UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE MATEMÁTICA

Implementación de un Esquema Predictor-Corrector para Continuación Numérica

Trabajo Especial de Grado presentado ante la ilustre Universidad Central de Venezuela por el **Br. Javier Alejandro Machado Sánchez** para optar al título de Licenciado en Matemática.

Tutora: Dra. Zenaida Castillo

Caracas, Venezuela
Mayo 2014

Nosotros, los abajo firmantes, designados por la Universidad Central de Venezuela como integrantes del Jurado Examinador del Trabajo Especial de Grado titulado "**Nombre del Trabajo Especial de Grado**", presentado por el **Br. Javier Alejandro Machado Sánchez**, titular de la Cédula de Identidad **V-18.539.151**, certificamos que este trabajo cumple con los requisitos exigidos por nuestra Magna Casa de Estudios para optar al título de **Licenciado en Matemática**.

Dra. Zenaida Castillo
Tutor

Profa. Carmen Da Silva
Jurado

Prof. Adrian Bottini
Jurado

Dedicatoria

A mis Abuelos.

Agradecimiento

Quiero agradecer a las personas sin las cuales este trabajo no hubiese sido posible. En primer lugar a mis abuelos, Deyzi y Fernando, por apoyarme en cada momento de mi vida, este trabajo es para ustedes. A mis padres, Elein y Julio. A mi tutora, la profesora Zenaida Castillo, por compartir sus conocimientos, por su guía, pero sobre todo por su paciencia. A todos mis profesores durante la carrera, por los conocimientos que me impartieron. A todas estas personas, mis más profundas y sentidas palabras de agradecimiento.

Resumen

En este trabajo, se estudiaron diferentes métodos para encontrar conjuntos solución de un sistema de ecuaciones $\mathsf{G} : \mathbb{R}^{n+1} \longrightarrow \mathbb{R}^n$ no lineal que depende de un parámetro. Se desarrolló un método de continuación por pseudo-longitud de arco donde se presenta una alternativa para calcular el vector tangente y detectar ciertos puntos de interés de una curva solución del sistema mediante el cálculo del espectro de la matriz jacobiana de G. Explicamos las consideraciones que se tomaron en cuenta para implementar este método en el ambiente *Matlab* versión 7.8.0.347 (R2009a), con dos maneras diferentes de calcular el espectro de la jacobiana de G, y por último, presentamos varias pruebas numéricas para comprobar la estabilidad del método y determinar las condiciones para el uso de alguna de las dos implementaciones.

Índice general

Introducción

En muchas áreas científicas se modelan matemáticamente diferentes tipos de fenómenos y es frecuente que se requiera calcular y analizar el conjunto solución de un sistema de ecuaciones no lineal, el cual puede depender de uno o varios parámetros. Por ejemplo, si se quiere estudiar el crecimiento de una población al variar un parámetro o, si se quisiera estudiar la variación de una reacción química a medida que se varia uno de sus compuestos.

En este trabajo, se estudiará un sistema de ecuaciones de la forma $G(x, \alpha) = \vec{0}$ donde $G : \mathbb{R}^n \times \mathbb{R} \longrightarrow \mathbb{R}^n$, $x \in \mathbb{R}^n$, y $\alpha \in \mathbb{R}$. Generalmente, una solución aislada (x_0, α_0) no es suficiente información sobre el fenómeno; por esto, se estudian diferentes soluciones de $G(x, \alpha) = \vec{0}$ mientras se varía el parámetro α en un intervalo conocido. Además, se quiere identificar diferentes puntos de interés que pueden representar cambios importantes en el fenómeno que se esté estudiando, algunos de ellos suelen estar ligados a la estabilidad del método. Entre los puntos de interés más comunes estan los *Turning Points* (puntos donde la curva cambia de sentido, indicando que para el mismo valor del parámetro existen otras soluciones), los *Bifurcation Points* (puntos donde varias curvas se intersectan) y *Hoft Points* (puntos que señalan el comienzo de soluciones periódicas). En [**3, 11, 9, 6**]

Existen varios métodos con los cuales se calculan curvas de soluciones de la función G, entre los cuales se encuentran los métodos de Continuación Numérica, se puede consultar [**2, 7, 10, 4**] para estudiar sobre los métodos de continuación. Estos métodos consisten en darle valores a una de las variables del problema (el parámetro) y tomando en cuenta la solución previamente calculada hallar una aproximación de una nueva solución; este procedimiento se repite hasta agotar los posibles valores del parámetro. Usualmente, se usan técnicas de tipo predictor-corrector para implementar los métodos de Continuación Numérica. Primero se estiman las coordenadas de una solución usando información de soluciones previas (paso

predictor) y luego se usa esta estimación para calcular las coordenadas de la siguiente solución (paso corrector). De esta manera, los métodos de Continuación Numérica existentes difieren en la forma de escoger el predictor y en el algoritmo iterativo que utilicen para hallar un cero de $G(x, \alpha)$.

