB, $Color$ contendrá los colores con los que serán marcadas las bifurcaciones. Esta función permite observar los diagramas de bifurcación en distintas perspectivas, esto mediante una función de selección de datos, la cual obtiene del DB los valores correspondientes a cada variable contenida en $PlotVaribles$, retornando estos en el orden definido en $PlotVaribles$. Además permite aplicar alguna función a los datos (i.e. obtener el valor absoluto, obtener la parte real o imaginaria, etc.), esto mediante la adición de la función a las variables contenidas en $PlotVariables$ (v.g. $PlotVariables = \{Re[x], Abs[y], Im[z]\}$). A continuación se presenta un ejemplo del uso de cada una de las funciones descritas en esta sección:

Por ejemplo para el modelo de pesca que es representado por el Sistema (3.1), en él se asume que en ausencia de pesca, la población crece. Los efectos de la pesca son modelados por el término h, el cual nos dice que los peces son capturados de manera constante, esto asume que los pescadores no están preocupados por la población de peces, simplemente pescan el mismo número de peces cada día. Este sistema es tratado por Steven Strogatz [Strogatz94].

$$\dot{x}[t] = rx[t](1 - \frac{x[t]}{k}) - h \tag{3.1}$$

El sistema puede ser reescrito de forma adimensional como se muestra en la Ecuación (3.2).

$$\dot{x}[t] = x[t](1 - x[t]) - h \tag{3.2}$$

Por lo tanto en DynamicaM el DB de dicho sistema es obtenido de la siguiente manera:

```
1  Ode = DefineODE[ODE -> x'[t] = x[t](1-x[t])-h]
2
3  Sistema = DefineSystem[
4            OrdinaryDifferentialEquations -> {ODE},
5            StateVariables -> {x},
```

```
6              BifurcationVariables -> {h}
7             ]
8
9    Experimento = DefineExperiment[
10                    System -> Sistema,
11                    BifRanges -> {{h, -4, 4, 0.01}},
12                    StateVariablesRanges->{{x,-15,15}}
13                  ]
14
15   BifDiagram = CreateBifurcationDiagram[Experiment ->  Experimento]
16
17 PlotBD[
18   BD -> BifDiagram,
19   PlotVaribles-> {h,x}
20     ]
```

Una vez ejecutadas las líneas de código anteriores, se tiene como resultado el DB mostrado en la Figura 3.1.

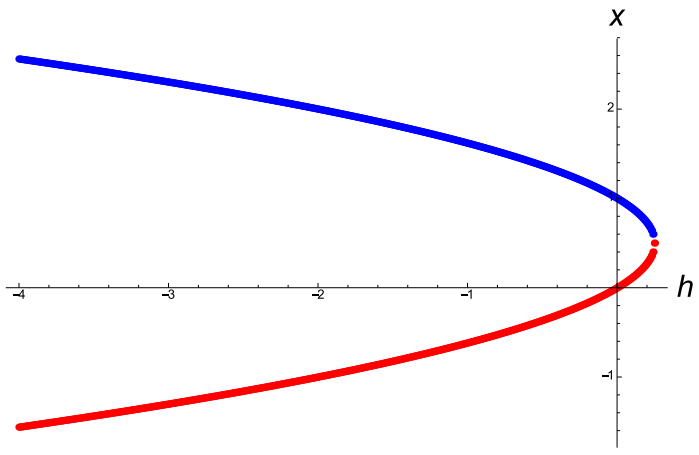


Figura 3.1: Diagrama de bifurcación del Sistema (3.2).

3.4. Identificación de bifurcaciones

En DymanicaM se ha integrado la identificación de los tres tipos de bifurcaciones locales, Silla-Nodo, Pitchfork y Hopf. Esto se hace mediante las tres funciones que se encuentran en la Tabla 3.2, en las cuales se implementaron las condiciones definidas en la Sección 2.3. Estas funciones cuentan con parámetros obligatorios los cuales son: el sistema de ecuaciones diferenciales ordinarias (SystemODE) y el diagrama de bifurcación (DB). Como parámetro opcional, Color indica el color con el cual será marcado el punto de bifurcación. Estas funciones se integraron a la función *PlotBD*, con la finalidad de realizar de manera automática la identificación de los tres tipos de bifurcación.

Por ejemplo, para el sistema $x'[t] = r + x[t]^2$, el cual su DB contiene una bifurcación Silla-Nodo, para que DynamicaM identifique este tipo bifurcación es imprescindible haber generado el DB anteriormente. Una vez que se cuenta con el DB se ejecuta el siguiente comando:

```
1 PlotBD[
2   BD-> BifDiagram,
3   PlotVaribles-> {r,x},
4   Color -> {Saddle-Node-> Green, Pitchfork-> Yellow, Hopf-> Orange}
5   ]
```

Tabla 3.2: Funciones para identificar los tres tipos de bifurcaciones locales.

Nombre de la función	Parámetros obligatorios	Parámetros opcionales
IdentificationSaddleNodeBifurcation	SystemODE BD	Color
IdentificationPitchforkBifurcation	SystemODE BD	Color
IdentificationHopfBifurcation	SystemODE BD	Color

En la Figura 3.2 se observa el resultado de la ejecución de dicho comando, identificando la bifurcación silla-nodo mediante el color verde.

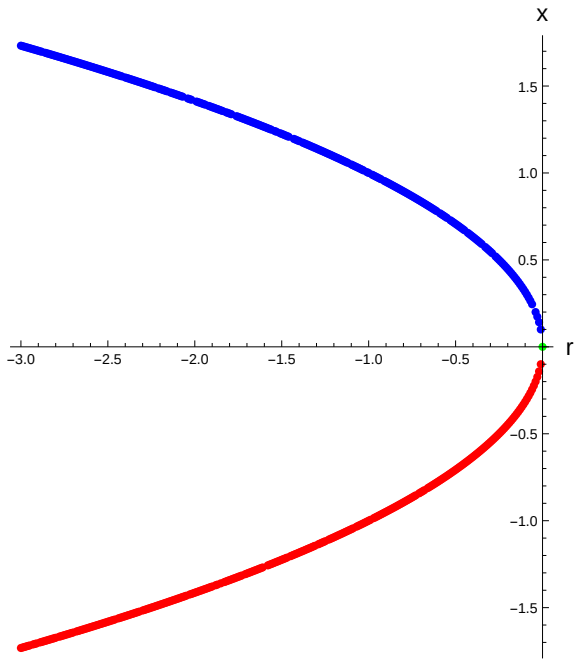


Figura 3.2: Identificación de la Bifurcación Silla-Nodo.

3.5. Obtención de diagramas de fase

Un diagrama de fase es una colección de trayectorias que representan las soluciones de ecuaciones en el espacio de fase. Los diagramas de fase proporcionan información sobre los comportamientos transitorio y asintótico de la solución del sistema [Lerm03].

En DynamicM se integró la función llamada *PhaseDiagram* la cual tiene como propósito la generación de diagramas de fase, los parámetros opcionales y obligatorios para esta función se muestran en la Tabla 3.3. Esta función cuenta con la capacidad de obtener diagramas de fase, secciones de Poincaré y campos vectoriales en un mismo gráfico. En esta sección sólo se abordarán los diagramas de fase.

Para obtener los diagramas de fase DynamicaM sigue el Algoritmo 1, en el cual muestra los pasos generales que realiza la función *PhaseDiagram* para obtener un diagrama

de fase. En este algoritmo lo primero que se hace es obtener la solución del sistema dinámico para cada condición inicial dada por el parámetro *InitialCondition*. Esto mediante el uso de una función propia de Mathematica NDSolve, la cual encuentra una solución numérica del sistema de ecuaciones diferenciales ordinarias. Una vez que son obtenidas las soluciones, la función evaluarSoluciones evalúa cada una de las soluciones en el rango del tiempo definido por *Time* teniendo como resultado una lista que contendrá el resultado de las evaluaciones. Por último la función graficarSoluciones, la cual es la encargada de realizar el gráfico que contendrá el diagrama de fase.

Tabla 3.3: Parámetros de la función *PhaseDiagram*.

PhaseDiagram		
Parámetros obligatorios	Opcionales	Función
SystemODE		Definir sistema
PlotVaribles		Variables a desplegar
InitialCondition		Condiciones iniciales del sistema
BifurcationParameters		Valores del los parámetros de bifurcación
Time		Rango de evaluación
VectorField		
Parámetros obligatorios	Opcionales	Función
ShowVF		Mostrar campo vectorial
Constants		Definir constantes (variables de estado)
EvaluationRanges		Rangos de evaluación
	VectorScale	Escala de los vectores
	VectorPoints	Numero de vectores
PoincareSection		
Parámetros obligatorios	Opcionales	Función
ShowPS		Mostrar mapas de Poincaré
Plane		Definir plano
	Displacement	Desplazamiento del plano

Algoritmo 1 DiagramadeFase(SistemaODE,CondionesIniciales,Tiempo)

1: soluciones ← obtenerSoluciones(SistemaODE,CondionesIniciales,Tiempo)

2: evaluaciones ← evaluarSoluciones(soluciones,Tiempo)

3: Grafico ← graficarSoluciones(evaluaciones)

4: **return** Grafico

Por ejemplo, si se considera el Sistema (3.3), el cual es un ejemplo teórico tratado por Steven Strogatz [Strogatz94].

$$\begin{aligned} w'[t] &= w[t] + e^{-z[t]} \\ z'[t] &= -z[t] \end{aligned} \tag{3.3}$$

Para generar el diagrama de fase se toma en cuenta el conjunto de condiciones iniciales $(w, z) \in ((-5, 3), (1, -1), (1, 1), (-1, 1), (5, 3), (-5, -2), (-5, -3))$; entonces el diagrama de fase se genera mediante DynamicaM con el comando:

```
PhaseDiagram[
  SystemODE -> {w'[t] == w[t] + e^-z[t], z'[t] == -z[t]},
  PlotVaribles -> {w[t],z[t]},
  InicialCondition -> {{w[t], z[t]},{1,-2.5},{-1.5,10},{1,3},
                       {-1,5},{5,3},{-5,-2.5},{-9,-3},{-5,-3}},
  Time -> {t, 1, 5}
  ]
```

Una vez ejecutado dicho comando se tendrá como resultado el diagrama de fase mostrado en la Figura 3.3.

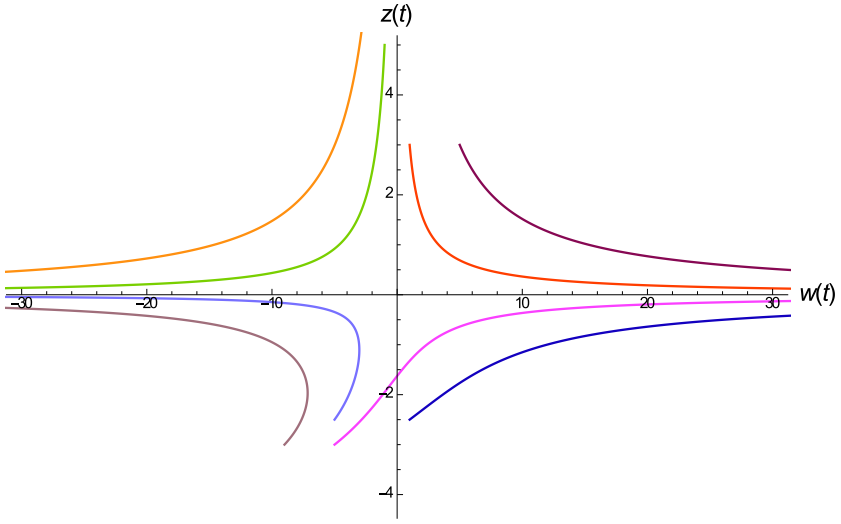


Figura 3.3: Diagrama de fase del Sistema (3.3).

3.6. Obtención de campos vectoriales

Esta herramienta se integró al paquete de DynamicaM, usando las funciones llamadas VectorPlot y VectorPlot3D, las cuales permiten visualizar un campo vectorial mediante el trazado de vectores en un espacio, trazando lineas del flujo, para ilustrar el vector gradiente y la densidad de los vectores. Estas funciones forman parte del software de Mathematica. Debido a los parámetros que requieren estas funciones, se diseñaron dos filtros: transformar sistema y transformar variables. Estos filtros estandarizan la notación, dando la oportunidad de utilizar una notación única. El primer filtro tiene como función realizar una transformación al sistema, toma como parámetro el sistema de ecuaciones diferenciales ordinarias y extrae de éste las ecuaciones algebraicas y diferenciales, aplica a éstas mediante reglas de sustitución una transformación en la cual toma únicamente la parte derecha de la ecuación (Figura 3.4), esto debido a que las funciones VectorPlot y VectorPlot3D requieren esta parte. El segundo filtro Figura 3.5 aplica una transformación a las variables de estado, por ejemplo si un sistema cuenta con tres variables de estado $\{\mu, \delta, \omega\}$, al aplicarle este filtro al sistema las variables de estado son sustituidas por las variables $\{x, y, z\}$, esto es debido a que en el programa que se codifico los rangos de las variables son: $x \in (x_{Min}, x_{Max}),\ y \in (y_{Min}, y_{Max}),\ z \in (z_{Min}, z_{Max})$ en donde x, y, z tienen que ser definidas de manera simbólica, por lo tanto al aplicar este filtro al sistema se evita realizar un cambio a las variables que son utilizadas en estos rangos.

Figura 3.4: Primer filtro implementado

Figura 3.5: Segundo filtro implementado

Esta función cuenta con cinco opciones, las cuales son: $ShowVF$, $VectorScale$, $VectorPoints$, $EvalutionRanges$ y $Constants$. La primera de estas permite visualizar el campo vectorial, mediante los valores True o False. $VectorScale$ permite definir la escala del vector mediante un vector $\{x, y\}$, siendo x lo largo y y la punta del vector; $VectorPoints$ define el numero de vectores que se desean ver en el plano. $EvalutionRanges$ define el área que en la que se desea ver el campo vectorial; por último, la opción $Constants$ permite definir como constante una variable de estado. Por lo tanto para obtener un campo vectorial del Sistema (3.3), en DynamicaM basta con ejecutar la siguiente linea:

```
1 PhaseDiagram[
2   SystemODE -> {w'[t] == w[t] + e^-z[t], z'[t] == -z[t]},
3   PlotVaribles -> {w[t],z[t]},
4   InicialCondition -> {{w[t], z[t]},{1,-2.5},{-1.5,10},{1,3},
5                        {-1,5},{5,3},{-5,-2.5},{-9,-3},{-5,-3}},
6   Time -> {t, 1, 5},
7   VectorField -> {ShowVF -> True,
8   EvalutionRanges -> {{w, -30, 30}, {z, -4, 5}}},
9   ]
```

Una vez que esta línea es ejecutada se tiene como resultado el diagrama de fase y el campo vectorial que se observa en la Figura 3.6.

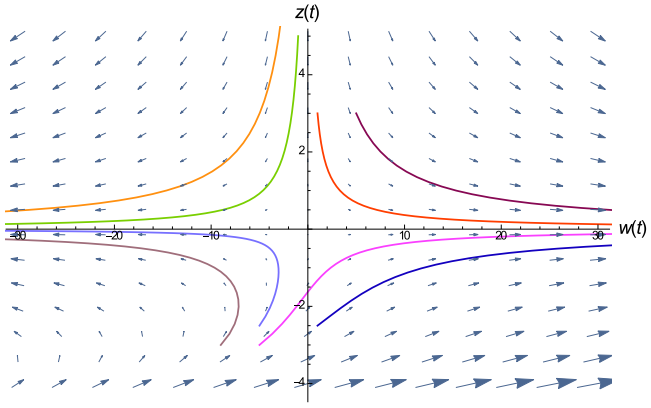


Figura 3.6: Campo vectorial y diagrama de fase del Sistema (3.3)

3.7. Obtención de secciones de Poincaré

Como se menciono anteriormente las secciones de Poincaré son la intersección de una órbita periódica en el espacio del sistema dinámico con un cierto subespacio de menor dimensión.

DynamicaM para facilitar la generación de los mapas de Poincaré cuenta con la función *Phasediagram* que contiene un parámetro alternativo llamado *PoincareSection* tal como se observa en la Tabla 3.3, mediante el cual es posible generar las secciones de Poincaré, teniendo esta función tres opciones, *ShowPS*, *Plane* y *Displacement*. La opción *ShowPS* tiene como alternativa la de asignarle el valor *True* o *False* indicando mediante este si se desea mostrar o no las secciones de Poincaré que serán plasmadas en el plano definido mediante la opción *Plane*. Por otra parte la opción *Displacement* sera un vector $\{x, y, z\}$ el cual indica el desplazamiento que se le aplicara al plano en cada eje. Por lo tanto para generar los mapas de Poincaré la función *Phasediagram*, los obtiene mediante un método propio de Mathematica NDSolve, en el cual se indica que capture un evento (punto) cuando una solución cumple una determinada condición , es este caso en particular se pide que retorne los puntos en donde la trayectoria de una solución para una determinada variable atraviese un plano. Una vez que se tienen estos puntos, se procede a realizar el gráfico.

Para generar los mapas de Poicaré es indispensable que el usuario tenga conocimiento previo del comportamiento que exhibe el sistema, debido a que es imprescindible definir un plano que sea transversal al sistema. Por lo tanto, como se conoce la estructura que presenta el Sistema (3.3), es posible obtener el mapa de Poincaré de éste. Por ejemplo, para realizar un mapa de Poincaré para el Sistema (3.3) se define una recta que se encuentre situada en $x[t] = 20$. Las secciones de Poincaré que se obtendrán serán generadas en DynamicaM mediante el comando:

```
PhaseDiagram[
  System -> {w'[t] == w[t] + E-z[t], z'[t] == -z[t]},
  PlotVaribles -> {w[t],z[t]},
  InicialCondition -> {{w[t], z[t]},{1,-2.5},{-1.5,10},{1,3},
                       {-1,5},{5,3},{-5,-2.5},{-9, -3},{-5,-3}},
  Time -> {t,1,5},
  VectorField -> {True,{{w,-30,30},{z,-3,3}},
  PoincareSection -> {True,x[t]==20}
 ]
```

Ejecutando esta línea se tiene como resultado el diagrama de fase, campo vectorial y secciones de Poincaré (Figura 3.7).

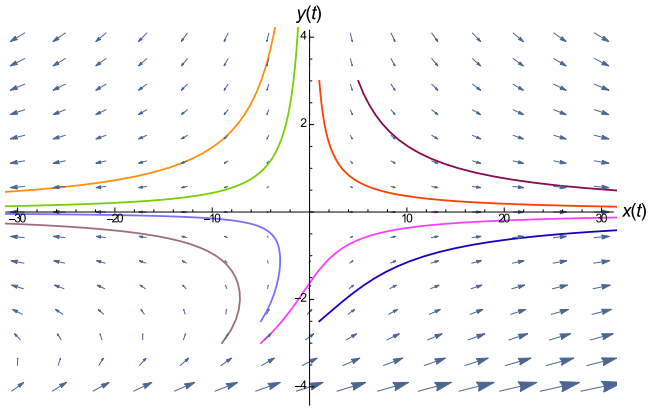


Figura 3.7: Secciones de Poincaré, campo vectorial y diagrama de fase del Sistema (3.3).

3.8. Simulación en el tiempo

DynamicaM ofrece al usuario la función de realizar simulaciones en el tiempo, generando gráficos que describen el comportamiento a lo largo del tiempo que exhibe un sistema bajo ciertos parámetros de operación. Esta función llamada *TimeSimulation* presenta los parámetros mostrados en la Tabla 3.4. En donde el parámetro *System* es el sistema de ecuaciones diferenciales, *PlotVaribles* son las variables de las cuales se desea conocer el comportamiento que presentarán a lo largo del tiempo, *InitialConditions* son los puntos o condiciones iniciales de los cuales partirá la simulación y *TimeRange* es el rango de tiempo en el cual se realizará la simulación.

Por lo tanto esta función comienza obteniendo la solución del sistema dinámico para los parámetros que se desean analizar, esto mediante el método de Mathematica NDSolve. Una vez obtenidas las soluciones, estas son evaluadas en el rango determinado por el parámetro *TimeRange* y posteriormente se realiza el gráfico.

Tabla 3.4: Parámetros que presenta la función *TimeSimulation*.

Paámetros	Función
SystemODE	Definir el sistema
PlotVaribles	Variables a desplegar
InitialConditions	Condiciones iniciales del sistema
TimeRange	Rango del tiempo para la simulación

Por lo tanto, para realizar un análisis en el dominio del tiempo del Sistema (2.15), en DynamicaM basta con ejecutar el siguiente comando:

```
1 TimeSimulation[
2   SystemODE -> {x'[t] == v[t], v'[t] == -(2 v[t] + 3 x[t])/4},
3   PlotVaribles-> {x[t], v[t]},
4   InitialConditions -> {{x[t], v[t]},{20,-20}},
5   TimeRange-> {t,0,20}
6 ]
```

El resultado de ejecutar esta línea es la simulación que se observa en la Figura 3.8. La línea

azul corresponde al parámetro $x[t]$, mientras que la línea de color amarillo corresponde al parámetro $v[t]$.

En este tipo de simulaciones la precisión numérica tiene un papel fundamental al momento de realizarlas, debido a que en diversos casos se ha observado que dependiendo de la precisión utilizada en el sistema, esta puede afectar el resultado obtenido.

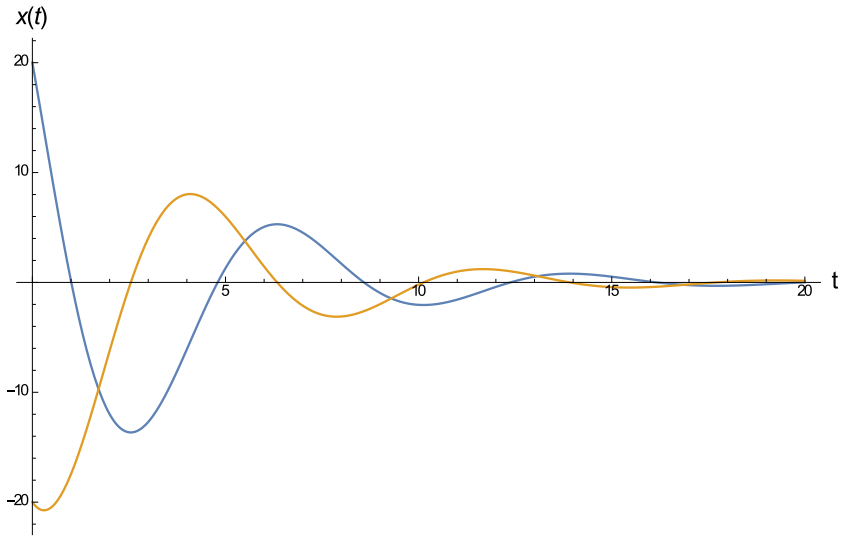


Figura 3.8: Simulación en el tiempo del Sistema (2.15)

3.9. Visualización

Dentro del ámbito de la investigación, específicamente en el área de Ingeniería Eléctrica, se pueden encontrar un número significativo de sistemas de ecuaciones diferenciales, muchos de los cuales cuentan con una dimensionalidad alta. Al generar los diagramas de bifurcación de estos, nos encontramos que no es posible visualizar los resultados obtenidos debido a la alta cantidad de parámetros que se manejan.

Por tal motivo se integró a DynamicaM la función llamada $PlotBD$, la cual tiene como parámetros los observados en la Tabla 3.5. El parámetro $BifurcationDiagram$ es el diagra-

ma de bifurcación obtenido mediante la función $CreateBifurcationDiagram$, $PlotVaribles$ es una lista que contiene los parámetros de bifurcación o las variables de estado que se desean visualizar en el gráfico.

Tabla 3.5: Parámetros que presenta la función $PlotBD$.

Parámetros	Función
$BifurcationDiagram$	Recibe el BD del sistema
$PlotVaribles$	Variables que se desean visualizar

La función realiza una selección de los datos del diagrama de bifurcación, ésta filtra los valores correspondientes a las variables de estado o los parámetros de bifurcación contenidos en el parámetro $PlotVaribles$. Esto facilita la visualización de distintas perspectivas de un mismo diagrama de bifurcación. Así mismo esta función tiene la capacidad de aplicar distintas funciones a los datos seleccionados, es decir el parámetro de $PlotVaribles$ recibirá un vector, el cual contendrá los parámetros que se desean visualizar acotados por la función que se desea aplicar tal y como se muestra a continuación: $\{Re[x], Abs[y], Im[z]\}$.

Por ejemplo, considere el Sistema (3.4) tratado en [Strogatz94], el cual cuenta una variable de estado x, y dos parámetros de bifurcación h y r.

$$x'[t] = h + rx[t] - x[t]^2 \tag{3.4}$$

En DynamicaM el diagrama de bifurcación correspondiente a este sistema es generado mediante los siguientes comandos:

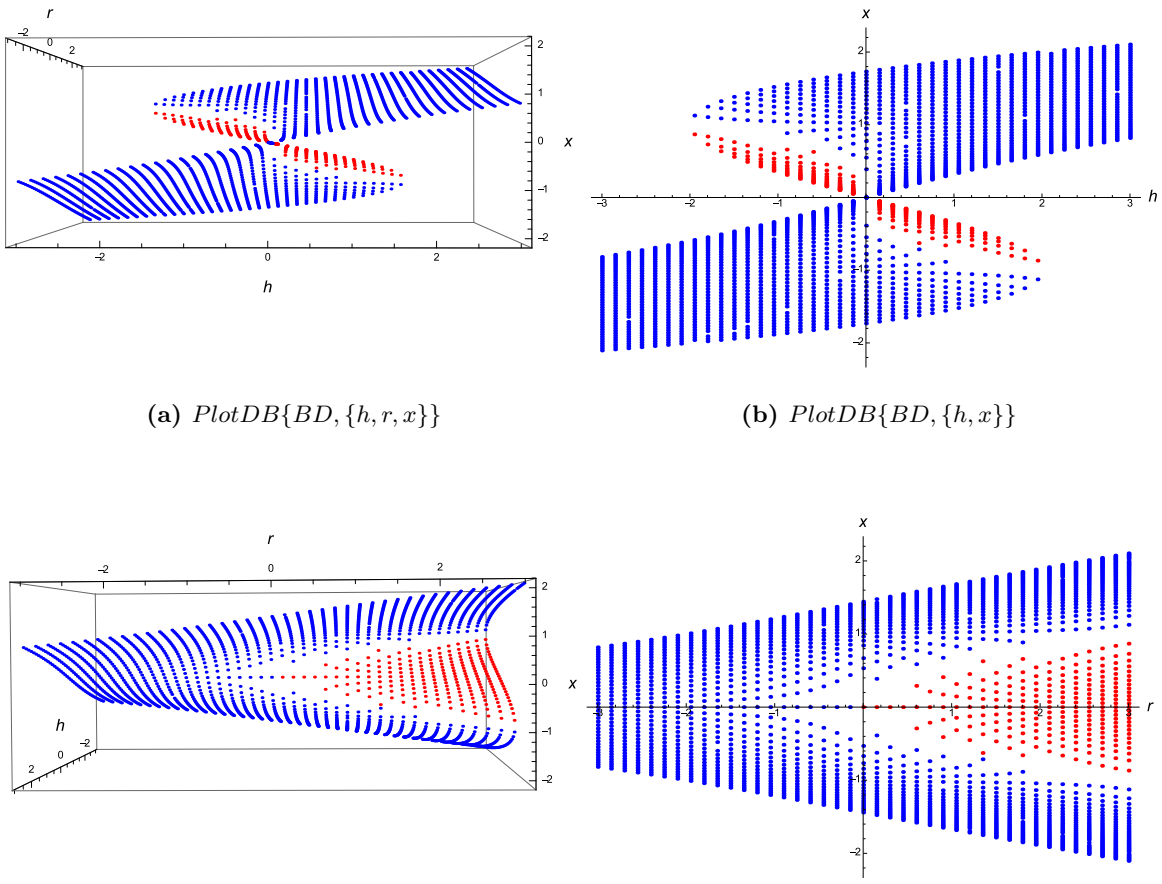
```
1 Sistema = DefineSystem[
2             OrdinaryDifferentialEquations -> {x'[t]= h+rx[t]-x[t
                  ]^2},
3             StateVaribles -> {x},
4             BifurcationVaribles -> {h,r}];
5
6 Experimento = DefineExperiment[
7                 System -> Sistema,
8                 StateVariblesRanges-> {{x,-2,2}},
```

```
9                    BifRanges -> {{h,-3,3,0.05},{r,-3,3,0.05}}];
10
11 BD = CreateBifurcationDiagram [Experiment -> Experimento];
```

Como se observa, el rango de búsqueda para la variable de estado es $x \in (-2, 2)$, mientras que los parámetros de bifurcación son variados en los rangos de $h \in (-3, 3),\ r \in (-3, 3)$.

Una vez obtenido el diagrama de bifurcación del sistema, este es visualizado mediante el comando: $PlotBD[BD, \{h, r, x\}]$. Los ejes x, y y z corresponden al orden definido en la opción $PlotVaribles$.

El DB generado puede ser visualizado en diferentes perspectivas tal y como observa en las Figuras 3.9. En las figuras 3.9(a) y 3.9(c) se observa el diagrama de bifurcación en tres dimensiones, visto desde diferentes ángulos, mientras que en las figuras 3.9(b) y 3.9(d) se realiza una proyección de los diagramas de bifurcación a dos dimensiones, se toman solo los parámetros de $\{h, x\}$ y $\{r, x\}$, respectivamente. Todas las visualizaciones son obtenidas mediante la función $PlotBD$.



(a) $PlotDB\{BD, \{h, r, x\}\}$

(b) $PlotDB\{BD, \{h, x\}\}$

(c) $PlotDB\{BD, \{h, r, x\}\}$

(d) $PlotDB\{BD, \{r, x\}\}$

Figura 3.9: Diferentes perspectivas para el diagrama de bifurcación del Sistema (3.4)

3.10. Formato de archivos

El proceso de generación de los diagramas de bifurcación DynamicaM genera dos tipos distintos de archivos. El primer tipo contiene el último enjambre generado por PSO-Nichos durante el proceso de búsqueda de cada raíz. Este archivo es creado cada vez que termina este proceso, contando con la estructura mostrada a continuación:

```
1  {
2    {BifurcationParameter1,BifurcationParameter2,...,
       BifurcationParametern},
3    {
4      Particle[
5       position[{x1,x2,x3}],
6       velocity[{v1,v2,v3}],
7       aptitude[{fitness}], positionBest[{y1,y2,y3}],
8       aptitudeBest[{Best_fitness}]
9      ],
10     Particle[
11      position[{z1,z2,z3}],
12      velocity[{v4,v5,v6}],
13      aptitude[{fitness}], positionBest[{w1,w2,w3}],
14      aptitudeBest[{Best_fitness}]
15     ]
16   }
17 }
```

Este tipo de archivo contiene el número total de partículas definidas dentro de la función *DefineExperiment*. Cada partícula esta conformada de la misma forma que se describe en la Sub-sección 2.8.2. Además contiene los valores correspondientes a los parámetros de bifurcación (BifurcationParameters).

El segundo tipo de archivo es generado una vez que termina la búsqueda de todas las raíces. Este archivo contendrá el DB correspondiente al sistema dinámico (BifDiagram), las ecuaciones diferenciales ordinarias del sistema (OrdinaryDifferentialEquations), los rangos de los parámetros de bifurcación (BifRanges) y los rangos de búsqueda de las variables de estado (StateVariables). Tal y como se observa a continuación:

```
1  <|
2    OrdinaryDifferentialEquations->SistemODE,
3    BifRanges-> {{X1,X1min,X1max,X1Delta}},
4    StateVariables-> {{Y1,Y1min,Y1max}},
5    BifDiagram-> DB
6  |>
```

Estos archivos son generados con el propósito de evitar la pérdida de información a causa de factores externos que obstaculicen el avance de la determinación de los puntos fijos. En caso de interrumpirse el proceso de búsqueda, DynamicaM (gracias a estos archivos) tiene la capacidad de reanudar la búsqueda de raíces del sistema de ecuaciones diferenciales ordinarias.

3.11. Interfaz gráfica de usuario (GUI)

DynamicaM cuenta con una interfaz gráfica de usuario(GUI), ésta fue creada para la comodidad de los usuarios, evitando de esta manera que los usuarios tengan la necesidad de aprender las funciones imprescindibles para realizar un análisis de un sistema dinámico. Esta interfaz ha sido diseñada con un enfoque reduccionista, un diseño simple y una estructura definida, facilitando de esta forma al usuario, realizar análisis a sistemas dinámicos.

Internamente la GUI se a diseñado con un modelo-vista-controlador (MVC), el cual es una arquitectura de software para la implementación de interfaces de usuario. Esta arquitectura propone la construcción de tres secciones: el modelo, la vista y controlador, es decir, separa la interfaz con la que interactúa el usuario de las funciones o representación de información

que maneja el software. Esta arquitectura se encuentra basada en la reutilización de código y la separación de conceptos, características que facilitan al desarrollador la construcción de la aplicación y posteriormente su mantenimiento.

Por lo tanto la GUI se encuentra conformada por tres secciones principales: Bifurcation Diagram, Dynamical System Analysis y Time Simulation. Estas funciones en conjunto forman la interfaz con la que interactuará el usuario, dichas funciones se presentan en el Apéndice B. El controlador utiliza las mismas funciones descritas en la API (ésta se encuentra descrita el la Sección 3.10).

En la Tabla 3.6 se muestran las funciones con las que cuenta cada sección de la GUI.

Tabla 3.6: Secciones de la GUI

Sección	Funciones
Diagrama de bifurcación	Crear Diagramas de Bifurcación Mostrar diagramas de bifurcación Cargar diagramas de bifurcación
Análisis de sistemas dinámicos	Secciones de Poincaré Campos vectoriales Diagramas de fase
Simulación en el tiempo	Simulaciones en el tiempo

En los siguientes apartados se realiza una descripción de los componentes que integran la GUI.

3.11.1. Diagramas de bifurcación

En esta parte del software se encuentran tres principales funciones: crear, cargar y mostrar diagramas de bifurcación. En las Figuras 3.10 y 3.11 se observa la distribución de éstas dentro de la GUI. En la primer figura se observa la opción de crear y mostrar un diagrama de bifurcación, mientras que en la segunda se presentan las opciones de cargar y visualizar un diagrama de bifurcación generado anteriormente.

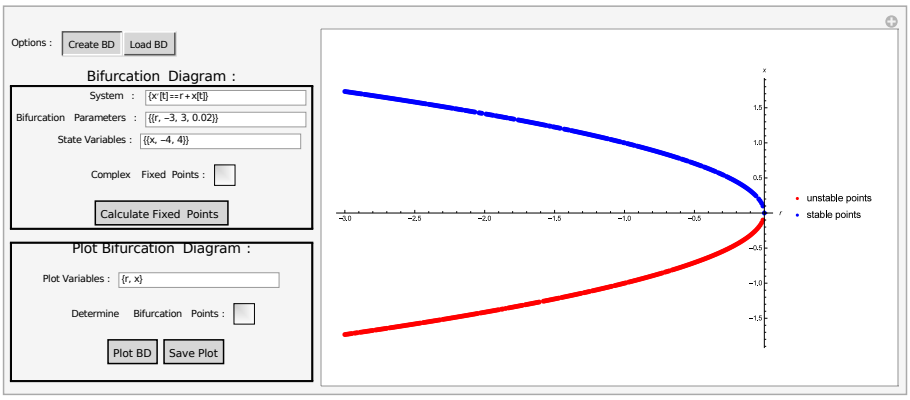


Figura 3.10: Sección crear diagrama de bifurcación

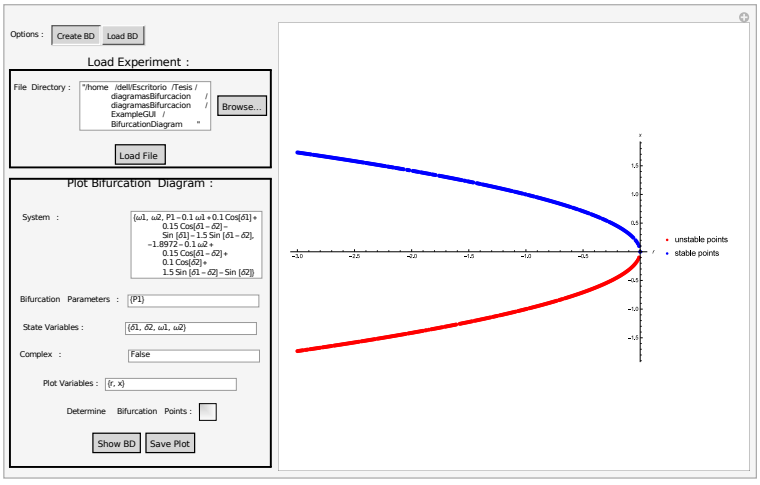


Figura 3.11: Sección cargar diagrama de bifurcación

Crear diagramas de bifurcación

Este apartado como se observa en la Figura 3.10 se encuentra conformado por dos opciones, crear y mostrar diagramas de bifurcación. Se realiza a continuación una descripción detallada de cada una de las partes que conforman a estas opciones:

- Generación de diagramas de bifurcación: Tal como se observa en la Figura 3.12 se encuentra dividida en tres campos fundamentales. El primero de estos tiene como

finalidad definir el sistema de ecuaciones diferenciales ordinarias, el segundo campo permite definir los rangos de los parámetros de bifurcación y el tercero tiene como propósito indicar los rangos de búsqueda de las variables de estado del sistema. Por último se cuenta con un botón, mediante el cual se iniciara el proceso de generación del DB y un checkbox para indicar si se desea o no realizar la búsqueda de puntos fijos complejos.

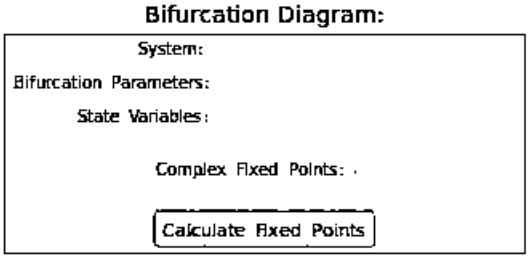


Figura 3.12: Crear diagramas de bifurcación

- Mostrar diagrama de bifurcación: Una vez generado el DB, este puede ser visualizado mediante el apartado mostrado en la Figura 3.13, el cual se encuentra conformada por un área de texto, dos botones y un checkbox. En la área de texto se indican las variables que serán los ejes del gráfico, el primer botón es para mostrar el gráfico correspondiente a las variables definidas en el área de texto, el otro botón es para guardar DB y por ultimo el checkbox tiene como función identificar los tres tipos de bifurcaciones locales.

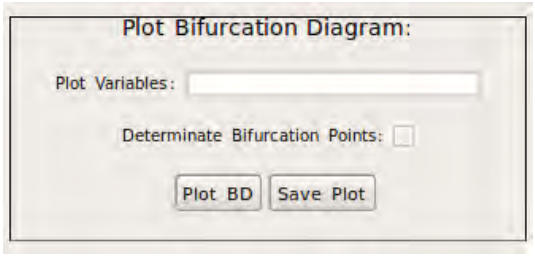


Figura 3.13: Mostrar diagramas de bifurcación

Cargar diagramas de bifurcación

Este apartado permite cargar un diagrama de bifurcación generado anteriormente (Figura 3.11). El cuadro de diálogo se encuentra conformado por dos opciones, cargar y mostrar diagramas de bifurcación. A continuación se realiza una descripción de cada opción:

- Mostrar diagrama de bifurcación: Esta opción es una variante de la descrita la opción de crear diagramas de bifurcación. Se agregan cuatro campos mas. Una vez cargado el diagrama de bifurcación, los datos con los que ha sido generado son obtenidos del archivo y desplegados en estos campos Figura 3.14.

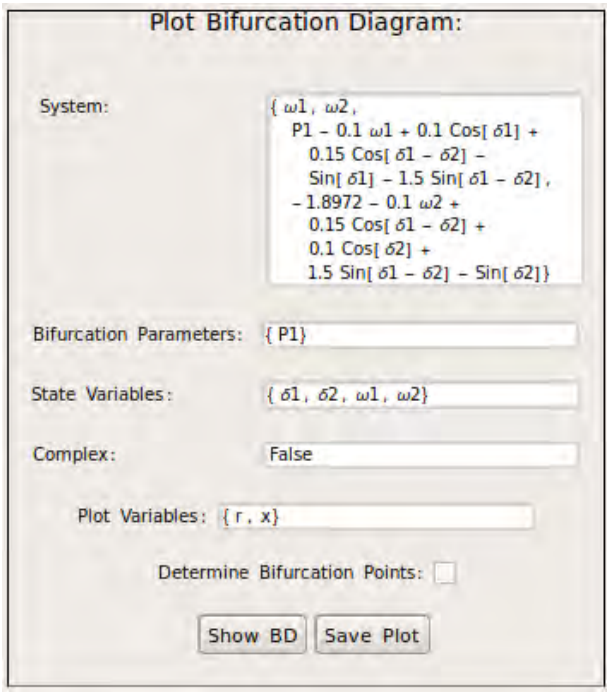


Figura 3.14: Mostrar diagrama de bifurcación

- Cargar diagrama de bifurcación: Como se aprecia en la Figura 3.15, esta opción cuenta con dos botones y una área de texto. El primer botón muestra un explorador de archivos, en el cual se selecciona el diagrama de bifurcación generado con DynamicaM, una vez que es seleccionado el BD, en el área de texto se muestra la ruta en donde se

encuentra el BD. El otro botón tiene la función de cargar el diagrama de bifurcación seleccionado.

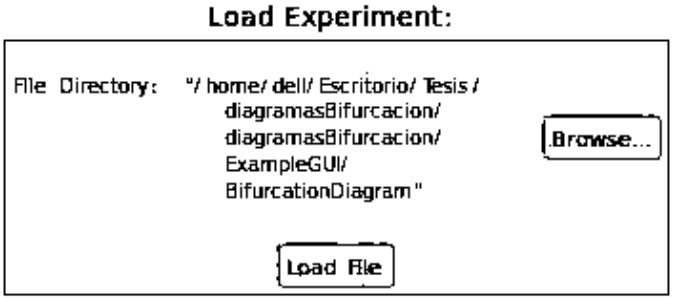


Figura 3.15: Cargar diagramas de bifurcación

3.11.2. Análisis de sistemas dinámicos

En la Figura 3.16, se observa la distribución de la sección de análisis de sistemas dinámicos dentro del GUI. Ésta se encuentra dividida en cuatro partes, definición del sistema dinámico, diagrama de fase, campos vectoriales y secciones de Poincaré.

- Definición de sistema dinámico: Este apartado esta dedicado a definir un sistema dinámico, así como definir las condiciones iniciales de operación del mismo, contando para este propósito con dos campos de texto.
- Diagrama de fase: Una vez definido el sistema dinámico, así como sus condiciones iniciales, para visualizar el campo vectorial sera indispensable definir un rango de evaluación.
- Campos vectoriales: Esta sección es la encargada de generar el campo vectorial del sistema, definiendo únicamente en el área de texto el rango que se desea observar.
- Secciones de Poicaré: Como se menciono anteriormente para obtener las secciones de Poincaré se tiene que conocer la estructura que presenta el sistema en el plano de fase, esto para definir un plano que se encuentre situado transversalmente al flujo. Por lo tanto en este apartado se define dicho plano.

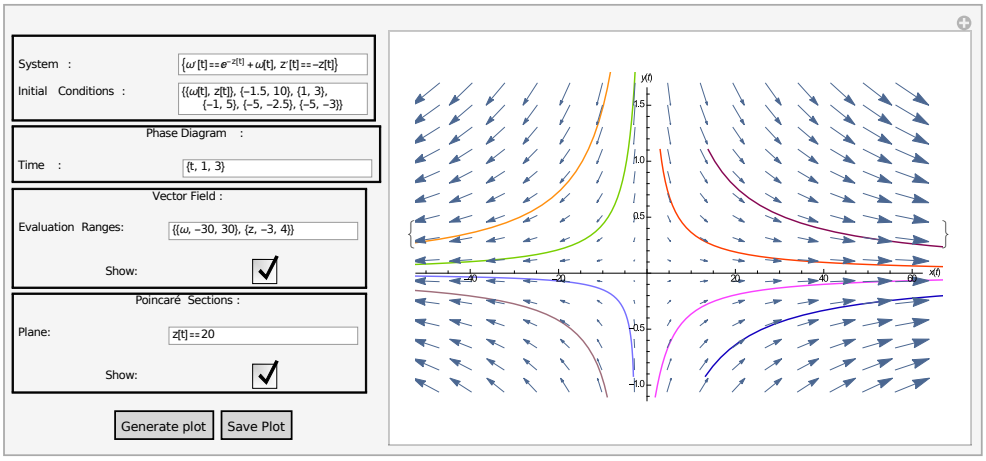


Figura 3.16: Sección análisis de sistemas dinámicos

3.11.3. Simulación en el dominio del tiempo

En la GUI este apartado esta dedicado a generar simulaciones en el dominio del tiempo Figura 3.17, contando este con las secciones descritas a continuación:

- Definir sistema dinámico: En este apartado se define el sistema dinámico ha analizar, contando para este propósito con una sola área de texto en la cual se introducirá el sistema.