En el paso corrector puede utilizarse cualquier algoritmo que calcule un cero de una función, por ejemplo el método de Newton:

Algoritmo 1 Método de Newton

Datos de Entrada: x_0, tol.

Datos de Salida: Solución x_s

1: $x_s = x_0$
2: **Mientras** $\|G(x_s)\| >$ tol **hacer**
3: $\quad x_0 = x_s$
4: $\quad$ Resolver el sistema $G_{x_0} z = G(x_0)$
5: $\quad x_s = x_0 - z$
6: **Fin Mientras**

En este método se resuelve un sistema de ecuaciones lineales cuya matriz asociada es la matriz jacobiana de G en la variable x, Gx, la cual debería ser no-singular para que el método no falle. En la práctica, uno de los métodos de continuación más usados es el denominado pseudo-longitud de arco, el cual extiende el sistema original para evitar la no-singularidad de G_x.

En el primer capítulo estudiaremos como se realizan las gráficas de bifurcación, definiremos ciertos puntos de interés y daremos una visión general de los métodos de continuación numérica utilizados en la actualidad. En el segundo capítulo estudiaremos la propuesta presentada en este trabajo, que consiste en un método para calcular el vector tangente de la curva solución y detectar puntos de interés. En el tercer capítulo presentaremos el algoritmo de continuación de nuestra propuesta, junto con las consideraciones que se tomaron en cuenta para implementarlo en el ambiente *Matlab* versión 7.8.0.347 (R2009a). En el cuarto capítulo veremos ciertas pruebas numéricas que realizamos con el método y en el quinto capítulo presentaremos las conclusiones.

1. Continuación Numérica

Los métodos de continuación son un tópico de gran importancia en algunas aplicaciones de la ciencia y la ingeniería. Este capítulo presenta una visión general sobre los métodos de continuación numérica más utilizados en la actualidad.

En muchas aplicaciones industriales se requiere la discretización de una ecuación en derivadas parciales, o EDP. Esta discretización da origen a un sistema de ecuaciones no lineal parametrizado y se desea estudiar el comportamiento de las soluciones de este sistema cuando se varía uno de sus parámetros. Por esto, consideraremos un sistema de ecuaciones no lineal de la forma:

$$G(\vec{x}, \alpha) = \vec{0} \tag{1.1}$$

donde $G : \mathbb{R}^n \times \mathbb{R} \longrightarrow \mathbb{R}^n$, $\vec{x} \in \mathbb{R}^n$ y $\alpha \in I = (a, b) \subset \mathbb{R}$, donde I es un intervalo conocido. Luego, para diferentes valores de $\alpha_i \in I$ se desea calcular un $\vec{x}_i \in \mathbb{R}^n$ tal que $G(\vec{x}_i, \alpha_i) = \vec{0}$. Una solución $(\vec{x}_i, \alpha_i)$ del sistema (1.1) se denomina *punto estacionario* o *punto de equilibrio*.

Existen diferentes métodos para calcular los puntos de equilibrio para diferentes valores del parámetro α, entre los cuales se encuentran los métodos de continuación. Estos se caracterizan por resolver (1.1) para un valor específico de α conociendo la solución en un punto previo, de allí viene el nombre de método de continuación. Este proceso se repite hasta que se hallan calculado suficientes puntos de equilibrio como para tener una buena aproximación del conjunto solución. En la sección 1.3 discutiremos sobre algunos métodos de continuación utilizados en la actualidad.

1.1 Diagrama de Bifurcación. El conjunto de soluciones obtenidas a partir de la ejecución de algún método de continuación puede ser utilizado para graficar en $\mathbb{R}^2$ o $\mathbb{R}^3$ relaciones entre el parámetro α y una representación en $\mathbb{R}$ o $\mathbb{R}^2$ del vector $\vec{x}$, obteniendo así lo que se conoce como *diagrama de bifurcación*, la cual proporciona una representación visual del comportamiento de una curva solución de (1.1) a medida que se varía el parámetro, lo que facilita el estudio y análisis de las soluciones. Por ejemplo, al tener un conjunto $\{(\vec{x}_i, \alpha_i)\}_{i=1}^n$ de soluciones de (1.1), donde $\vec{x}_i = (x_1^{(i)}, x_2^{(i)}, \cdots, x_n^{(i)})$, se puede realizar un diagrama de bifurcación con los pares ordenados $([\vec{x}_i], \alpha_i)$, donde $[\vec{x}_i]$ puede representar la norma euclidea $[\vec{x}_i] = \|\vec{x}_i\|_2$, o también $[\vec{x}_i] = x_j^{(i)}$ para alguna variable x_j de interés.