- Definir condiciones iniciales: Una vez definido el sistema dinámico, en esta sección se introducirán las condiciones iniciales de operación.

- Precisión: Esta opción tiene como fin la de definir la precisión numérica que será utilizada en los cálculos matemáticos.

- Tiempo de evaluación: En esta sección se definirá el rango de tiempo en el que se realizara la simulación.

- Iniciar y guardar simulación: Para este propósito se cuentan con dos botones, el primero sera para iniciar la simulación y el segundo servirá para guardar la simulación realizada al sistema en distintos formatos como son .png, .eps, .pdf ,etc..

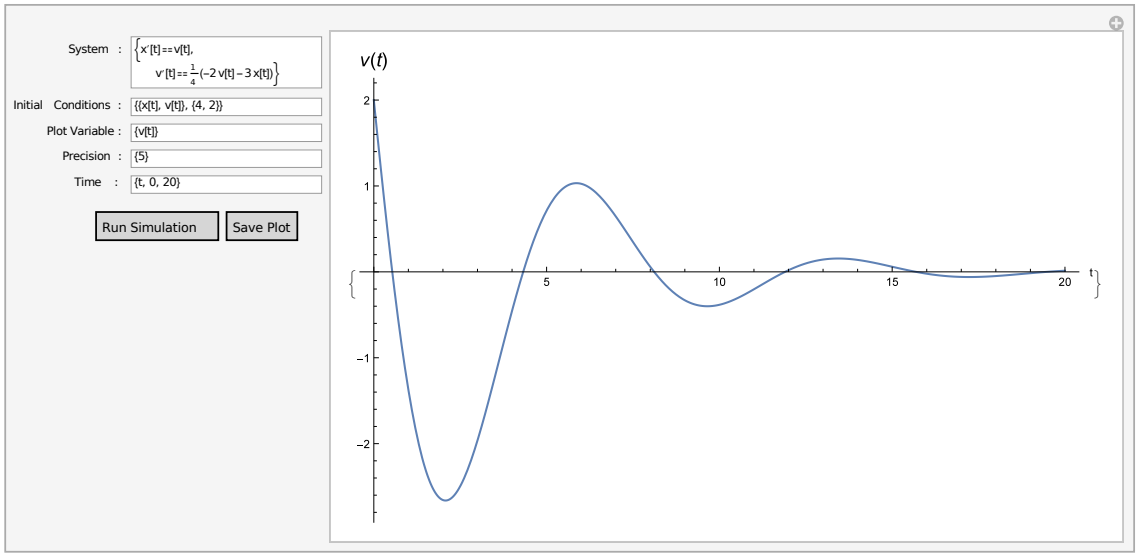


Figura 3.17: Sección simulación en el dominio del tiempo

3.12. Conclusiones del capítulo

En este capítulo se realizó una descripción de cada una de las funciones con las que cuenta DynamicaM. Así mismo se presentaron ejemplos del uso de la API y la GUI, mediante los cuales se da una idea general del funcionamiento de DynamicaM. El siguiente capítulo aborda tres casos de estudios, con los cuales se valida el funcionamiento correcto de DynamicaM.

Capítulo 4

Resultados

En este capítulo se exponen tres modelos de sistemas eléctricos, el primero es el modelo clásico de un sistema eléctrico, el segundo un sistema de dos máquinas y un bus infinito y por ultimo un sistema eléctrico de potencia con carga dinámica tipo II. Se presentan sus diagramas de bifurcación, campos vectoriales, secciones de Poincaré, simulaciones en el tiempo y diagramas de fase obtenidos mediante DynamicaM. Los tres modelos presentados en este capítulo han sido tratados en diversos trabajos de investigación como son [Kasusky02], [Pérez07], por lo que se considera que la reproducción de éstos como un medio de validación para los resultados generados con DynamicaM.

4.1. Caso 1: Modelo clásico de un sistema eléctrico

El modelo clásico de un sistema eléctrico es mostrado en la Figura 4.1. Este sistema está conformado por dos generadores, uno de los cuales es una fuente de voltaje constante cuya dinámica es descrita por medio de la ecuación de oscilación y el otro es considerado como nodo de referencia. El sistema puede describirse como un área conectada a una red modelada como un bus infinito, las líneas de transmisión son admitancias, la carga es modelada como constante, un capacitor y un motor de inducción dependiente de la frecuencia y de la magnitud del voltaje en el nodo de carga. El modelo del motor de inducción es propuesto por [Walve86]. Este sistema está descrito por las ecuaciones algebraicas (4.2) y el conjunto de ecuaciones diferenciales (4.3). El código de DynamicaM con el que se generan

todos los gráficos que se presentan en este caso de estudio se encuentra disponible en el Apéndice C. En esta tesis únicamente se realizo una reproducción de algunos gráficos presentados por Raúl García [Kasusky02], dando una breve explicación de estos. Si se desea observar una análisis mas detallado de este sistema. Éste se encuentra en los trabajos de investigación de Raúl García [Kasusky02] y Ramiro Carvajal [Pérez07].

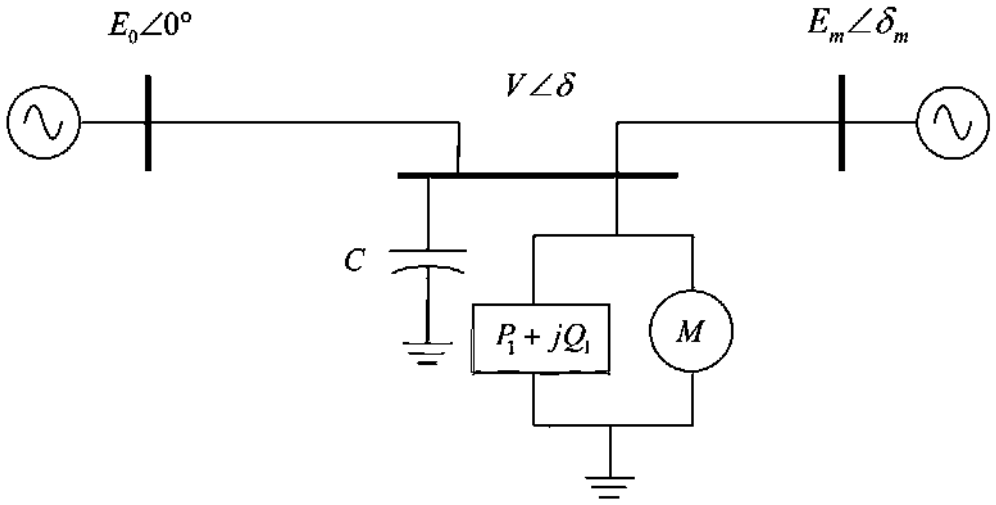


Figura 4.1: Sistema eléctrico de potencia con carga dinámica

La carga dinámica combinada con la carga constante son descritas por las Ecuaciones (4.1).

$$\begin{aligned} P &= P_0 + P_1 + K_{pw}\dot{\delta} + K_{pv}(V + T\dot{V}) \\ Q &= Q_0 + Q_1 + K_{qw}\dot{\delta} + K_{qv}V + K_{qv}V^2 \end{aligned} \tag{4.1}$$

En donde P y Q son las potencias que la carga dinámica demanda al sistema. En su forma polar están descrita por las ecuaciones de flujos de potencia (4.2). La obtención de estas ecuaciones se describe en el trabajo de investigación de Raúl García [Kasusky02].

$$
\begin{aligned}
P &= -VE_0Y_0\sin(\delta+\theta_0) - VE_mY_m\sin(\delta-\delta_m+\theta_m) + V^2(Y_0\sin(\theta_0)+Y_m\sin(\theta_m)) \\
Q &= VE_0Y_0\cos(\delta+\theta_0) + VE_mY_m\cos(\delta-\delta_m+\theta_m) - V^2(Y_0\cos(\theta_0)+Y_m\cos(\theta_m))
\end{aligned} \tag{4.2}
$$

El sistema dinámico está descrito por el conjunto de Ecuaciones (4.3).

$$
\begin{aligned}
\dot{\delta}_m &= \omega \\
\dot{\omega} &= \tfrac{1}{M}(P_m - D_m\omega + E_mVY_m\sin(\delta-\delta_m-\theta_m) + E_m^2Y_m\sin(\theta_m)) \\
\dot{\delta} &= K_{qw}^{-1}(-k_{qv2}V^2 - k_{qv}V + Q - Q_0 - Q_1) \\
\dot{V} &= \tfrac{1}{TK_{qw}K_{pv}}(-K_{qw}(P_0+P_1-P) + (K_{pw}K_{qv} - K_{qw}K_{pv})V \\
&\qquad +K_{pw}(Q_0+Q_1-Q) + K_{pw}K_{qv2}V^2)
\end{aligned} \tag{4.3}
$$

En donde las primeras dos ecuaciones son ecuaciones de oscilación. δ_m es el ángulo que existe entre el campo magnético del rotor y el campo magnético del estator (ángulo de carga). Si este ángulo es negativo el generador consume potencia eléctrica, si es positivo suministra potencia al sistema. E_m es la magnitud del voltaje interno del generador, ω es la variación de velocidad del rotor con respecto a la velocidad de sincronía. Cuando $\omega = 0$ el generador se encuentra en sincronía, cuando $\omega > 0$ el rotor gira a una velocidad mayor que la de sincronía y cuando $\omega < 0$ el rotor gira a una velocidad menor que la de sincronía. δ representa el ángulo de voltaje en el nodo de carga y está medido con respecto al nodo de referencia, V representa la magnitud de voltaje en el nodo de carga y $Q1$ representa la potencia reactiva en el nodo de carga; este es el parámetro de bifurcación. Los ángulos se expresan en radianes y la velocidad en radianes por segundos. Los parámetros restantes se muestran en la Tabla 4.1.

Tabla 4.1: Valores de los parámetros del Sistema (4.3).

Parámetro de bifurcación $Q1$ **de** 10 **hasta** $12pu$			
$k_{pw} = 0.4pu$	$k_{pv} = 0.3pu$	$k_{qw} = -0.3pu$	$k_{qv} = 0 - 2.8pu$
$k_{qv2} = 2.1pu$	$T = 8.1pu$	$P_0 = 0.6pu$	$P_1 = 0pu$
$P_m = 1pu$	$Q_0 = 1.3pu$	$E_0 = 2.5pu$	$E_m = 1pu$
$Y_m = 5pu$	$Y_0 = 8pu$	$M = 0.3seg^2/rad$	$\theta_0 = -0.2094Rad$
$\theta_m = -0.08726Rad$	$Dm = 0.05pu$		

La Figura 4.2 muestra el diagrama de bifurcación correspondiente al Sistema (4.3). En esta figura se observan los parámetros de $Q1$ contra V; los puntos azules corresponden a la región estable, mientras que los puntos rojos representan la región inestable del sistema. Este diagrama fue generado mediante DynamicaM para el parámetro de bifurcación $Q1 = 10$ hasta $Q1 = 12$ dentro de los rangos de búsqueda mostrado en la Tabla 4.2.

Tabla 4.2: Valores de los rangos de búsqueda definidos para el Sistema (4.3).

Parámetro	Rangos de búsqueda	Parámetro	Rangos de búsqueda
δ_m	$\{0, 2\}$	δ	$\{0, 2\}$
V	$\{0, 3\}$	ω	$\{-1, 2\}$

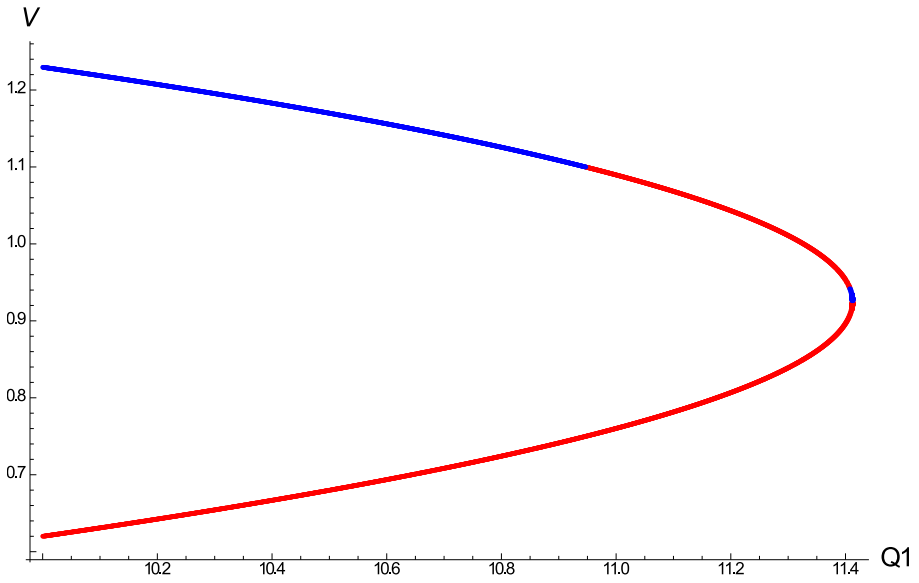
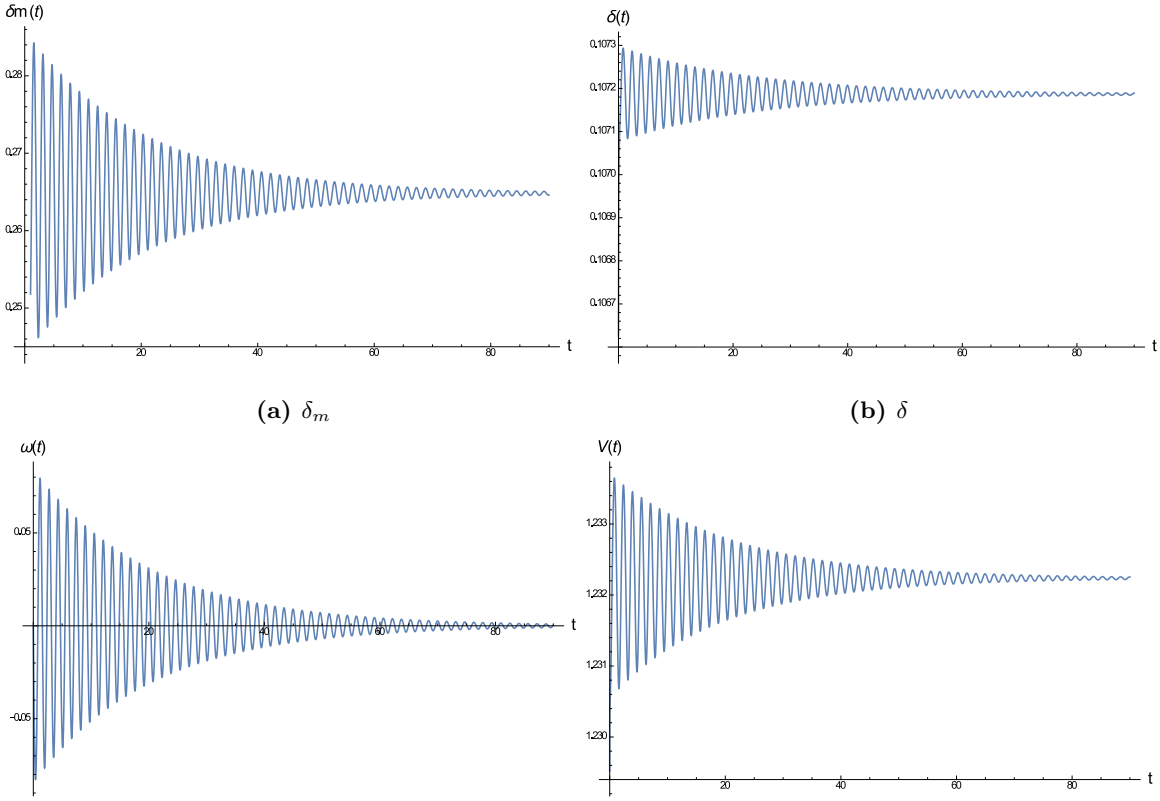


Figura 4.2: Diagrama de bifurcación del Sistema (4.3).

La dinámica de las variables de estado convergerá hacia algún punto de equilibrio, si sus condiciones iniciales son definidas dentro de la región estable [Nayfeh95]. Para mostrar lo mencionado anteriormente, se toma un punto de equilibrio ($\{\delta_m = 0.2858, \delta = 0.10662, \omega = 0, V = 1.2295\}$) el cual se encuentra en la primer región estable del DB y se realiza una simulación en el dominio del tiempo; mediante esta simulación se observa que las variables de estado llegan a un punto de equilibrio estable, conforme el tiempo tiende a infinito $t \to \infty$ Figura 4.3.



(a) δ_m **(b)** δ

(c) ω **(d)** V

Figura 4.3: Comportamiento de las variables del sistema cuando $Q1 = 10$

Tomando el punto $\{Q1 = 10.886, \delta_m = 0.7, \delta = -0.5, \omega = 0.3, V = 0.85\}$ que se encuentra cerca de un punto crítico en una región de biestabilidad del diagrama de bifurcación, explicando la aparición de esta región se realiza por Raúl García [Kasusky02], es posible observar mediante un diagrama de fase el comportamiento que presentara el sistema bajo estos parámetros de operación. En este punto el sistema experimenta oscilaciones de amplitud variable de forma aperiódica, este comportamiento es llamado caos. La Figura 4.4 muestra el diagrama de fase para esta condición inicial, se observa que la trayectoria caótica que describen las variables en $Q1 = 10.886$ pu, es un atractor extraño. Un atractor extraño es aquel que presenta oscilaciones de forma irregular, las cuales nunca se repiten exactamente pero éstas siempre permanecen en una región limitada del espacio de fase. Debido al comportamiento tan complejo que presenta el atractor extraño esta forma de operación es

muy peligrosa.

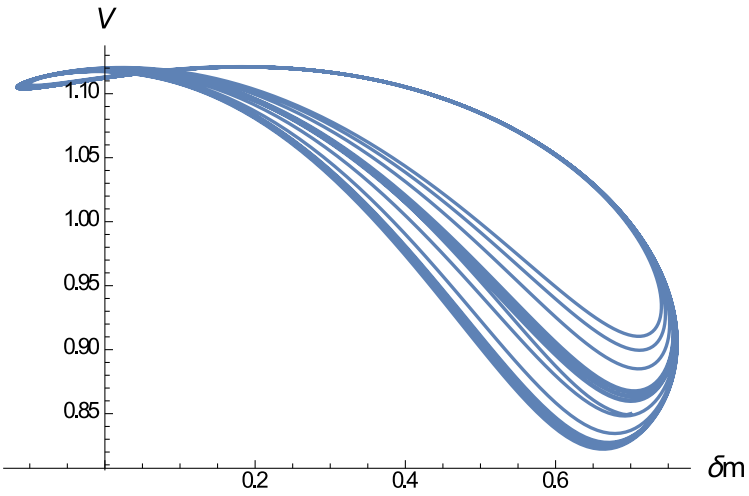


Figura 4.4: Atractor extraño generado en $Q1 = 10.886$.

En la Figura 4.5 se muestra la dinámica del voltaje en el nodo de carga, para el punto de operación en donde aparece este atractor. Estas oscilaciones caóticas permanecen si no se cambia el parámetro de bifurcación.

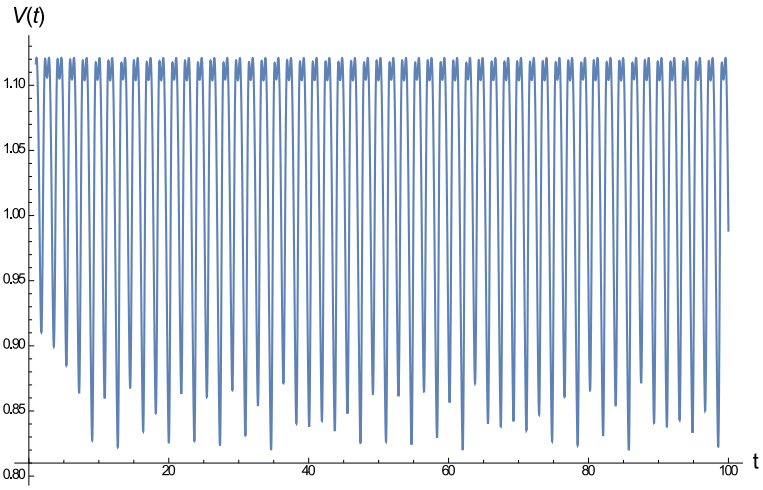


Figura 4.5: Voltaje en el nodo de carga.

En la Figura 4.6 se observa el atractor extraño el cual es ocasionado por bifurcaciones de doble periodo en cascada en la región biestable. Esto se explica en el trabajo de investigación de Raúl García [Kasusky02]. Además se observa el campo vectorial que presenta el sistema en esta región, mediante cual podemos confirmar que cualquier condición inicial que se encuentre cerca de este atractor ésta sera atraída por el. Debido a que en la Figura 4.6, no se aprecian que los vectores son tangentes a las trayectorias. Por lo tanto en la Figura 4.7 presenta un acercamiento a la región $0.5 \leq \delta_m \leq 0.7, -1.5 \leq \omega \leq 0, 0.8 \leq V \leq 0.9$ en la cual se aprecia que el campo vectorial es el correspondiente al sistema dinámico.

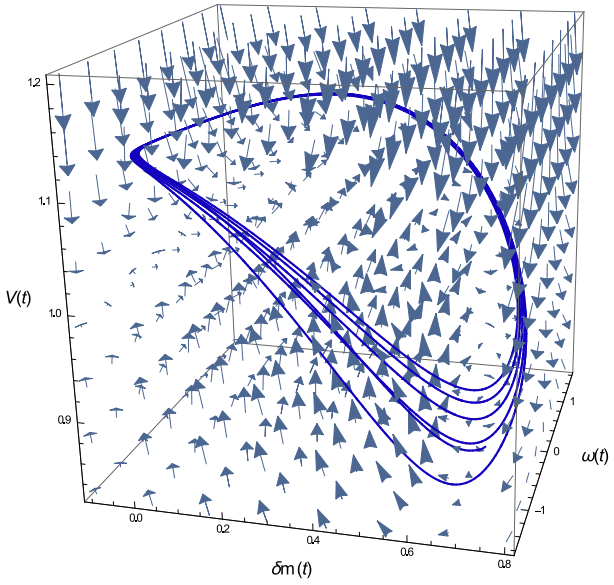


Figura 4.6: Atractores extraños que presenta el sistema.

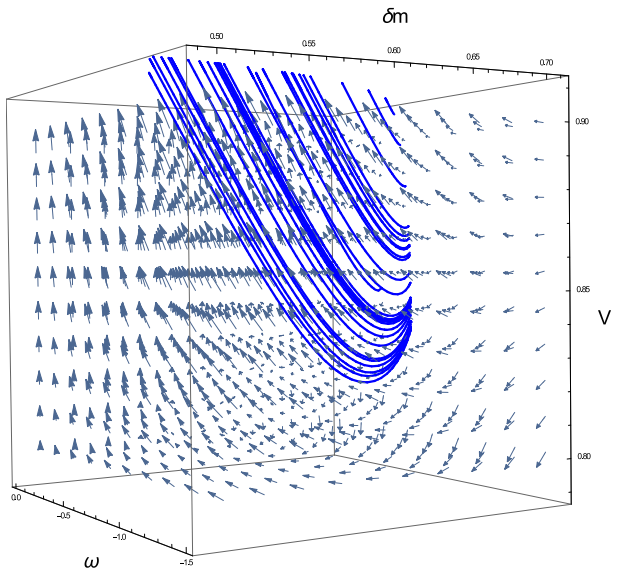


Figura 4.7: Sección del campo vectorial.

Conociendo la estructura que presenta el sistema en el plano de fase es sencillo obtener las secciones de Poincaré. Se define un plano transversal al flujo $\omega[t] = -1.5V[t] + \delta_m[t] + 1$, tal como se observa en la Figura 4.8. En este plano se proyectaran las lineas de flujo que lo atraviesan. La sección de Poincaré definida por este plano se observan en la Figura 4.9.

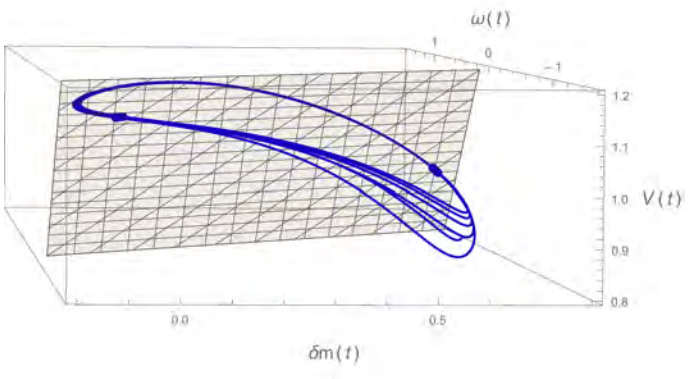


Figura 4.8: Plano transversal al primer atractor.

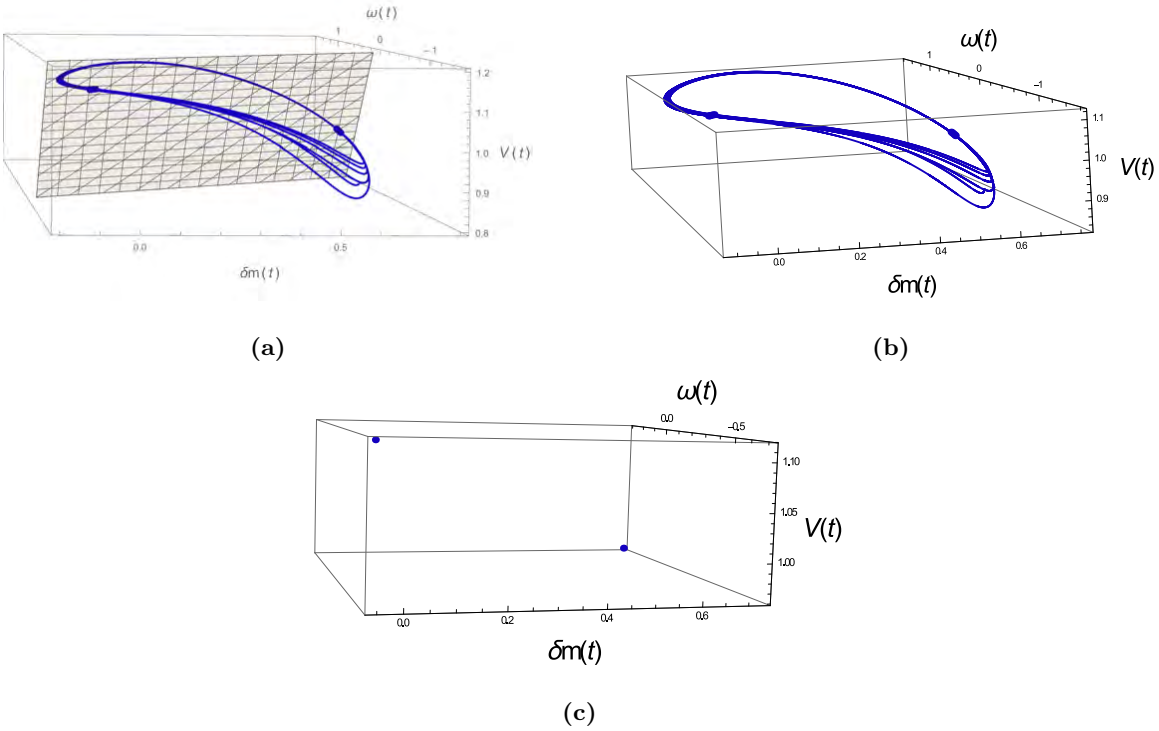


Figura 4.9: Mapa de Poincaré del primer atractor

Para este caso de estudio se a tomado un punto del diagrama de bifurcación el cual se comporta como un atractor extraño. Esto se ha comprobando mediante la generación del campo vectorial en esta zona, con el cual es posible observar la dirección que toman los vectores, los cuales nos confirman que cualquier punto que se encuentre en en la cercanía sera atraído a éste. De igual forma se realizaron simulaciones en el tiempo, tomando dos puntos. El primero se encuentra en la parte estable del sistema y la simulación comprueba la estabilidad del mismo. El segundo se encuentra cerca de un punto crítico en una región de biestabilidad, la simulación en el tiempo muestra el caos que presenta el sistema en este punto.

4.2. Caso 2: Sistema eléctrico de potencia con carga dinámica tipo II

Este sistema eléctrico es parecido al presentado en el caso 1, tiene como variante un generador de menor capacidad, aproximadamente de 500 MW. El capacitor que se encuentra conectado al nodo de carga ahora cuenta con un valor de $c = 3.5\ pu$, la línea de transmisión que conecta el nodo de carga con el bus de referencia es de mayor longitud, considerándola puramente inductiva. Con estas modificaciones el sistema dinámico queda descrito mediante el Sistema (4.4), expresado en forma rectangular [Wang94]. El código de DynamicaM con el que se generan todos los gráficos que se presentan en este caso de estudio se encuentra disponible en el Apéndice C.

$$
\begin{aligned}
&\dot{\delta}_m = \omega \\
&\dot{\omega} = \tfrac{1}{M}(P_m - D_m\omega - (VE_m(-G_m\cos(\delta-\delta_m) - (-B_m)\sin(\delta-\delta_m)) + E_m^2 G_m)) \\
&\dot{\delta} = K_{qw}^{-1}(-k_{qv2}V^2 - k_{qv}V + Q - Q_0 - Q_1) \\
&\dot{V} = \tfrac{1}{TK_{qw}K_{pv}}(-K_{qw}(P_0 + P_1 - P) + (K_{pw}K_{qv} - K_{qw}K_{pv})V + K_{pw}(Q_0 + Q_1 - Q) \\
&+K_{pw}K_{qv2}V^2) \\
&P = -(VE_0^{'}(-G_0\cos(-\delta) - (-B_0)\sin(-\delta)) + VE_m(-G_m\cos(\delta_m-\delta) - (-B_m) \\
&\sin(\delta_m-\delta)) + V^2(G_0+G_m)) \\
&Q = -(-VE_0^{'}(-G_0\sin(-\delta) + (-B_0)\cos(-\delta)) - VE_m(-G_m\sin(\delta_m-\delta) + (-B_m) \\
&\cos(\delta_m-\delta)) - V^2(B_0+B_m))
\end{aligned}
\tag{4.4}
$$

El parámetro de bifurcación del sistema es la potencia reactiva denotada por Q_1, el cual varía de 2.5 pu hasta 3.1 pu, los valores restantes del sistema son mostrados en la Tabla 4.3.

Tabla 4.3: Valores de los parámetros del Sistema (4.4).

Parámetro de bifurcación $Q1$ **de** $2.5pu$ **hasta** $3.1pu$			
$k_{pw} = 0.4pu$	$k_{pv} = 0.3pu$	$k_{qw} = -0.3pu$	$k_{qv} = -2.8pu$
$k_{qv2} = 2.1pu$	$T = 8.5pu$	$P_0 = 0.6pu$	$P_1 = 0pu$
$P_m = 1pu$	$Q_0 = 1.3pu$	$E_0 = 19.588pu$	$E_m = 1.05pu$
$M = 0.01464seg^2/rad$	$G_m = 0pu$	$G_0 = 0pu$	$B_m = -5pu$
	$B_0 = -0.17pu$		

El diagrama de bifurcación correspondientes a los parámetros V y $Q1$ del Sistema (4.4), es

mostrado en la Figura 4.10. Al igual que en el caso anterior, los puntos azules representan la región de estabilidad del sistema, mientras que los rojos corresponden a la región inestable del mismo. En este diagrama al variar la potencia reactiva en $Q1 = 2.5$ pu el sistema esta en un estado estable hasta la potencia toma el valor $Q1 = 2.98$ pu. Después de este punto el sistema pierde su estabilidad en $Q1 = 2.9889$ pu. Este diagrama fue generado mediante DynamicaM, para el parámetro de bifurcación Q_1, variado este de 2.1 *pu* hasta 3.1 *pu* con un incremento de 0.0005, dentro de los rangos de búsqueda de las variables de estado mostrados en la Tabla 4.4.

Tabla 4.4: Rangos de búsqueda de las variables de estado, para el Sistema (4.4).

Parámetro	Rangos de búsqueda	Parámetro	Rangos de búsqueda
V	$\{0, 2\}$	δ_m	$\{0, 2\}$
ω	$\{-1, 2\}$	δ	$\{0, 3\}$

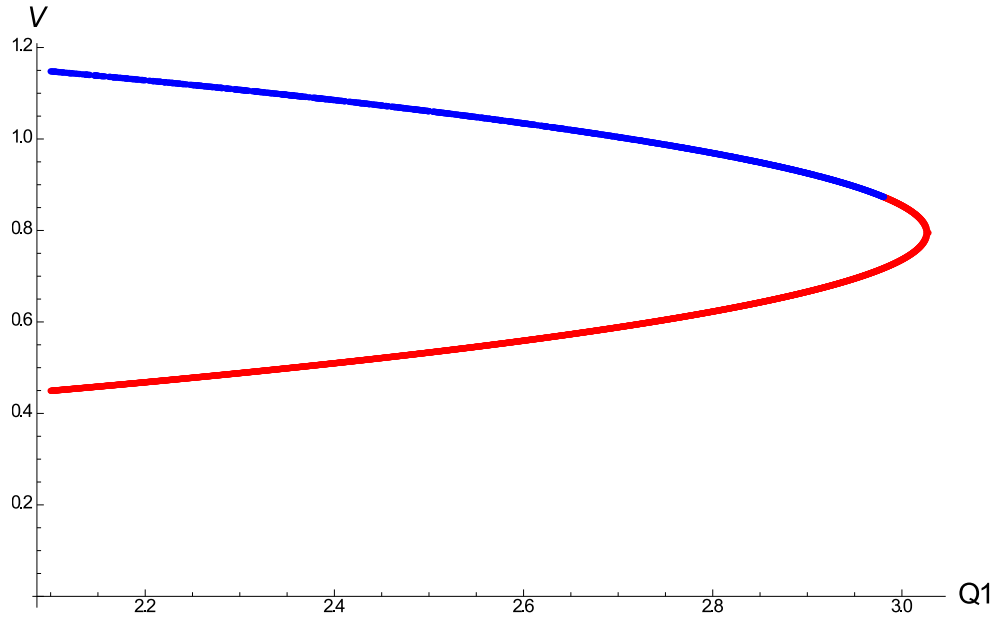
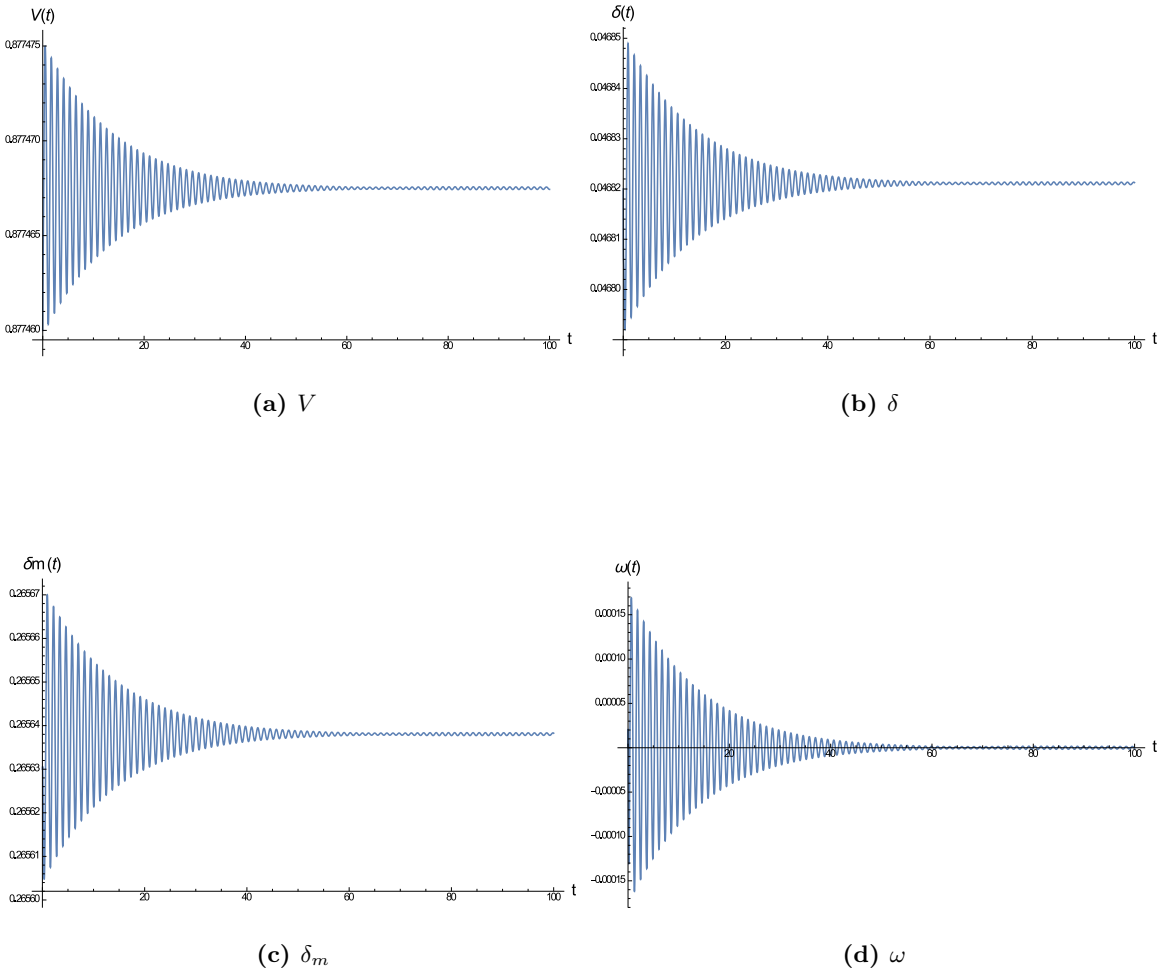


Figura 4.10: Diagrama de bifurcación del Sistema (4.4) para los parámetros $Q1$, V.

Partiendo de la región estable del sistema, se simula la dinámica que presenta el sistema en el punto $Q_1 = 2.975$, $V = 0.87746$, $\delta_m = 0.26563$,$\omega = 0$, $\delta = 0.04682$, el cual corresponde a un punto estable en el diagrama de bifurcación (Figura 4.10). La Figura 4.11 muestra la dinámica que presentan las variables del sistema en este punto de bifurcación, como se observa si el sistema se encuentra en este punto de operación las variables llegaran a un punto de equilibrio conforme el tiempo tiende a infinito ($t \to \infty$).



(a) V **(b)** δ

(c) δ_m **(d)** ω

Figura 4.11: Comportamiento de las variables del sistema cuando $Q1 = 2.975$

Esto se puede reafirmar mediante el diagrama de fase Figura 4.13 y el campo vectorial

Figura 4.12 del sistema en esta región. Por lo tanto como se observa en la Figura 4.12 y con ayuda del campo vectorial, se puede decir que la linea de flujo que genera este punto llegara a un punto de equilibrio conforme el tiempo transcurre. Confirmando mediante esto la estabilidad de el punto fijo. Debido a que en la Figura 4.12 no se aprecia que los vectores son tangentes a la linea de flujo, se realizó un acercamiento a la zona comprendida por $0.874 \leq V \leq 0.877$, $0.40 \leq \delta_m \leq 0.55$, $-0.01 \leq \omega \leq -0.04$. Este acercamiento se aprecia en la Figura 4.14.

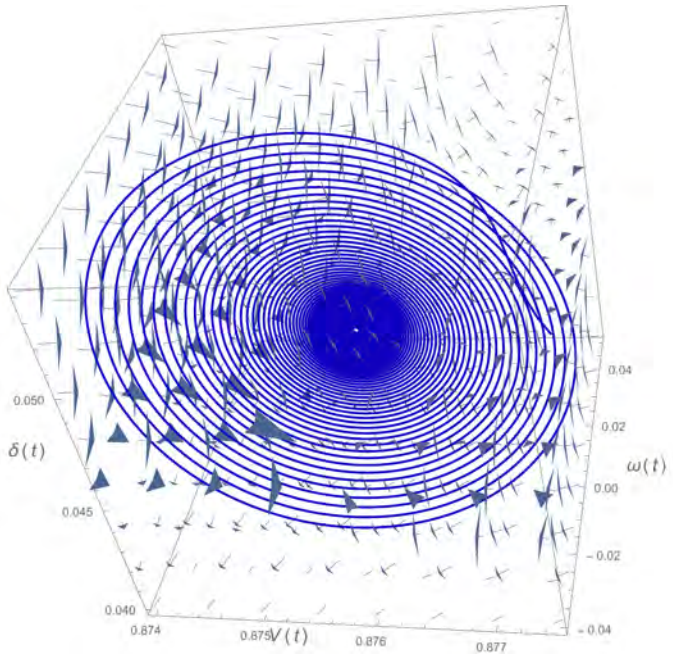


Figura 4.12: Campo vectorial del Sistema (4.4).

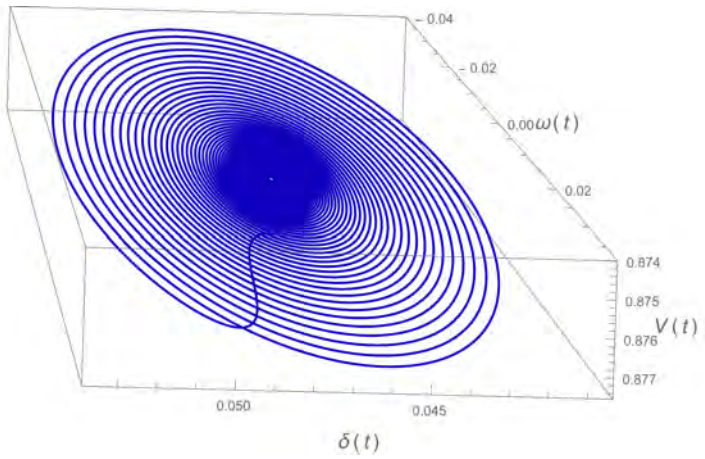


Figura 4.13: Diagrama de fase para $Q_1 = 2.975$.

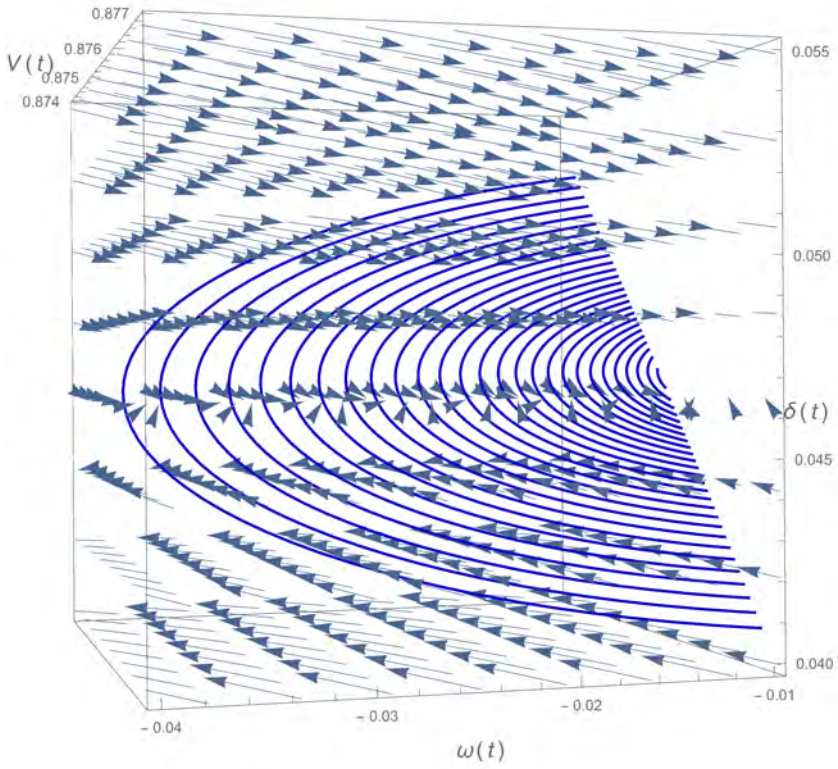


Figura 4.14: Campo vectorial del Sistema (4.4), para las sección comprendida por $0.874 \leq V \leq 0.877$, $0.40 \leq \delta_m \leq 0.55$, $-0.01 \leq \omega \leq -0.04$.