Veamos algunos ejemplos sencillos de continuación:

Ejemplo 1.1.

Consideremos la siguiente función:

$$\begin{aligned} &f : \mathbb{R}^2 \longrightarrow \mathbb{R} \quad \text{dada por} \\ &f(x, y) = xy - 1 \end{aligned}$$

La jacobiana de f viene dada por:

$$f_u = (x \quad y)$$

Al ejecutar un código de continuación sobre la función f, obtenemos una serie de puntos (x, y), cuya gráfica es la hipérbola definida por la ecuación $y = \frac{1}{x}$. En la figura 1.1, podemos ver la gráfica obtenida. Note que, en el caso de la hipérbola se tendrán dos curvas solución que no se tocan nunca; para poder recorrer ambas curvas se deberá tener por lo menos una solución sobre cada una de las curvas para poder obtener la gráfica completa.

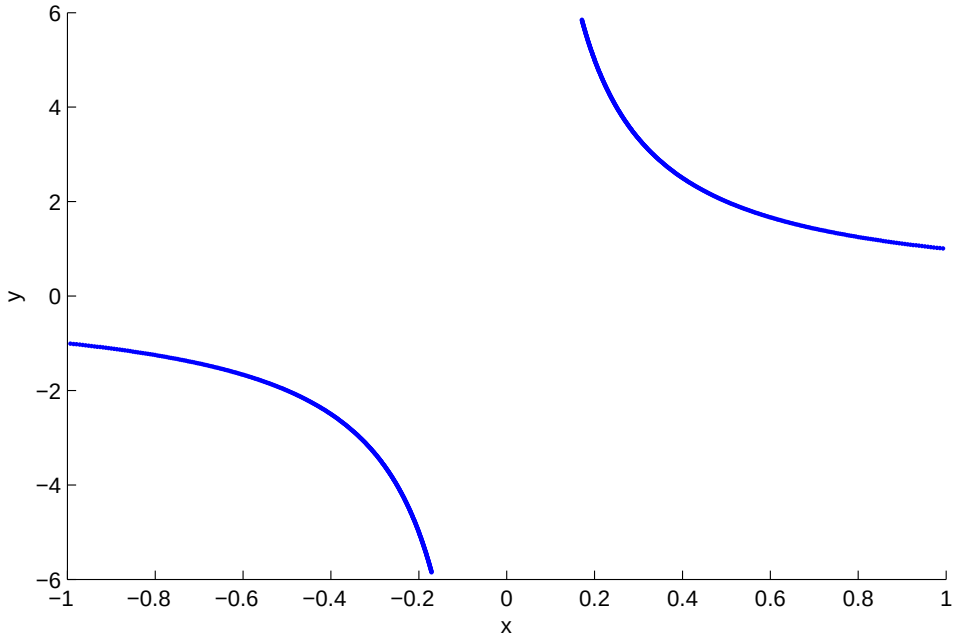

FIGURA 1.1. Diagrama de la Hipérbola

EJEMPLO 1.2.

El siguiente ejemplo contiene dos curvas solución que se intersectan. Consideremos la función:

$$f : \mathbb{R}^2 \longrightarrow \mathbb{R} \quad \text{dada por}$$
$$f(x, y) = (x^2 + y^2 - 1)(y - x)$$

La jacobiana de f viene dada por:

$$f_u = (2x(y - x) - (x^2 + y^2 - 1) \quad 2y(y - x) + (x^2 + y^2 - 1))$$

Al ejecutar un código de continuación sobre la función f, obtenemos una serie de puntos (x, y), cuya gráfica es un círculo centrado en el origen de radio 1, y también la recta definida por $y = x$. Note que, en este ejemplo las dos curvas solución del sistema se intersectan en dos puntos, tal como puede apreciarse en la figura 1.2