Una vez que se conoce la estructura que presenta este punto en el espacio de fase es posible definir un plano que sea transversal al flujo para obtener las secciones de Poincaré. Para este particular caso se han obtenido los mapas de Poincaré para un plano que se encuentra situado en $\delta = 0.045$, obteniendo como resultado el observado en la Figura 4.15.

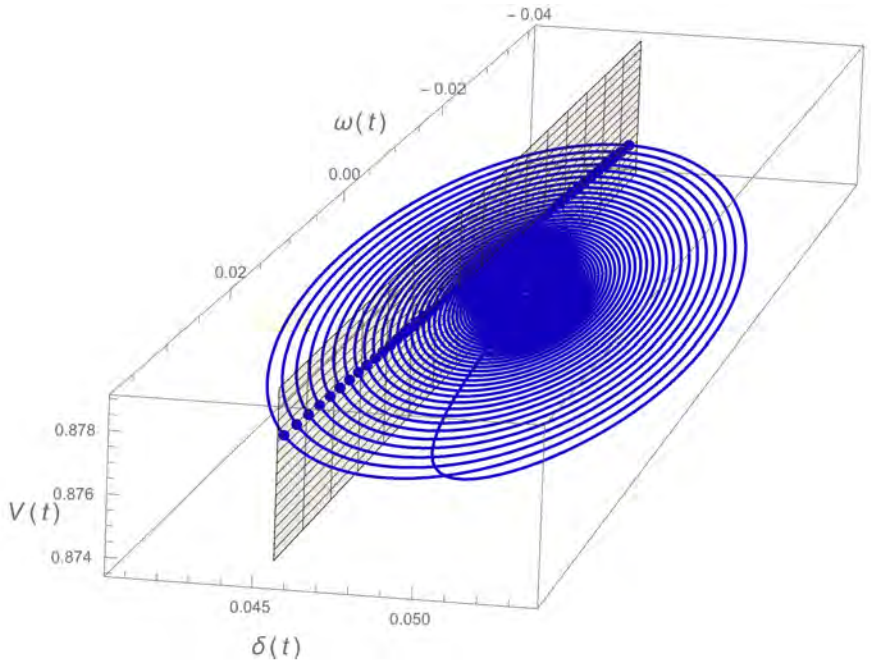


Figura 4.15: Mapa de Poincaré para un plano situado en $\delta[t] = 0.045$.

En esta tesis únicamente se realizo una reproducción de algunos gráficos presentados por Raúl García [Kasusky02], dando una breve explicación de estos. Si se desea observar una análisis mas detallado de este sistema. Éste se encuentra en el trabajo de investigación de Raúl García [Kasusky02].

4.3. Caso 3: Sistema de dos máquinas y un bus infinito

El modelo más simple de un sistema de potencia es el de una máquina síncrona conectada a un bus infinito, El diagrama eléctrico del sistema de dos máquinas - bus infinito es mostrado en la Figura 4.16 [Bilir00]. Debido a que presenta dos máquinas se espera observar un comportamiento mas complejo.

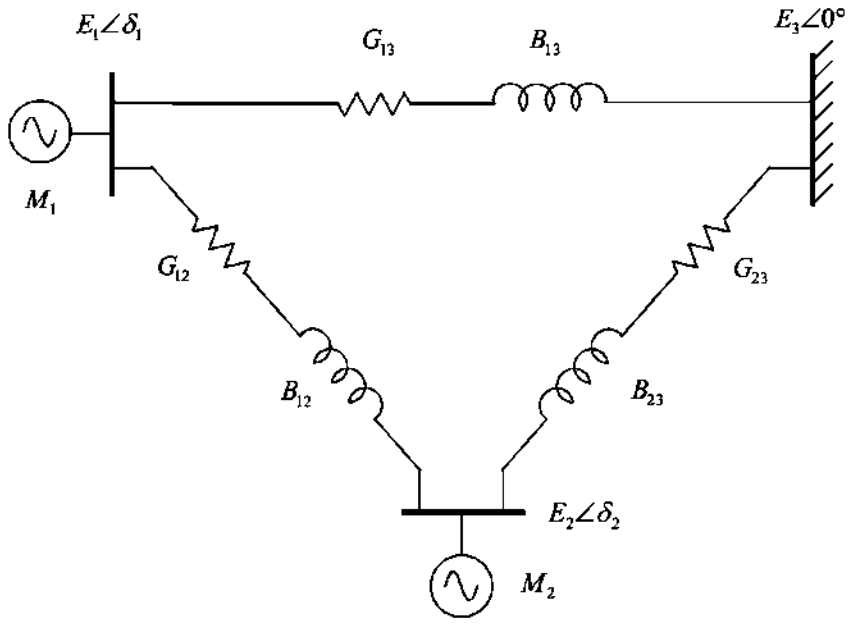


Figura 4.16: Sistema eléctrico de dos máquinas y un bus infinito

El sistema dinámico está descrito por el conjunto de Ecuaciones (4.5).

$$
\begin{aligned}
\dot{\delta}_1 &= \omega_1 \\
\dot{\delta}_2 &= \omega_2 \\
\dot{\omega}_1 &= \tfrac{1}{M_1}[P_1 - Dm_1\omega_1 - E1E2(B12)\sin(\delta_1-\delta_2) - E1E3(B13)\sin(\delta_1) \\
&\qquad -E1E2(G12)\cos(\delta_1-\delta_2) - E1E3(G13)\cos(\delta_1)] \\
\dot{\omega}_2 &= \tfrac{1}{M_2}[P_2 - Dm_2\omega_2 - E2E1(B12)\sin(\delta_2-\delta_1) - E2E3(B23)\sin(\delta_2) \\
&\qquad -E2E1(G12)\cos(\delta_2-\delta_1) - E2E3(G23)\cos(\delta_2)]
\end{aligned}
\tag{4.5}
$$

En este caso la potencia mecánica $P1$ es el parámetro de bifurcación, el cual es variado de 1 hasta 3 pu. El resto de los parámetros utilizados en este modelo se encuentran listados en la Tabla 4.5.

Tabla 4.5: Valores de los parámetros del Sistema (4.5).

Parámetro de bifurcación $P1$ **de** 1 **hasta** $3pu$		
$M_1 = 1seg^2/rad$	$Dm_1 = 1seg/rad$	$E_1 = 1pu$
$M_2 = 1seg^2/rad$	$P_2 = -1.8972pu$	$Dm_2 = 1seg/rad$
$E_2 = 1pu$	$G_{12} = -1.43pu$	$B_{12} = 1pu$
$G_{13} = -0.3$	$B_{13} = 3pu$	$G_{23} = -0.125pu$
$E_3 = 1pu$	$B_{23} = 1.25pu$	

Como se observa en las Figuras 4.17 y 4.18, se encuentran los diagramas de bifurcación del Sistema (4.5) para los parámetros $\{\delta_1, \omega_1\}$ y $\{\delta_2, \omega_2\}$ respectivamente. En estos diagramas la región azul representa los puntos estables y la región roja los inestables. Dichos diagramas fueron obtenidos mediante DynamicaM, para el parámetro de bifurcación P_1 de 1 hasta $3pu$ con un incremento de 0.01, dentro de los rangos de búsqueda mostrado en la Tabla 4.6.

Tabla 4.6: Valores de los rangos de búsqueda definidos para el Sistema (4.5)

Parámetro	Rangos de búsqueda	Parámetro	Rangos de búsqueda
δ_1	$\{-2, 3\}$	δ_2	$\{-2, 3\}$
ω_1	$\{-2, 3\}$	ω_2	$\{-2, 3\}$

Partiendo del estado estable, se ha simulado la dinámica del sistema para $P_1 = 2.26pu$, $\delta_1[t] = 1.06551$, $\delta_2[t] = -0.30654$, $\omega_1[t] = 0$, $\omega_2[t] = 0$, donde este es un punto de equilibrio

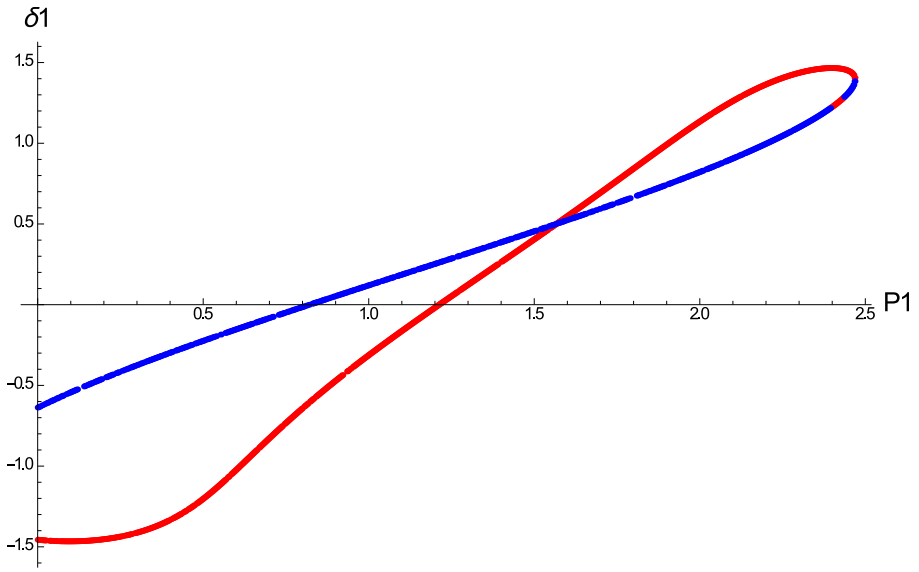


Figura 4.17: Diagrama de bifurcación del Sistema (4.5), para los parámetros δ_1, ω_1.

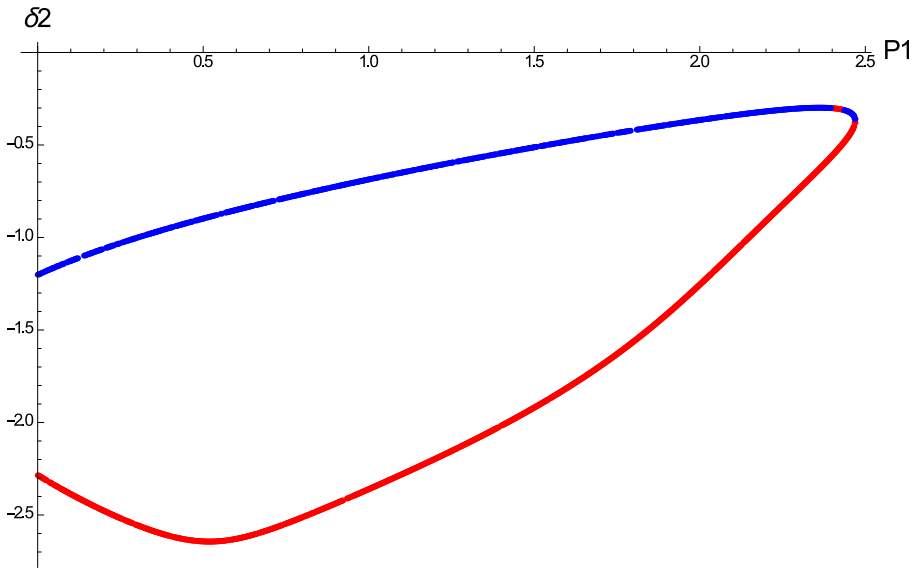
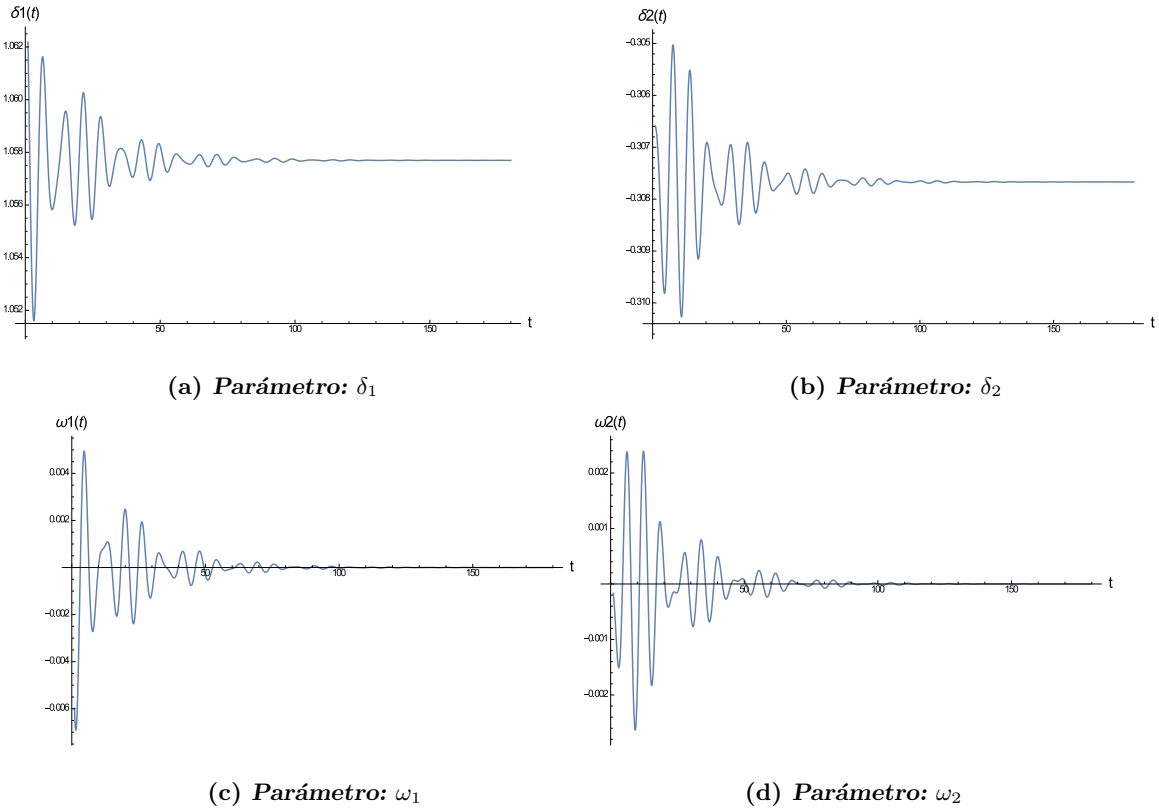


Figura 4.18: Diagrama de bifurcación del Sistema (4.5), para los parámetros δ_2, ω_2.

estable en el diagrama de bifurcación. La dinámica que presentan las variables para este punto es mostrado en la Figura 4.19.



(a) *Parámetro:* δ_1 **(b) *Parámetro:*** δ_2

(c) *Parámetro:* ω_1 **(d) *Parámetro:*** ω_2

Figura 4.19: Comportamiento de las variables del Sistema (4.5)

Las Figuras 4.20 y 4.22 muestran el diagrama de fase y el campo vectorial para este punto. Se observa que la línea de flujo tiende a un estado de equilibrio mientras que $t \to \infty$, confirmando mediante esto la estabilidad del punto fijo. Debido al teorema de no intersección el cual nos dice que: una trayectoria que pasa a través de al menos un punto que no es un punto crítico, no puede cruzarse así mismo si no es una curva cerrada. En este caso la trayectoria corresponde a una solución periódica del sistema [Boyce01]. Por lo tanto se podría pensar que este diagrama de fase está equivocado, debido a que la trayectoria se intersecta. Esto pasa porque la trayectoria se está proyectada en un espacio 2D, si se realiza un gráfico en un espacio n-dimensional ($n > 2$) estas intersecciones no ocurren tal como se muestra en la Figura 4.21.

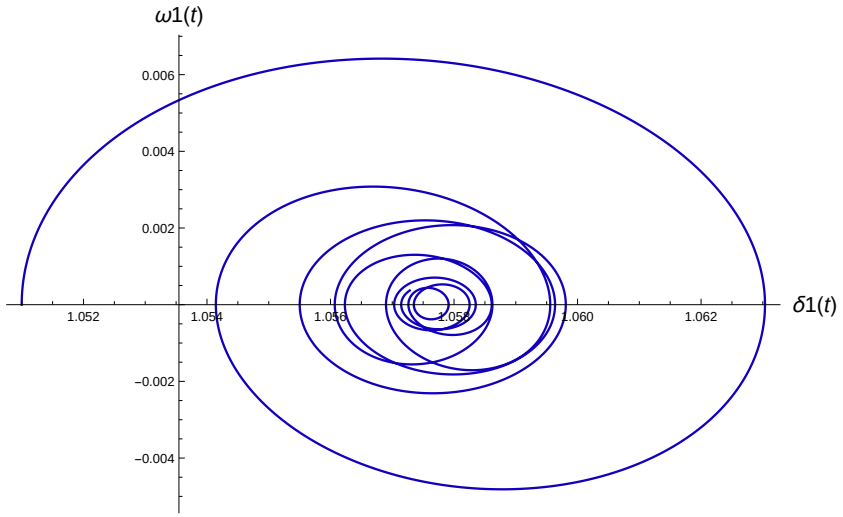


Figura 4.20: Diagrama de fase para los parámetros: ω_1, δ_1

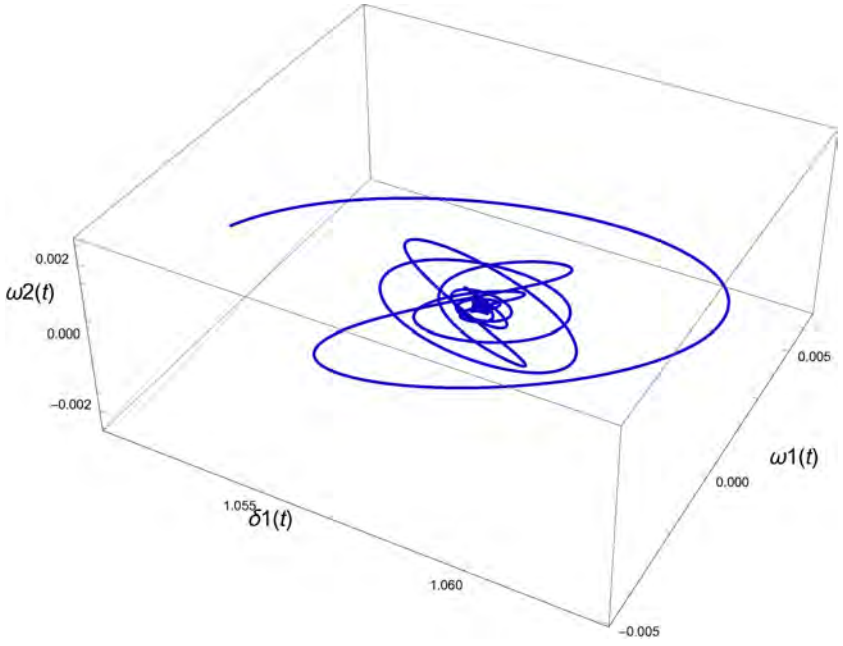


Figura 4.21: Diagrama de fase para los parámetros: ω_1, ω_2 y δ_1

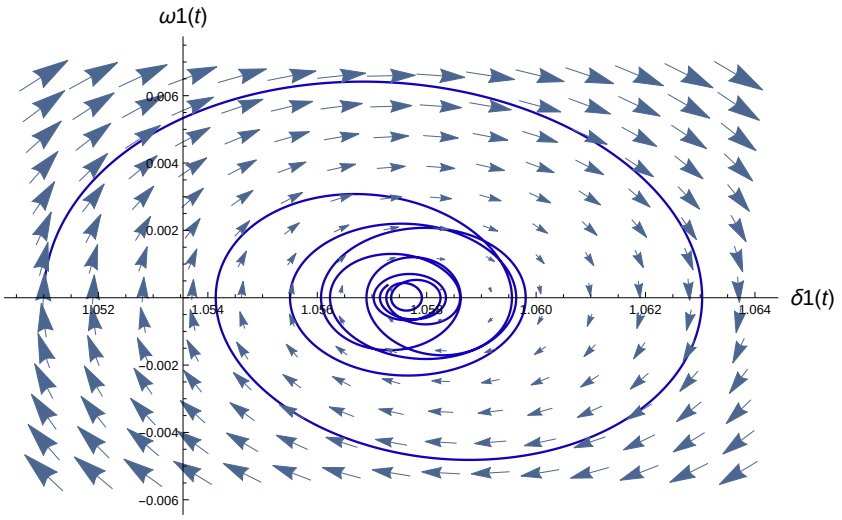


Figura 4.22: Campo Vectorial que se presenta en el punto: $P_1 = 2.26pu$, $\delta_1[t] = 1.06551$, $\delta_2[t] = -0.30654$.

Para obtener el mapa de Poicaré se define una recta que se encuentra situada en $\delta_1 = 1.056$, la cual es transversal a la línea de flujo del sistema. Las secciones de Poincaré para este diagrama de fase se muestran en la Figura 4.23.

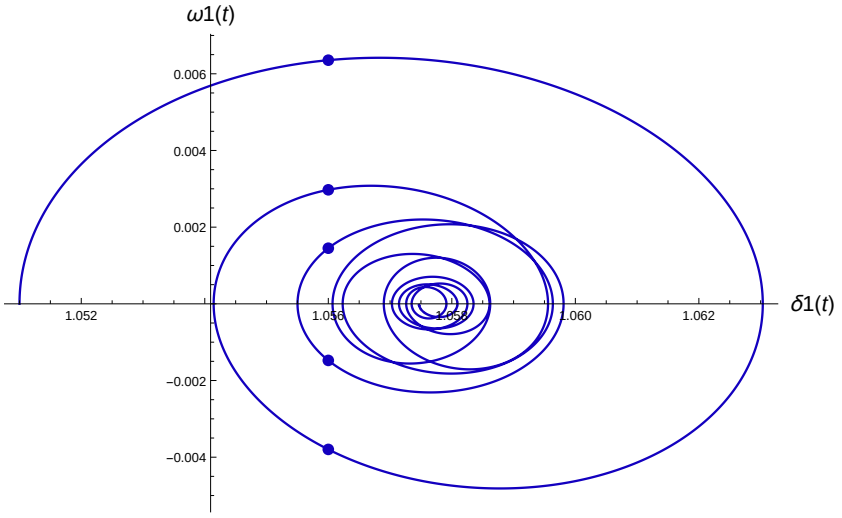


Figura 4.23: Secciones de Poicaré para la recta situada $\delta_1 = 1.056$.

Para este caso de estudio se ha tomado un puntos del diagrama de bifurcación, siendo éste estable. Esto se ha comprobando mediante la generación del diagrama de fase y campo vectorial en esta zona, con los cuales es posible observar la dirección que toma la solución. Se observa que esta llegara a un punto de equilibrio conforme transcurre el tiempo.

Al igual que los casos anteriores en esta tesis únicamente se realizo una reproducción de algunos gráficos presentados por Raúl García [Kasusky02], dando una breve explicación de estos. Si se desea observar una análisis mas detallado de este sistema. Éste se encuentra en el trabajo de investigación de Raúl García [Kasusky02].

4.4. Conclusiones del capítulo

En el presente capítulo se realizó un análisis a tres sistemas eléctricos, mediante los cuales se demostró que DynamicaM es una herramienta funcional, capaz de generar diagramas de bifurcación, secciones de Poincaré, campos vectoriales, diagramas de fase y simulaciones en el dominio del tiempo. Mediante la reproducción de estos gráficos se puede decir que DynamicaM es de gran utilidad para realizar análisis a sistemas dinámicos conformados por ecuaciones diferenciales ordinarias.

Capítulo 5

Conclusiones y trabajos futuros

En el presente capítulo se presentan las conclusiones generales del trabajo de investigación, así mismo los trabajos futuros que se derivan del mismo.

5.1. Conclusiones generales

Se ha logrado construir un sistema de software para el análisis de sistemas dinámicos, llamado DynamicaM, que permite realizar diagramas de fase, secciones de Poincaré, campos vectoriales, diagramas de bifurcación con puntos fijos reales y complejos. Para este propósito DynamicaM cuenta con una API y una GUI, se facilita mediante éstas el uso del mismo. Se comprobó la correctitud del software mediante la generación de una serie de diagramas para distintos sistemas de ecuaciones no lineales, se realizan comparaciones de los resultados obtenidos con resultados de tesis y artículos en los cuales son tratados los casos de estudio.

DynamicaM tiene algunas ventajas sobre los trabajos de investigación mencionados en los antecedentes. Por ejemplo, en Julio Barrera [Barrera07] , Mauricio López [Villanueva10] y Rodrigo López [López10] únicamente abordan la generación de DB mediante métodos heurísticos. En estos trabajos no se da acceso a alguna librería que permita el uso de estas técnicas. Mientras que para este propósito DynamicaM ofrece una interfaz de programación de aplicaciones y una interfaz gráfica de usuario. En las cuales además de generar

DB se cuentan con los métodos de generación de secciones de Poicaré, campos vectorial, diagramas de fase y simulaciones en el dominio del tiempo. Por otra parte el software de XPP/XPPAUT [Doedel94] tiene la capacidad de generar DB, la gran desventaja con la que cuenta este software en la construcción de DB, es la técnica que utiliza, en la cual es necesario contar con un punto fijo inicial que en muchas ocasiones no es trivial calcularlo, esto para aplicar un método de continuación para construir el DB. Además para la describir un sistema dinámico es imprescindible tener conocimientos previos de Fortran/C y poseer conocimientos previos de la nomenclatura usada. Por lo tanto DynamicaM tiene como ventajas sobre este software, la generación de DB de forma automática, sin necesidad de tener un punto fijo inicial, la descripción simbólica de un sistema dinámico (i.e. tal y como se describen el los libros de texto), etc.

Mencionado lo anterior se concluye que el objetivo de investigación de esta tesis ha sido cumplido, al lograr construir sistema de un software que cumpliera con los objetivos propuestos, proporcionando al usuario una herramienta útil que cumple con los requerimientos indispensables para realizar un análisis a un sistema de ecuaciones diferenciales. Así mismo DynamicaM constituye una herramienta que auxilia a los alumnos en el análisis de sistemas, favoreciendo y estimulando el uso y aprendizaje de la misma.

5.2. Trabajos futuros

- Realizar gráficos interactivos, en los cuales sea posible obtener el valor de cualquier punto fijo que se encuentre contenido en el diagrama de bifurcación.
- Implementar animaciones de las dinámicas que puede presentar un parámetro del sistema al presentarse una perturbación, al igual que el comportamiento que exhibe un sistema al variar un parámetro de bifurcación.
- Codificar diversos métodos heurísticos para la generación de diagramas de bifurcación, como son: evolución diferencial, algoritmos genéticos, recocido simulado, etc.
- Codificar diferentes métodos para la generación de diagramas de fase, en los cuales

no sea necesario incluir condiciones iniciales del sistema.

- Implementar métodos de visualización de alta dimensionalidad, en los cuales sea posible visualizar mas de tres dimensiones.

Apéndice A

Apéndice A: Funciones API

En este apéndice se presenta el código implementado en el entorno de Mathematica, para realizar un análisis a un sistema dinámico. Este código conforma la interfaz de programación de aplicaciones.

Función *DefineConstan*:

```
Clear[DefineConstant];
DefineConstant[options__]:=
  Block[{constant,Constant2},
    {Constant2}={Constant}/.{options}/.Options[DefineConstant];
    Constant2
  ];

Options[DefineConstant]={
  Constant-> Null
};
```

Función *DefineAlgebraicEquation*:

```
Clear[DefineAlgebraicEquation];
DefineAlgebraicEquation[options__]:=
  Block[{AlgebraicEquation},
    AlgebraicEquation={AE}/.{options}/.Options[
```

```
        DefineAlgebraicEquation];
    AlgebraicEquation
  ];

Options[DefineAlgebraicEquation]={
  AE-> Null
};
```

Función *DefineODE*:

```
Clear[DefineODE];
DefineODE[options__]:=
  Block[{ode},
    ode=ODE/.options/.Options[DefineODE];
    ode
  ];

Options[DefineODE]={
  ODE-> Null
};
```

Función *DefineSystem*:

```
Clear[DefineSystem];
DefineSystem[options__]:=
Block[{Odes,SV,BV,Sy,AE,C},
 {AE,Odes,C,SV,BV}={AlgebraicEquation,OrdinaryDifferentialEquations,
       Constant,StateVariables,BifurcationVariables}/.{options}/.
       Options[DefineSystem];
  Odes=Odes/.AE;
  Odes=Odes/.C;
  Sy=System[
    AlgebraicEquation-> AE,
    OrdinaryDifferentialEquations-> Odes,
    Constant-> C,
```

```
13      StateVariables-> SV,
14      BifurcationVariables-> BV];
15    Return[Sy];
16   ];
17
18 Options[DefineSystem]={
19    AlgebraicEquation->{Null->Null},
20    OrdinaryDifferentialEquations-> {Null->Null},
21    Constant->{Null-> Null},
22    StateVariables->Null,
23    BifurcationVariables-> Null
24    };
```

Función *DefineExperiment*:

```
1  Clear[DefineExperiment];
2  DefineExperiment[options__]:=
3    Block[{St,BR,D,FN,MH,Args,Experiment,Expe},
4      {St,BR,D,FN,MH,Args}={SystemODE,BifRanges,Dir,FName,
           MetaHeuristic,Arguments}/.{options}/.Options[DefineExperiment
           ];
5      Expe=Experiment[SystemODE-> St,
6           BifRanges-> BR,
7           Dir->D,
8           FName-> FN,
9           MetaHeuristic-> MH,
10          Arguments-> Args];
11     CreateDirectory[Dir<>FName];
12 Return[Expe];
13 ]
14
15 Options[DefineExperiment]={
16    SystemODE-> Null,
17    BifRanges-> Null,
```

```
18    Dir-> NotebookDirectory[] <> "MyDiagram/",
19    FName-> "Niches-",
20    MetaHeuristic-> NichesPSO,
21    Arguments-> {NumberOfParticles->100,
22       NumberOfIterations->100,
23       RadiusSize->0.1,SizeNiche->20,C0->10,C1->2.1,
24       C2->2.1,TS->0.9}
25  }
```

Función *CreateBifurcationDiagram*:

```
1  Clear[CreateBifurcationDiagram]
2  CreateBifurcationDiagram[options__]:=
3  Module[{Exp,bd,owr,mf,fct,aptitude,varsBifList,varsBifParams,
       allVarsBif,res,archivo,f,file,j,FileNumber,ranges,rangesVarsBif,
       opts,susts,stVars,prVars,odes,pd,rute,nameFile,diagram,poinstSt,n
       ,maxIter,r,tamNiche,c1,c2,threshold,c,mapFct,aux,MH,Args,BR,SODE,
       FPComplex,FileName},
4    mapFct=g1;
5    Exp=Experiment/.{options}/.Options[CreateBifurcationDiagram];
6    FPComplex= ComplexFixedPoints/.{options}/.Options[
         CreateBifurcationDiagram];
7    FileName=FName/.Exp[[4]];
8    MH=MetaHeuristic/.Exp[[-2]];
9    Args=Arguments/.Exp[[-1]];
10   SODE= SystemODE/.Exp[[1]];
11   BR= BifRanges/.Exp[[2]];
12   bd=BifurcationDiagram[System[SODE[[2]],SODE[[4]],SODE[[5]]],
         BifRanges-> BR,Exp[[3]],FName-> FileName];
13   pd=bd[[1]];
14   odes=getOrdinaryDifferentialEquations[pd];
15   stVars = getStateVariables[pd];
16   odes=FilterODE[odes,stVars];
17   rangesVarsBif=getBifurcationRanges[bd];
```

```
18  prVars= getBifurcationVariables[pd];
19  allVarsBif=generateVariations[rangesVarsBif];
20  aptitude=(odes/.susts[stVars,#1]~Join~susts[prVars,#2])&;
21  mf=mapFct;(*mapFunction/.{mapFct};*)
22  fct=(mf[aptitude[#1,#2]])&;
23  opts=EvaluationFunction->fct;
24  res=Table[{},{2}];
25
26  susts=(MapThread[(#1->#2)&,{#1,#2}])&;
27  FileNumber=1;
28  nameFile=getFName[bd];
29  rute=getDirectory[bd];
30  If[existDirectory[bd]=="true",
31    If[FPComplex== True,
32      ranges=getSearchRangesC[pd],
33      ranges=getSearchRanges[pd]
34    ];
35    If[MH== DifferentialEvolution,Print["DifferentialEvolution"]];
36    If[MH== PSO, Print["PSO"]];
37    If[MH== PG,  Print["PG"]  ];
38    If[MH== NichesPSO,
39      {n,maxIter,r,tamNiche,c,c1,c2,threshold}={NumberOfParticles,
          NumberOfIterations,RadiusSize,SizeNiche,C0,C1,C2,TS}/.Args;
40      For[j=1,j<=Length[allVarsBif],j++,
41        Print["Searching fixed points for:", allVarsBif[[j]]];
42        Print["*****************",j,  "of" ,Length[allVarsBif],"
            *****************"];
43        If[FPComplex== True,
44          res=nichesPSOC[n,maxIter,r,ranges,allVarsBif[[j]],opts,
              res,tamNiche,c1,c2,threshold,c,rute<>"convergencia"<>
              nameFile<>"-"<>ToString[FileNumber]];,
45          res=nichesPSO[n,maxIter,r,ranges,allVarsBif[[j]],opts,
              res,tamNiche,c1,c2,threshold,c,rute<>"convergencia"<>
```

```
                    nameFile<>"-"<>ToString[FileNumber]];
46              ];
47              file={allVarsBif[[j]]}~Join~{First[res][[2]]}~Join~{{Last[
                    res]}};
48              f=OpenWrite[rute<>nameFile<>ToString[FileNumber]];
49              Write[f,file];
50              FileNumber++;
51              Close[f];
52            ];
53      ];
54      If[FPComplex==True,
55        diagram = FilterFixedPoints[bd, MapFunction -> mapFct,True];,
56        diagram = FilterFixedPoints[bd, MapFunction -> mapFct,False];
57      ];
58      f = OpenWrite[rute<>"FilterFixedPoints"];
59      Write[f,{{prVars,stVars},diagram}];
60      Close[f];
61      Print["Determine conjugates"];
62      diagram=Conjugates[diagram];
63      poinstSt = DeterminateStability[diagram, Exp];
64      Print["Process completed"];
65      f = OpenWrite[rute<>"BifurcationDiagram"];
66      (*Write[f,<|Exp[[1, 2, 1]],Exp[[2]],Exp[[1,2,2]],BifDiagram->{{
            prVars,stVars},poinstSt},PlaneComplex-> FPComplex|>];(*quite
            llaves*)*)
67      Write[f,{{prVars,stVars},poinstSt}];(*quite llaves*)
68      Close[f];
69      Return[{{prVars,stVars},poinstSt}];
70    ];
71  ]
72
73  Options[CreateBifurcationDiagram]={
74    Experiment-> Null,
```

```
75    ComplexFixedPoints-> False
76   }
```

Función *PlotBD*:

```
1  Clear[PlotBD];
2  PlotBD[BD_,SelectVars_]:=
3   Block[{p1,p2,p,Dim,plt0={},plt1={},V,selectvars, f},
4    Dim=Length[SelectVars];
5    selectvars = Filter[SelectVars];
6    If[Length[BD[[2,1]]]>1,
7      plt0 = SelectPoints[BD[[2, 1]], selectvars, BD[[1]]];
8      plt0=Table[SelectVars[[i]] /. selectvars[[i]] -> #[[i]], {i,
           Length[SelectVars]}] & /@ plt0;
9      ];
10   If[Length[BD[[2,2]]]>1,
11     plt1 = SelectPoints[BD[[2, 2]], selectvars, BD[[1]]];
12     plt1=Table[SelectVars[[i]] /. selectvars[[i]] -> #[[i]], {i,
           Length[SelectVars]}] & /@ plt1;
13     ];
14   If[Length[plt0]>0,
15     If[Length[plt1]>0,
16       If[Dim<3,
17         ListPlot[{plt1,plt0},PlotStyle-> {Red,Blue,PointSize[10]},
               PlotLegends->{"unstable points","stable points"},
               AxesLabel -> {Style[SelectVars[[1]],15],Style[SelectVars
               [[2]],15]},ImageSize->Large],
18         If[Dim== 3,
19         ListPointPlot3D[{plt0,plt1},PlotStyle-> {Blue,Red,PointSize[
               Large]},PlotLegends->{"stable points","unstable points"},
               AxesLabel -> {Style[SelectVars[[1]],15],Style[SelectVars
               [[2]],15],Style[SelectVars[[3]],15]},ImageSize->Large]
```

```
20            ]
21        ],
22          If[Dim<3,
23            ListPlot[{plt0},PlotStyle-> {Blue,PointSize[10]},
                PlotLegends->{"unstable points","stable points"},
                AxesLabel -> {Style[SelectVars[[1]],15],Style[
                SelectVars[[2]],15]},ImageSize->Large],
24            If[Dim== 3,
25            ListPointPlot3D[{plt0},PlotStyle-> {Blue,PointSize[Large
                ]},PlotLegends->{"stable points","unstable points"},
                AxesLabel -> {Style[SelectVars[[1]],15],Style[
                SelectVars[[2]],15],Style[SelectVars[[3]],15]},
                ImageSize->Large]
26            ]
27          ]
28        ],
29          If[Length[plt1]>0,
30            If[Dim<3,
31              ListPlot[{plt1},PlotStyle-> {Red,PointSize[10]},
                  PlotLegends->{"unstable points","stable points"},
                  AxesLabel -> {Style[SelectVars[[1]],15],Style[
                  SelectVars[[2]],15]},ImageSize->Large],
32              If[Dim== 3,
33              ListPointPlot3D[{plt1},PlotStyle-> {Red,PointSize[Large
                  ]},PlotLegends->{"stable points","unstable points"},
                  AxesLabel -> {Style[SelectVars[[1]],15],Style[
                  SelectVars[[2]],15],Style[SelectVars[[3]],15]},
                  ImageSize->Large]
34              ]
35            ]
36          ]
37    ]
38  ]
```

Función *PhaseDiagram*:

```
Clear[PhaseDiagram1];
PhaseDiagram1[options__] :=
 Block[{Sol, F, IC, pv2, S1, S2, Data, DataMP, x1, x2, y1, y2, z1,
    z2,t, Vars,Plot1, Plot2, pv, Data1, x, y, z, system, system2,
    ODER, syst, PV,  IC1, BP, T, VF, PS, VFS, VFC, VS = Automatic,
    VP = Automatic, VFER,PSS, PSP, PSD},
  {syst, PV, IC1, BP, T, VF, PS} ={SystemODE, PlotVaribles,
     InitialCondition, BifurcationParameter,EvalutionRanges,
     VectorField, PoincareSection} /. {options};
  {VFS, VFC, VFER, VS,VP} = {ShowVF, Constants, EvalutionRanges,
     VectorScale,VectorPoints} /. VF(*/.Options[VectorField]*);
  {PSS, PSP, PSD} = {ShowPS, Plane, Displacement} /. PS(*/.Options[
     PoincareSection]*);
  system = GetSystem[syst] /. BP;
  system2 = system;
  pv = IC1[[1]] /. x_[t_] -> x;
  IC = Table[MapThread[(#1 == #2) &, {IC1[[1]], IC1[[i]]}] /. x_[t_]
      -> x[0], {i, 2, Length[IC1]}];
  Sol = NDSolve[Join[system2, #], IC1[[1]], T] & /@ IC;
  F = IC1[[1]] /. # & /@ Sol;
  Data = { {IC1[[1]]},Table[{Flatten[#]} /. T[[1]] -> i, {i, T[[2]],
     T[[3]], 0.01}] & /@ F};
  Data = SelectPoints[#, PV, Data[[1]]] & /@ Data[[2]];
  If[VFS == True,
    ODER = Table[VFER[[i, 1]], {i, Length[VFER]}];
    pv2 = ODER /. x_[t_] -> x'[t];
    system = Flatten[Table[Select[system, #1[[1]] == pv2[[i]] &], {i
       , Length[pv2]}]];
    system = system /. VFC;
    system = FilterODE[system, pv];
    pv2 = ODER /. x_[t_] -> x;
    If[Length[system] == 1, Vars = {x};system = system /. MapThread
```

```
    [(#1 -> #2) &, {pv2, Vars}];];
23  If[Length[system] == 2, Vars = {x, y};system = system /.
    MapThread[(#1 -> #2) &, {pv2, Vars}];];
24  If[Length[system] == 3, Vars = {x, y, z};system = system /.
    MapThread[(#1 -> #2) &, {pv2, Vars}];];
25  S2 = system;
26  x1 = VFER[[1, 2]]; x2 = VFER[[1, 3]]; y1 = VFER[[2, 2]];
27  y2 = VFER[[2, 3]];
28  If[Length[PV] == 2,
29    Plot1 = Show[Table[ListLinePlot[Data[[i]], PlotStyle -> {
        Colors[[i]]},ImageSize -> Large,PlotRange->All], {i, Length
        [Data]}],VectorPlot[S2, {x, x1, x2}, {y, y1, y2}, ImageSize
         -> Large,VectorScale -> VS, VectorPoints -> VP],AxesLabel
        -> {Style[PV[[1]], 15], Style[PV[[2]], 15]},PlotRange->All,
        ImageSize -> Large];,(*Else*)
30    If[Length[PV] == 3,
31      z1 = VFER[[3, 2]]; z2 = VFER[[3, 3]];
32      Plot1 = Show[VectorPlot3D[S2, {x, x1, x2}, {y, y1, y2}, {z,
          z1, z2}, ImageSize -> Large,VectorScale -> VS,
          VectorPoints -> VP], Table[Graphics3D[{Colors[[i]],
          Thickness[0.004],Line[Data[[i]]]}, Axes -> True,
          ImageSize -> Large], {i,Length[Data]}], ImageSize ->
          Large,AxesLabel -> {Style[PV[[1]], 15], Style[PV[[2]],
          15],Style[PV[[3]], 15]}, PlotRange -> {{x1, x2}, {y1, y2
          }, {z1, z2}}];];
33  ];,(*Else*)
34    If[VFS == False,
35      If[Length[PV] == 2,
36        Plot1 =Show[Table[ListLinePlot[Data[[i]], PlotStyle -> {
            Colors[[i]]},ImageSize -> Large,PlotRange->All], {i,
            Length[Data]}],AxesLabel -> {Style[PV[[1]], 15], Style[
            PV[[2]], 15]}, ImageSize -> Large, PlotRange -> All
            ];,(*Else*)
```

```
37            If[Length[PV] ==  3,
38              Plot1 = Show[Table[Graphics3D[{Colors[[i]], Thickness
                    [0.004],Line[Data[[i]]]}, Axes -> True, ImageSize ->
                    Large], {i, Length[Data]}],ImageSize -> Large ,
                    PlotRange -> All, AxesLabel -> {Style[PV[[1]], 15],
                    Style[PV[[2]], 15],Style[PV[[3]], 15]}];
39             ];
40            ];
41           ];
42   ];
43   If[PSS == True,
44    DataMP = PoincareSection2[system2, Take[IC1, {2, -1}], IC1[[1]],
        PSP, PV,T];
45     If[Length[PV] == 2,
46       Plot2 = Table[ListPlot[DataMP[[i]],PlotStyle -> {Colors[[i]],
           PointSize[0.015]},ImageSize -> Large ], {i, Length[DataMP
           ]}];
47       Show[Plot1, Plot2, ImageSize -> Large],(*Else*)
48       If[Length[PV] == 3,
49         DataMP = Table[#[[i]] + {PSD[[1]], PSD[[2]], PSD[[3]]}, {i,
              Length[#]}] & /@ DataMP;
50         Plot2 =Show[Table[ListPointPlot3D[DataMP[[i]], PlotStyle ->
              {PointSize[0.015], Colors[[i]]},ImageSize -> Large], {i,
              Length[DataMP]}],ImageSize -> Large];
51         Show[Plot1, Plot2, ImageSize -> Large]
52        ]
53     ],
54     If[PSS == False,
55       Show[Plot1, ImageSize -> Large]
56     ]
57    ]
58   ]
59
```

```
60 Unprotect[PhaseDiagram];
61 Options[PhaseDiagram] =
62  {
63   SystemODE -> Null,
64   PlotVaribles -> Null,
65   InitialCondition -> Null,
66   BifurcationParameter -> Null,
67   EvalutionRanges -> {t, 0, 100}
68   }
69 Unprotect[VectorField];
70 Options[VectorField] = {
71   ShowVF -> False,
72   Constant -> Null,
73   EvalutionRanges -> Null,
74   VectorScale -> Automatic,
75   VectorPoints -> Automatic
76   }
77 Unprotect[PoincareSection];
78 Options[PoincareSection] = {
79   ShowPS -> False,
80   Plane -> Null ,
81   Displacement -> Null
82   }
```

Apéndice B

Apéndice B: Funciones GUI

En este apéndice se presenta el código que conforma la estructura de la interfaz gráfica de aplicaciones, el cual se encuentra separado por cinco secciones principales : GUI, CreateBifurcationDiagramGUI, LoadBiburcationDiagramGUI, DynamicalSystemsAnalysisGUI y TimeSimulationGUI.

Interfaz *GUI*:

```
GUI[] := Module[{},
   CreateDialog[
    TabView[{
      "Bifurcation Diagram" -> TabView[{"Create Bifurcation Diagram"
          -> CreateBifurcationDiagramGUI[],"Load Bifurcation Diagram
         " -> LoadBifurcationDiagramGUI[]}],
      "Dynamical Systems Analysis" -> DynamicalSystemsAnalysisGUI[],
      "Simulation" -> TimeSimulationGUI[]
      }
     ], WindowTitle -> "DynamicM",
    NotebookEventActions -> {"WindowClose" :> {}}]
   ];
```

Interfaz $CreateBifurcationDiagramGUI$:

```
CreateBifurcationDiagramGUI[] :=
 Manipulate[
  If[CalculateBD == True,
   If[BDC == True,
    EV = Table[{EV[[i, 1]], Take[EV[[i]], {2, -1}],Take[EVI[[i]],
        {2, -1}]}, {i, Length[EV]}];
    ];
   BD = CreateBifurcationDiagram2[Experiment -> DefineExperiment[
       SystemODE -> Flatten[DefineSystem[AlgebraicEquation -> ALEQ,
       OrdinaryDifferentialEquations -> ODE, Constant -> CONST,
       StateVariables -> EV, BifurcationVariables -> Table[BP[[i,
       1]], {i, Length[BP]}]]], BifRanges -> BP, Arguments -> {
       NumberOfParticles -> NumPart, NumberOfIterations -> NumIter,
       RadiusSize -> RadSize, SizeNiche -> SizeNi, C0 -> C00, C1 ->
       C11, C2 -> C22, TS -> TSS}], ComplexFixedPoints -> BDC];
   CalculateBD = False;
   ];
  If[ShowBD1 == True,
    ShowBD1 = False;
    If[DBP == True,
      DBP = False;
      PltE =IdentificationSaddleNode[BD[[1]], BD[[2, 1]],FilterODE[
          ODE, Table[BP[[i, 1]], {i, Length[BP]}]], Green,"*"];
      PltU = IdentificationSaddleNode[BD[[1]], BD[[2, 2]], FilterODE
          [ODE, Table[BP[[i, 1]], {i, Length[BP]}]], Green,"*"];
    ];
    If[Length[BD] > 0 ,
      ExportImage = Show[PlotBD[BD, Vars], PltE, PltU];,
      Print["Calculate BD or upload a file"];
    ]
   ];
  If[Length[ExportImage] == 1,
```

```
23      ExportImage[[1]],
24      ExportImage
25     ],
26    {{PltE, {}}, ControlType -> None},
27    {{PltU, {}}, ControlType -> None},
28    {{CalculateBD, False}, ControlType -> None},
29    {{ODE, {x'[t] == r + x[t]^2}}, ControlType -> None},
30    {{ALEQ, {Null -> Null}}, ControlType -> None},
31    {{CONST, {Null -> Null}}, ControlType -> None},
32    {{BP, {{r, -3, 3, 0.02}}}, ControlType -> None},
33    {{EVI, {{x, -3, 3}}}, ControlType -> None},
34    {{DBP, False}, ControlType -> None},
35    {{EV, {{x, -3, 3}}}, ControlType -> None},
36    {{Vars, {r, x}}, ControlType -> None},
37    {{ShowBD1, False}, ControlType -> None},
38    {{BDC, False}, ControlType -> None},
39    {{ExportImage,Import[NotebookDirectory[] <> "/ExampleBifurcation.
          pdf"]},ControlType -> None},
40    {{Dir, NotebookDirectory[]}, ControlType -> None},
41    {{NameExport, "Example"}, ControlType -> None},
42    {{BD, False}, ControlType -> None},
43    {{NumPart, 100}, ControlType -> None},
44    {{NumIter, 100}, ControlType -> None},
45    {{RadSize, 0.1}, ControlType -> None},
46    {{SizeNi, 20}, ControlType -> None},
47    {{C00, 10}, ControlType -> None},
48    {{C11, 2.1}, ControlType -> None},
49    {{C22, 2.1}, ControlType -> None},
50    {{TSS, 0.9}, ControlType -> None},
51    {{AAE, False}, ControlType -> None},
52    {{AC, False}, ControlType -> None},
53    Pane[
54     Grid[{{" "}, {     Style["Bifurcation diagram:", 13] }, {" "},
```

```
55      {Control[{{ODE, {x'[t] == r + x[t]^2},"
                                    System:           "}}]},
56      {Control[{{BP, {{r, -3, 3, 0.02}},"   Bifurcation parameters:
                  "}} ]},
57      {Control[ {{EV, {{x, -4, 4}},"        State variables (SV):
                  "}}]},
58      {Grid[{{"Add algebraic equations: ", Checkbox[Dynamic[AAE]],
           PaneSelector[{True ->
59            Control[{{ALEQ, {Null -> Null}, " "}}]},
60         Dynamic[AAE]]} }]},
61
62      {Grid[{{"                  Add constants: ", Checkbox[Dynamic[AC
           ]],
63          PaneSelector[{True ->
64            Control[{{CONST, {Null -> Null}, "    "}}]  },
65         Dynamic[AC]]} }]},
66      {Grid[{{" "}, {"Complex Fixed Points:" Checkbox[Dynamic[BDC]],
67          " "}, {PaneSelector[{True ->
68            Control[{{EVI, {{x, -4, 4}},
69               "     Ranges Imaginary SV:             "}}]},
70         Dynamic[BDC]], " "}}]},
71      {Grid[{{Button["Arguments PSONiches",
72          CreateDialog[{TextCell[" Enter arguments of PSONiches: "],
73
74            Grid[{{"      Number of particles: ",
75               InputField[ Dynamic[NumPart], Number]},
76              {"     Number of iterations: ",
77               InputField[Dynamic[NumIter], Number]},
78              {"                   Radius size: ",
79               InputField[  Dynamic[RadSize], Number]},
80              {"                    Size niche: ",
81               InputField[  Dynamic[SizeNi], Number]},
82              {"                                C0: ",
```

```
83                InputField[  Dynamic[C00], Number]},
84               {"                                C1: ",
85                InputField[ Dynamic[C11], Number]},
86               {"                                C2: ",
87                InputField[ Dynamic[C22], Number]},
88               {"                                TS: ",
89                InputField[ Dynamic[TSS], Number]} }],
90
91             Grid[{{"" , "", ""}, {"                              ",
92                Button[" Accept ", DialogReturn[]]}}]},
93            WindowTitle -> "Arguments PSONiches"] ],
94          Button["Run experiment", {CalculateBD = True}]}}]},
95      {" "},
96      {" "},
97      {    Style["Plot bifurcation diagram:", 13] },
98      {Grid[{{"", ""},
99         {Control[{{Vars, {r, x}, "Plot variables:"}}]},
100        {Grid[{{" ", " " }, {"Determine bifurcation points:",
101            Checkbox[Dynamic[DBP]]}}]},
102        {Grid[{{" ", " ", ""}, { " ",
103            Button["Plot BD", ShowBD1 = True],
104            Button["Save Plot",
105
106             CreateDialog[{TextCell["Enter a file name: "],
107               InputField[Dynamic[NameExport], String,
108                FieldHint -> "Example"],
109               Grid[{{"" , ""}, {Button["Save",
110
111                   If[Length[BD] > 0,
112                    Export[Dir <> NameExport <> ".pdf", ExportImage
                         ];
113                    Print["Saved bifurcation diagram."],
114                    Print["Error: The bifurcation diagram dont exist
```

```
                        "]]],
115                   Button["Close", DialogReturn[],
116                    Appearance -> {"DialogBox"}] }}]},
117               WindowTitle -> "Save image"] ]}}] }, {"", ""}
118         }
119        ]
120       }, {Grid[{{"", "",
121          ""}, {"

              ", Button["  Close  ", DialogReturn[], Appearance -> {"
              DialogBox"}],
122          Button["  Abort evaluation ", {Print["#Aborted"],
123            Interrupt[]}, Appearance -> {"DialogBox"}]}
124         }]}
125        },
126     Frame -> {None,
127       None, {{{3, 11}, {1, 1}} -> True, {{13, 14}, {1, 1}} -> True
            }}],
128     ImageSize -> {395, 450}, Scrollbars -> {False, True}]
129   , ControlPlacement -> Left
130 ]
```

Interfaz *LoadBiburcationDiagramGUI*:

```
1  LoadBifurcationDiagramGUI[] := Manipulate[
2    If[ShowBD == True,
3     ShowBD = False;
4     If[Length[BD[BifDiagram]] > 0,
5      If[DBP == True,
6       DBP = False;
7       BD1 = BD[BifDiagram];
8       PltE =
9        IdentificationSaddleNode[BD1[[1]], BD1[[2, 1]],
10        BD[OrdinaryDifferentialEquations], Blue];
```

```
11      PltU =
12       IdentificationSaddleNode[BD1[[1]], BD1[[2, 2]],
13        BD[OrdinaryDifferentialEquations], Red];];
14     ExportImage = Show[PlotBD[BD[BifDiagram], Vars], PltE, PltU];,
15     Print["Calculate BD or upload a file"];
16     ]
17    ]; If[Length[ExportImage] == 1, ExportImage[[1]], ExportImage],
18     {{PltE, {}}, ControlType -> None},
19   {{PltU, {}}, ControlType -> None},
20   {{UploadFile, False}, ControlType -> None},
21   {{ShowBD, False}, ControlType -> None},
22   {{DBP, False}, ControlType -> None},
23   {{Vars, {\[Delta]1, P1}}, ControlType -> None},
24   {{ExportImage,
25     Import[NotebookDirectory[] <> "/ExampleBifurcation.pdf"]},
26    ControlType -> None}, {{Dir, NotebookDirectory[]},
27    ControlType -> None},
28   {{NameExport, "Example"},
29    ControlType -> None}, {{Directory1,
30     NotebookDirectory[] <> "ExampleGUI/" <> "BifurcationDiagram"},
31    ControlType -> None}, {{BD,
32     LoadBifurcationDiagram[
33      NotebookDirectory[] <> "ExampleGUI/" <> "BifurcationDiagram"]},
34    ControlType -> None},
35   {{BD1, LoadBifurcationDiagram[
36      NotebookDirectory[] <> "ExampleGUI/" <> "BifurcationDiagram"]},
37    ControlType -> None},
38   Pane[Grid[{{" "}, {Style["Load Experiment: ",
39        13]}, {Grid[{{" ",
40          " "}, {Control[{{Directory1, Directory1,
41             "File Directory: "}}],
42          FileNameSetter[
43           Dynamic[Directory1]]}}]}, {Grid[{{" "}, {Button["Load File
```

```
                ",
44              BD = LoadBifurcationDiagram[
45               Directory1]]}}]}, {Column[{"  "}]}, {Style[
46          "Plot Bifurcation Diagram:",
47          13]}, {Grid[{{" "}, {Grid[{{"",
48               ""}, {"  System:                              ",
49               InputField[Dynamic[BD[OrdinaryDifferentialEquations]],
50                Enabled -> False]}}]}, {Grid[{{"",
51               ""}, {"Bifurcation Parameters: ",
52               InputField[
53                Dynamic[
54                 Table[BD[BifRanges][[i, 1]], {i,
55                   Length[BD[BifRanges]]}]],
56                Enabled -> False]}}]}, {Grid[{{"",
57               ""}, {"State Variables:                ",
58               InputField[
59                Dynamic[
60                 Table[BD[StateVariables][[i, 1]], {i,
61                   Length[BD[StateVariables]]}]],
62                Enabled -> False]}}]}, {Grid[{{"",""}, {"Complex:
                                        ",
63               InputField[Dynamic[BD[PlaneComplex]],
64                Enabled ->
65                 False]}}]}}]}, {Grid[{{" "}, {Control[{{Vars, {\[Delta
                     ]1, P1}, "Plot Variables:"}}]}, {Grid[{{" ",
66               " "}, {"Determinate Bifurcation Points:",
67               Checkbox[Dynamic[DBP]]}}]}, {Grid[{{" ", " ", ""}, {" ",
68               Button["Show BD", ShowBD = True],
69               Button["Save Plot",
70                CreateDialog[{TextCell["Enter a file name: "],
71                  InputField[Dynamic[NameExport], String,
72                   FieldHint -> "Example"],
73                  Grid[{{"",
```

```
                    ""}, {Button["Save",
                     If[Length[BD] > 0,
                      Export[Dir <> NameExport <> ".pdf", ExportImage
                          ];
                      Print["Saved Bifurcation Diagram."],

                      Print["Error: The Bifurcation Diagram Dont exist
                          "]]],
                    Button["Close", DialogReturn[],
                     Appearance -> {"DialogBox"}]}]}],
                WindowTitle ->
                 "Save Image"]]}]}], {Column[{"  "}]}]}], {Grid[{{"",
          "", ""}, {"", ""  ,
          ""}, {"

              ", Button["  Close  ", DialogReturn[],
            Appearance -> {"DialogBox"}], " "}, {"", ""  , ""}
          }]}},
     Frame -> {None,
       None, {{{3, 4}, {1, 1}} -> True, {{6, 8}, {1, 1}} -> True}}],
    ImageSize -> {380, 450}, Scrollbars -> {False, True}]

  , ControlPlacement -> Left]
```

Interfaz *DynamicalSystemsAnalysisGUI*:

```
DynamicalSystemsAnalysisGUI[] := Manipulate[
  If[RunExperiment == True,
   If[ViewVF == True && ViewPS == True,
    RunExperiment = False;
    ExportImage =
     PhaseDiagram[System, IC[[1]],(*Table[IC[[i]],{i,2,Length[IC
         ]}]*)
      IC, T, VectorField -> {True, ER[[1]], ER[[2]]},
```

```
8         PoincareSection -> {True, Plane}];,
9       If[ViewVF == False && ViewPS == True,
10       RunExperiment = False;
11       ExportImage =
12        PhaseDiagram[System, IC[[1]],(*Table[IC[[i]],{i,2,Length[IC
              ]}]*)
13         IC, T, VectorField -> {False, ER[[1]], ER[[2]]},
14         PoincareSection -> {True, Plane}];
15       ,
16       If[ViewVF == True && ViewPS == False,
17        RunExperiment = False;
18        ExportImage =
19         PhaseDiagram[System, IC[[1]],(*Table[IC[[i]],{i,2,Length[IC
              ]}]*)
20          IC, T, VectorField -> {True, ER[[1]], ER[[2]]},
21          PoincareSection -> {False, Plane}];,
22        RunExperiment = False;
23        ExportImage =
24         PhaseDiagram[System, IC[[1]],(*Table[IC[[i]],{i,2,Length[IC
              ]}]*)
25          IC, T, VectorField -> {False, ER[[1]], ER[[2]]},
26          PoincareSection -> {False, Plane}];
27        ]
28      ]
29
30     ],
31    ExportImage
32    ],
33   {{T, {t, 1, 3}}, ControlType -> None},
34   {{IC, {{\[Omega][t], z[t]}, {1, -2.5}, {-1.5, 3}, {1, 2}, {-1,
35       5}, {2, 3}, {-5, -2.5}, {-9, -3}, {-5, -3}}},
36    ControlType -> None},
37   {{System, {\[Omega]'[t] == \[Omega][t] + E^-z[t], z'[t] == -z[t
```

```
    ]}},
38   ControlType -> None},
39  {{Plane, \[Omega][t] == 20}, ControlType -> None},
40  {{ViewPS, True}, ControlType -> None},
41  {{ViewVF, True}, ControlType -> None},
42  {{ER, {{\[Omega], -200, 200}, {z, -4, 4}}}, ControlType -> None},
43  {{ExportImage,
44    Import[NotebookDirectory[] <> "AnalisisSistemasGUI.pdf"]},
45   ControlType -> None},
46  {{RunExperiment, False}, ControlType -> None},
47  {{Dir, NotebookDirectory[]}, ControlType -> None},
48  {{NameExport, "Example"}, ControlType -> None},
49  Pane[Grid[{{""},
50     {Grid[{{" "}, {
51         Control[{{System, {\[Omega]'[t] == \[Omega][t] + E^-z[t],
52             z'[t] == -z[t]},
53            "System:                                    "}}]}, {
54         Control[{{IC, {{\[Omega][t], z[t]}, {1, -2.5}, {-1.5, 3},
               {1,
55                2}, {-1, 5}, {2, 3}, {-5, -2.5}, {-9, -3}, {-5, -3}},
56            "Initial Conditions:                "}}]}
57        }, Frame -> {None, None, {{{1, 3}, {1, 1}} -> True} }]},
58     {Grid[{{"Phase Diagram: "}, {
59         Grid[{{" ", " "}, {"Time:                                    ",
60            Control[{{T, {t, 1, 3}, ""}}]}}]
61         } }, Frame -> {None, None, {{{1, 2}, {1, 1}} -> True} } ]},
62     {Grid[{{"Vector Field: "}, {
63         Grid[{{" ", " "}, {  "Evaluation Ranges:          "     ,
64            Control[{{ER, {{\[Omega], -200, 200}, {z, -4, 4}},
65               ""}}] }}]},
66
67        {Grid[{{" ", " " }, {"Show:                                    ",
68            Checkbox[Dynamic[ViewVF]]}}]}
```

```
69
70          (*{Grid[{{" "," "} ,{ "Show:                                    "
                ,
71          RadioButtonBar[Dynamic[ViewVF],{True ,False}]}   }]}*)
72          }
73        , Frame -> {None, None, {{{1, 3}, {1, 1}} -> True} }]},
74       {Grid[{{"PoincarÃ© Sections: "}  , {
75           Grid[{{" ", " "}, {"Plane:                                  ",
76              Control[{{Plane, \[Omega][t] == 20, " "}}]   }}]},
77          {Grid[{{" ", " " }, {"Show:                                   ",
78              Checkbox[Dynamic[ViewPS]]}}]}
79          (*{Grid[{{" "," "},{"Show:                                  ",
80          RadioButtonBar[Dynamic[ViewPS],{True ,
81          False}]}}]}   *)
82
83          }, Frame -> {None, None, {{{1, 3}, {1, 1}} -> True} }]}
84       , {Grid[{{" ", "", ""}, {"                                  ",
85
86           Button["Generate plot", RunExperiment = True ],
87
88           Button["Save Plot",
89
90            CreateDialog[{TextCell["Enter a file name: "],
91              InputField[Dynamic[NameExport], String,
92               FieldHint -> "Example"],
93              Grid[{{"" , ""}, {Button["Save",
94
95                  If[Length[ExportImage] > 0,
96                   Export[Dir <> NameExport <> ".pdf", ExportImage];
97                   Print["Saved Analysis."],
98                   Print["Error: The Analysis Dont Exist"]]],
99                 Button["Close", DialogReturn[],
100                 Appearance -> {"DialogBox"}] }}]},
```

```
101             WindowTitle -> "Save Image"] ]
102           }}]}, {Grid[{{"", "", ""}, {"", ""  ,
103           ""}, {"

                ", Button["  Close  ", DialogReturn[],
104            Appearance -> {"DialogBox"}], " "}
105          }]}}], ImageSize -> {380, 500}, Scrollbars -> {False, True}]
106   , ControlPlacement -> Left
107   ]
```

Interfaz *TimeSimulationGUI*:

```
1  TimeSimulationGUI[] := Manipulate[
2   If[RunExp == True,
3     ExportImage = EvaluationTime[SystemET, PV, ICET, T];
4     RunExp = False;
5     ,
6     ExportImage
7     ],
8    {{ExportImage, Import[NotebookDirectory[] <> "ExampleSim.pdf"]},
9     ControlType -> None},
10   {{RunExp, False}, ControlType -> None},
11   {{Dir, NotebookDirectory[]}, ControlType -> None},
12   {{NameExport, "Example"}, ControlType -> None},
13   {{SystemET, {x'[t] == v[t], v'[t] == -(2 v[t] + 3 x[t])/4}},
       ControlType -> None},
14   {{ICET, {{x[t], v[t]}, {4, 2}}}, ControlType -> None},
15   {{PV, {v[t]}}, ControlType -> None},
16   {{Pre, {5}}, ControlType -> None},
17   {{T, {t, 0, 20}}, ControlType -> None},
18   Pane[Grid[{{""},
19      {Grid[{
20         {"",
21          Control[{{SystemET, {x'[t] == v[t],
```

```
22              v'[t] == -(2 v[t] + 3 x[t])/4},
23             "                System:"}}]},
24          {"",
25           Control[{{ICET, {{x[t], v[t]}, {4, 2}},
26              "Initial Conditions:"}}]},
27          {"", Control[{{PV, {v[t]}, "       Plot Variable:"}}]},
28          {"", Control[{{Pre, {5},    "              Precision:"} }]},
29          {"", Control[{{T, {t, 0, 20}, "                    Time:"
               }}]}
30          }]},
31       {Grid[{{" ", " ", " "},
32          {"                        ",
33           Button["Run Simulation", RunExp = True ],
34           Button["Save Plot",
35
36            CreateDialog[{TextCell["Enter a file name: "],
37              InputField[Dynamic[NameExport], String,
38               FieldHint -> "Example"],
39              Grid[{{"" , ""}, {Button["Save",
40
41                  If[Length[ExportImage] > 0,
42                   Export[Dir <> NameExport <> ".pdf", ExportImage];
43                   Print["Saved Simulation."],
44                   Print["Error: The Simulation Dont Exist"]]],
45                 Button["Close", DialogReturn[],
46                  Appearance -> {"DialogBox"}] }}]},
47            WindowTitle -> "Save Image"] ]
48
49           }}]},
50       {Grid[{{"", "", ""}, {"", "" ,
51           ""}, {"
52
53              ", Button["  Close  ", DialogReturn[],
```

```
52              Appearance -> {"DialogBox"}], " "}
53            }]}},
54       Frame -> {None,
55         None, {{{1, 2}, {1, 1}} -> True, {{1, 5}, {1, 1}} -> True}} ],
56       ImageSize -> {380, 500}, Scrollbars -> {False, True}]
57      ,
58      ControlPlacement -> Left]
```

Apéndice C

Apéndice C: Obtención de resultados mediante DynamicaM

En este apéndice se presentan las líneas de código con las cuales se obtuvieron los resultados presentados en el Capítulo 4.

C.1. Caso 1: Modelo clásico de un sistema eléctrico

Obtención del diagrama de bifurcación:

```
1 Definir ecuaciones diferenciales:
2
3 Ode1 = DefineODE[ODE -> \[Delta]m'[t] == \[Omega][t]];
4 Ode2 = DefineODE[ODE -> \[Omega]'[t] == 1/M*(-Dm*\[Omega][t] + Pm +
    Em*Ym*V[t]*Sin[\[Delta][t] - \[Delta]m[t] - \[Theta]m] + Em^2*Ym*
    Sin[\[Theta]m])];
5 Ode3 = DefineODE[ODE -> \[Delta]'[t] == 1/Kqw*( -Kqv2*V[t]^2 - Kqv*V
    [t] - Q0 - Q1 + Q)];
6 Ode4 = DefineODE[ODE -> V'[t] == 1/(T1*Kqw*Kpv)*(-Kqw*(P0 + P1 - P)
    + (Kpw*Kqv - Kqw*Kpv)*V[t] + Kpw*(Q0 + Q1 - Q) + Kpw*Kqv2*V[t]^2)
    ];
7
```

```
Definir ecuaciones algebraicas:

AlgebraicEquation1 = DefineAlgebraicEquation[AE -> {P -> -V[t]*Eop*
    Yop Sin[\[Delta][t] + \[Theta]op] - V[t]*Em*Ym* Sin[\[Delta][t] -
     \[Delta]m[t] + \[Theta]m] + V[t]^2 (Yop*Sin[\[Theta]op] + Ym*Sin
    [\[Theta]m]) ,Q -> V[t]*Eop*Yop*Cos[\[Delta][t] + \[Theta]op] + V[
    t]*Em*Ym*Cos[\[Delta][t] - \[Delta]m[t] + \[Theta]m] - V[t]^2*(
    Yop*Cos[\[Theta]op] + Ym*Cos[\[Theta]m]) }];!

Definir ecuaciones constantes:

Constants1 = DefineConstant[Constant -> {Kqw -> -0.03, Kqv2 -> 2.1,
    Kqv -> -2.8, Kpv -> 0.3, Kpw -> 0.4, Pm -> 1, P0 -> 0.6, P1 -> 0,
     T1 -> 8.5, Eop -> 2.5, Yop -> 8, Ym -> 5, \[Theta]op -> -0.2094,
     Em -> 1, Ym -> 5, \[Theta]m -> -0.08726, Q0 -> 1.3, M -> 0.3, Dm
     -> 0.05}];

Definir sistema:

system = DefineSystem[AlgebraicEquation -> AlgebraicEquation1,
    OrdinaryDifferentialEquations -> {Ode1, Ode2, Ode3, Ode4},
    Constant -> Constants1, StateVariables -> {{\[Delta]m, re[0, 2],
    img[0, 2]}, {\[Omega], re[-1, 2], img[-1, 2]}, {\[Delta], re[0,
    2], img[0, 2]}, {V, re[0, 3], img[0, 3]}}, BifurcationVariables
    -> {Q1}];

Definir experimento:

Exp1 = DefineExperiment[SystemODE -> system,BifRanges -> {{Q1, 10,
    11.415, 0.00005}}, Dir -> NotebookDirectory[] <> "modelo1/",
    Arguments -> {NumberOfParticles -> 70, NumberOfIterations -> 70,
    RadiusSize -> 0.1, SizeNiche -> 10, C0 -> 10, C1 -> 2.1, C2 ->
    2.1, TS -> 0.9}];
```

```
23
24 Crear DB:
25
26 BD = CreateBifurcationDiagram[Experiment -> Exp1,ComplexFixedPoints
      -> False];
27
28 Mostrar DB:
29
30 PlotBD[BD, {Q1, V}]
```

Simulaciones en el tiempo

Para generar todas las simulaciones en el dominio del tiempo basta con variar el parámetro de PlotVaribles.

```
1
2  TimeSimulation[SystemODE -> system, PlotVaribles -> {\[Delta]m[t]},
       InitialCondition -> {{\[Delta]m[t], \[Omega][t], \[Delta][t], V
      [t]}, {0.285796, 0, 0.106625, 1.22952}}, EvalutionRanges -> {t,
      1, 100}, BPConstant -> {Q1 -> 10}]
```

Diagramas de fase, campos vectoriales y secciones de Poincaré

Para observar los campos vectoriales y las secciones de Poincaré basta con poner los parámetros ShowPS y ShowVF en True.

```
1
2 PhaseDiagram[SystemODE -> system, PlotVaribles -> {\[Delta]m[t], \[
    Omega][t], V[t]}, InitialCondition -> {{\[Delta]m[t], \[Omega][t
    ], \[Delta][t], V[t]}, {0.7, -0.5, 0.3, 0.85}},
    BifurcationParameter -> {Q1 -> 10.886}, EvalutionRanges -> {t, 0,
     10}, VectorField -> {ShowVF -> True, Constants -> {\[Delta][t]
    -> 0.3}, VectorScale -> {0.05, 0.099}, VectorPoints -> 8,
    EvalutionRanges -> {{\[Delta]m[t], -0.1, 0.8}, {\[Omega][t],
```

```
    -1.6, 1.6}, {V[t], 0.83, 1.2}}}, PoincareSection -> {ShowPS ->
    False, Plane -> {\[Omega][t] == -1}, Displacement -> {0, 0, 0}}]
```

C.2. Caso 2: Sistema eléctrico de potencia con carga dinámica tipo II

Obtención del diagrama de bifurcación:

```
1  Definir ecuaciones diferenciales:
2
3  Ode1 = DefineODE[ODE -> V'[t] == 1/(T1*Kqw*Kpv)*(-Kqw*(P0 + P1 - P)
       + (Kpw*Kqv - Kqw*Kpv)*V[t] + Kpw*(Q0 + Q1 - Q) + Kpw*Kqv2*V[t]^2)
       ];
4
5  Ode2 = DefineODE[ODE -> \[Delta]m'[t] == \[Omega][t]];
6  Ode3 = DefineODE[ODE -> \[Omega]'[t] == 1/M*(-Dm*\[Omega][t] + Pm -
       ( Em*V[t] ((-gm)*Cos[\[Delta][t] - \[Delta]m[t]] - (-bm)*Sin[\[
       Delta][t] - \[Delta]m[t]]) + Em^2*gm))];
7  Ode4 = DefineODE[ODE -> \[Delta]'[t] == 1/Kqw*( -Kqv2*V[t]^2 - Kqv*V
       [t] - Q0 - Q1 + Q)];
8
9  Definir ecuaciones algebraicas:
10
11 AlgebraicEquation1 = DefineAlgebraicEquation[AE -> { P -> -(V[t]*Eop
       *((-go) Cos[-\[Delta][t]] - (-bo) Sin[-\[Delta][t]]) + V[t] Em
       ((-gm) Cos[\[Delta]m[t] - \[Delta][t]] - (-bm) Sin[\[Delta]m[t] -
        \[Delta][t]]) + V[t]^2 (go + gm)), Q -> -(-V[t]*Eop*((-go) Sin
       [-\[Delta][t]] + (-bo) Cos[-\[Delta][t]]) - V[t] Em ((-gm) Sin[\[
       Delta]m[t] - \[Delta][t]] + (-bm) Cos[\[Delta]m[t] - \[Delta][t
       ]]) - V[t]^2 (bo + bm))}];
12
13 Definir ecuaciones constantes:
```

```
14
15 Constant1 =
16   DefineConstant[Constant -> {Kpw -> 0.4, Kpv -> 0.3, Kqw -> -0.03,
       Kqv -> -2.8, Kqv2 -> 2.1, T1 -> 8.5, P0 -> 0.6, Q0 -> 1.3, P1
       -> 0, c -> 3.5, Eop -> 19.588235, Em -> 1.05, Pm -> 1, M ->
       0.01464, Dm -> 0.05, go -> 0, bo -> -0.17, gm -> 0, bm -> -5}];
17
18 Definir sistema:
19
20 system = DefineSystem[AlgebraicEquation -> AlgebraicEquation1,
     OrdinaryDifferentialEquations -> {Ode1, Ode2, Ode3, Ode4},
     Constant -> Constant1, StateVariables -> {{V, 0, 2}, {\[Delta]m,
     0, 2}, {\[Omega], -1, 2}, {\[Delta], 0, 3}}, BifurcationVariables
      -> {Q1}];
21
22 Definir experimento:
23
24 Exp1 = DefineExperiment[SystemODE -> system, BifRanges -> {{Q1, 2.1,
     3.1, 0.00005}}, Dir -> NotebookDirectory[] <> "modelo3(100X100)/"
     , Arguments -> {NumberOfParticles -> 100, NumberOfIterations ->
     100, RadiusSize -> 0.1, ZiseNiche -> 20, C0 -> 10, C1 -> 2.1, C2
     -> 2.1, TS -> 0.9}];
25
26 Crear DB:
27
28 BD = CreateBifurcationDiagram[Experiment -> Exp1, ComplexFixedPoints
     -> False];
29
30 Mostrar DB:
31
32 PlotBD[BD, {Q1, V}]
```

Simulaciones en el tiempo

Para generar todas las simulaciones en el dominio del tiempo basta con variar el parámetro de PlotVaribles.

```
TimeSimulation[SystemODE -> system, PlotVaribles -> {\[Delta]m[t]},
    InitialCondition -> {{V[t], \[Delta]m[t], \[Omega][t], \[Delta][t
    ]}, {0.87746, 0.26563, 0, 0.04682}}, EvalutionRanges -> {t, 0,
    100}, BPConstant -> {Q1 -> 2.975}]
```

Diagramas de fase, campos vectoriales y secciones de Poincaré

Para observar los campos vectoriales y las secciones de Poincaré basta con poner los parámetros ShowPS y ShowVF en True.

```
PhaseDiagram1[SystemODE -> system, PlotVaribles -> {V[t], \[Omega][t
    ], \[Delta][t]}, InitialCondition -> {{V[t], \[Delta]m[t], \[
    Omega][t], \[Delta][t]}, {0.87746, 0.26563, 0, 0.04682}},
    BifurcationParameter -> {Q1 -> 2.977}, EvalutionRanges -> {t, 0,
    100}, VectorField -> {ShowVF -> True, Constants -> {\[Delta]m[t]
    -> 0.26563}, VectorScale -> {0.1, 0.04},VectorPoints -> 8,
    EvalutionRanges -> {{V[t], 0.874, 0.8775}, {\[Omega][t], -0.04,
    0.05}, {\[Delta][t], 0.04, 0.054}}}, PoincareSection -> {ShowPS
    -> False, Plane -> {\[Delta][t] == 0.045} , Displacement -> {0,
    0, 0}}]
```

C.3. Caso 3: Sistema de dos máquinas y un bus infinito

Obtención del diagrama de bifurcación:

```
Definir ecuaciones diferenciales:

Ode1 = DefineODE[ODE -> {\[Delta]1'[t] == \[Omega]1[t]}];
Ode2 = DefineODE[ODE -> {\[Delta]2'[t] == \[Omega]2[t]}];
```

```
5  Ode3 = DefineODE[ODE -> {\[Omega]1'[t] == (1/M1) (P1 - Dm1 \[Omega
       ]1[t] - E1*E2 (B12) Sin[\[Delta]1[t] - \[Delta]2[t]] - E1 E3 (B13
       ) Sin[\[Delta]1[t] - d3] - E1 E2 (G12) Cos[\[Delta]1[t] - \[Delta
       ]2[t]] - E1 E3 (G13)*Cos[\[Delta]1[t] - d3] )}];
6  Ode4 = DefineODE[ODE -> {\[Omega]2'[t] == (1/M2) (P2 - Dm2 \[Omega
       ]2[t] - E2*E1 (B12) Sin[\[Delta]2[t] - \[Delta]1[t]] - E2 E3 (B23
       ) Sin[\[Delta]2[t] - d3] - E2 E1 (G12) Cos[\[Delta]2[t] - \[Delta
       ]1[t]] - E2 E3 (G23)*Cos[\[Delta]2[t] - d3] )}];
7
8  Definir ecuaciones constantes:
9
10 Constant1 = DefineConstant[Constant -> {M1 -> 1, Dm1 -> 0.1, E1 ->
       1, E2 -> 1, E3 -> 1, B13 -> 1, B12 -> 1.5, G13 -> -0.1, d3 -> 0,
       G12 -> -0.15, P2 -> -1.8972, M2 -> 1, Dm2 -> 0.1, B23 -> 1, G12
       -> -0.15, G23 -> -0.1}];
11
12 Definir sistema:
13
14 system = DefineSystem[OrdinaryDifferentialEquations -> {Ode1, Ode2,
       Ode3, Ode4}, Constant -> Constant1, StateVariables -> {{\[Delta]1,
        -3, 3}, {\[Delta]2, -3, 3}, {\[Omega]1, -3, 3}, {\[Omega]2, -3,
       3}}, BifurcationVariables -> {P1}];
15
16 Definir experimento:
17
18 Exp1 = DefineExperiment[SystemODE -> system, BifRanges -> {{P1, 0,
       2.47, 0.0003}}, Dir -> NotebookDirectory[] <> "Modelo3/",
       Arguments -> {NumberOfParticles -> 100, NumberOfIterations ->
       100, RadiusSize -> 0.1, SizeNiche -> 20, C0 -> 10, C1 -> 2.1, C2
       -> 2.1, TS -> 0.9}]
19
20 Crear DB:
21
```

```
22 BD = CreateBifurcationDiagram[Experiment -> Exp1,ComplexFixedPoints
       -> False];
23
24 Mostrar DB:
25
26 PlotBD[BD, {P1, \[Delta]1}]
```

Simulaciones en el tiempo

Para generar todas las simulaciones en el dominio del tiempo basta con variar el parámetro de PlotVaribles.

```
1 TimeSimulation[SystemODE -> system, PlotVaribles -> {\[Delta]1[t]},
    InitialCondition -> {{\[Delta]1[t], \[Delta]2[t], \[Omega]1[t],
    \[Omega]2[t]}, {1.06551, -0.30654, 0, 0}}, EvalutionRanges -> {t,
     1, 180}, BPConstant -> {P1 -> 2.26}]
```

Diagramas de fase, campos vectoriales y secciones de Poincaré

Para observar los campos vectoriales y las secciones de Poincaré basta con poner los parámetros ShowPS y ShowVF en True.

```
1 PhaseDiagram[SystemODE -> system, PlotVaribles -> {\[Delta]1[t], \[
    Omega]1[t]}, InitialCondition -> {{\[Delta]1[t], \[Delta]2[t], \[
    Omega]1[t], \[Omega]2[t]}, {1.051, -0.30654, 0, 0}},
    BifurcationParameter -> {P1 -> 2.26}, EvalutionRanges -> {t, 0,
    100},VectorField -> {ShowVF -> True, Constants -> {\[Omega]2[t]
    -> 0, \[Delta]2[t] -> -0.30654 }, VectorScale -> Automatic,
    VectorPoints -> Automatic, EvalutionRanges -> {{\[Delta]1[t],
    1.051, 1.0637}, {\[Omega]1[t], -0.0052, 0.0075}}},
    PoincareSection -> {ShowPS -> True, Plane -> {\[Delta]1[t] ==
    1.056},Displacement -> {0, 0, 0}}]
```

Referencias

[Aggarwal00] Aggarwal, V. Aggarwal, v. solving trascendental equations using genetics algorithms. http://web.mit.edu/varunag/www/techwrite.html, 2000. Consulta: marzo 2015.

[Barrera07] Barrera, J., Flores, J. J., y Esquivel, C. F. Generating complete bifurcation diagrams using a dynamic environment particle swarm optimization algorithm. *Journal of Artificial Evolution and Applications*, 2008, 2007.

[Barrera08] Barrera, J. A. *Análisis de Sistemas Dinámicos utilizando Herramientas de Inteligencia Artificial*. Tesis Doctoral, Universidad Michoacana de San Nicolás de Hidalgo,, 2008.

[Bejar08] Bejar, J. O. *Simulación Cualitativa sobre Diagramas de Bifurcación*. Tesis de maestría, Universidad Michoacana de San Nicolás de Hidalgo, Posgrado de Ingeniería Eléctrica, 2008.

[Bilir00] Bilir, B. *Bifurcation Analysis of Nonlinear Oscillations in Power Systems*. University of Missouri-Columbia, 2000.

[Boyce01] Boyce, W. E. y Diprima, R. C. *Elementary Differential Equations*. John Wiley and Sons, Inc., 2001.

[Brits02] Brits, R., Engelbrecht, A. P., y Bergh, F. V. D. A niching particle swarm optimizer. *En Proc. of the 4th Asia-Pacific Conference on Simulated Evolution and Learning 2002 (SEAL 2002)*. 2002.

[Doedel94] Doedel, E., Wang, X., y Fairgrieve, T. Software for continuation and bifurcation problems in ordinary differential equations. *En Technical report*. California Institute of Technology Pasadena, CA, 1994.

[Ermentrout03] Ermentrout, B. *Simulating, Analyzing, and Animating Dynamical Systems: A Guide to XPPAUT for Researchers and Students*. 2003.

[Flores06] Flores, J. J. y Proskurowski, A. Reasoning about dynamic systems and bifurcations. 2006.

[Galbis12] Galbis, Maestre, A., y Manuel. *Vector Analysis Versus Vector Calculus*. Springer, 2012.

[Kasusky02] Kasusky, R. G. *Análisis de Inestabilidades en Sistemas Eléctricos de Potencia por Medio de la Teoría de Bifurcación*. Tesis de maestría, Instituto Tecnológico de Morelia,Departamento de Ingeniería Elétrica y Electrónica, 2002.

[Kennedy95] Kennedy, J. y Eberhart, R. C. Particle swarm optimization. *En Proceedings of IEEE International Conference on Neural Networks (ICNN'95)*, págs. 1942–1948. December 1995.

[Kuznetsov98] Kuznetsov, Y. A. *Elements of Applied Bifurcation Theory*. Springer-Verlag New York, 2ª ed$^{\text{ón}}$., 1998.

[Lerm03] Lerm, A. A. P., Cañizares, C. A., y e Silva., A. S. Multiparameter bifurcation analysis of the south brazilian power system. 2003.

[López10] López, R. *Diagramas de Bifurcación de Funciones Discontinuas o no Diferenciales*. Tesis de maestría, Universidad Michoacana de San Nicolás de Hidalgo,Posgrado de Ingeniería Eléctrica, 2010.

[Méndez13] Méndez, A. R. y Madrid, G. J. G. Análisis de sistemas dinámicos con pplane8. m (matlab® toolbox). *Avances en Ciencias e Ingeniería*, 4(4):117–132, 2013.

[Mithulananthan02] Mithulananthan, N. *Hopf Bifurcation Control and Indices for Power System with Interacting Generator and FACTS Controllers.* Tesis Doctoral, Department of Electrical and Computer Engineering, 2002.

[Nayfeh95] Nayfeh, A. H. y Balachandran, B. *Applided Nonlinear Dynamics: Analytical, Computational and Experimental Methods.* WILEY-VCH, 1995.

[Ochoa16] Ochoa, A. L. G. *Puntos Fijos Complejos en Diagramas de Bifurcación.* Tesis de maestría, Universidad Michoacana de San Nicolás de Hidalgo, Posgrado de Ingeniería Eléctrica, 2016.

[Pérez07] Pérez, H. R. C. *Análisis de Bifurcaciones en Sistemas Eléctricos.* Tesis de maestría, Universidad Michoacana de San Nicolás de Hidalgo, Posgrado de Ingeniería Eléctrica, 2007.

[Polking09] Polking, J. Phase plane pplane8.m (matlab® toolbox). *Recuperado, Enero*, 10:2013, 2009.

[Rangel09] Rangel, H. R. *Cualitativización de Funciones, Aplicación a Diagramas de Bifurcación.* Tesis de maestría, 2009.

[Rasband90] Rasband, S. N. *Chaotic Dynamics of Nonlinear Systems.* Dover Publication inc. Mineola, New York, 1990.

[Seydel10] Seydel, R. *Practical Bifurcation and Stability Analysis.* Springer New York Dordrecht Heidelberg London, 3ª ed$^{\underline{\text{ón}}}$., 2010.

[Strogatz94] Strogatz, S. H. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry and Engineering.* Perseus Books Publishing, L.L.C, 1994.

[Villanueva10] Villanueva, M. L. C. *Herramienta para el análisis de sistemas dinámicos mediante diagramas de bifurcación basado en metaheurísticas.* Tesis de maestría, Universidad Michoacana de San Nicolás de Hidalgo, Posgrado de Ingeniería Eléctrica, 2010.

[Villanueva12] Villanueva, M. V., Soria, M. B., y Cantarero, T. Á. Tutorial de introducción a matlab, 2012.

[Walve86] Walve, K. Modelling of power system components at severe disturbances. *En Proceedings of the International Conference on Large High Voltage Electric Systems (CIGRE)*. 1986.

[Wang94] Wang, H. O., Abed, E. H., y Hamdan, A. M. A. Bifurcations, chaos, and crises in voltage collapse of a model power system. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 41(4):294–302, Apr 1994. ISSN 1057-7122. doi: 10.1109/81.285684.

[Wiggins03] Wiggins, S. *Introduction to Applied Nonlinear Dynamical Systems*. Springer, 2ª ed$^{\underline{\text{ón}}}$., 2003.

[XPP03] *XPPAUT5.41 The Differential Equations Tool*, Julio 2003.