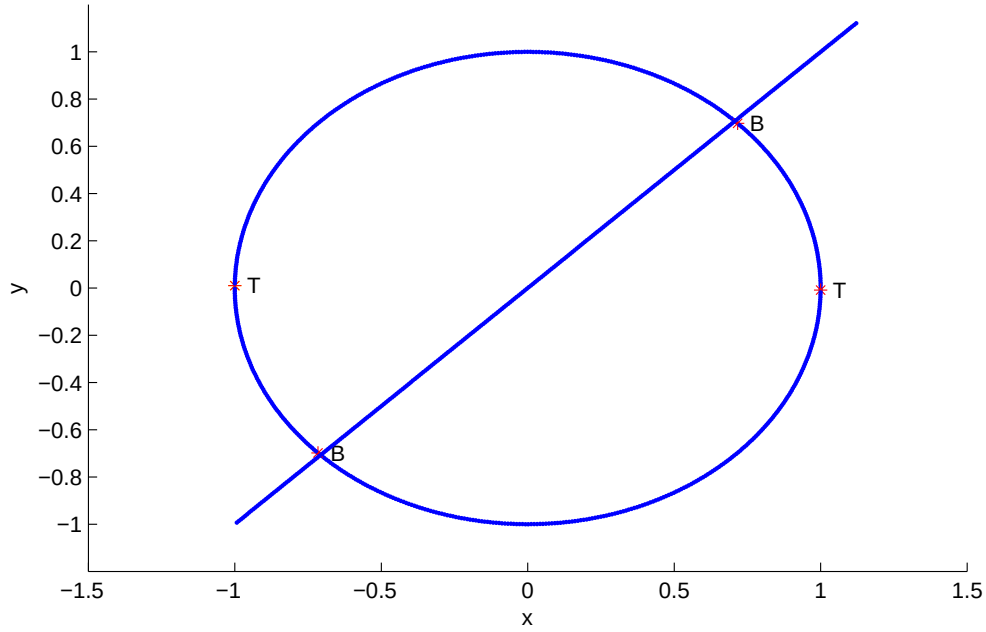

FIGURA 1.2. Diagrama del Círculo y la Recta

1.2 Puntos de Interés. Además de calcular soluciones del sistema de ecuaciones (1.1), es importante poder detectar e identificar diferentes puntos de interés sobre las curvas solución, ya que estos representan cambios importantes en el fenómeno que se esté estudiando y además, es frecuente que estén relacionados con la estabilidad del método de continuación que se esté utilizando. En este trabajo, realizaremos una breve descripción de tres (3) puntos de interés: *Turning Points, Bifurcation Points y Hoft Points*. Para un estudio detallado de estos puntos, se puede consultar [**2, 3, 6, 13**].

Sea $u = (\vec{x}, \alpha) \in \mathbb{R}^n \times \mathbb{R}$. En esta sección, supondremos que $G : \mathbb{R}^n \times \mathbb{R} \longrightarrow \mathbb{R}^n$ es una función diferenciable, denotaremos por $g_i, \quad i = 1, \cdots, n$ las funciones coordenadas de la función G y G_u denotará su matriz Jacobiana, definimos:

$$G_x = \begin{pmatrix} \frac{\partial g_1}{\partial x_1} & \cdots & \frac{\partial g_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_n}{\partial x_1} & \cdots & \frac{\partial g_n}{\partial x_n} \end{pmatrix} \quad y \quad G_\alpha = \begin{pmatrix} \frac{\partial g_1}{\partial \alpha} \\ \vdots \\ \frac{\partial g_n}{\partial \alpha} \end{pmatrix}$$

DEFINICIÓN 1.2.1. Diremos que u es una solución regular de (1.1) si $G(u) = \vec{0}$ y $Rg(G_u) = n$.

DEFINICIÓN 1.2.2. Diremos que $u_0 \in \mathbb{R}^{n+1}$ es un punto singular de (1.1) si u_0 es un punto de equilibrio y $G_x(u_0)$ es singular.

Observación:

- Si la matriz G_x es singular, entonces $det(G_x) = \lambda_1\lambda_2\cdots\lambda_n = 0$, donde λ_i son los autovalores de la matriz $\forall\ i = 1, \cdots, n$. Esto implica que, al menos uno de los autovalores es cero.

La detección de puntos singulares se puede realizar a través de funciones de control $\tau(u)$ que permitan saber si el punto actual está cerca de un punto singular. Por ejemplo, se puede considerar $\tau(u) = det(G_x(u))$ y monitorear los puntos en los que τ se anula. Sin embargo, a medida que aumenta el número de variables del sistema de ecuaciones, el cálculo del determinante se vuelve más complejo e inestable numéricamente. Otra alternativa es realizar el cálculo de los autovalores de la matriz G_x que estén cercanos a cero.

DEFINICIÓN 1.2.3. Diremos $u_0 \in \mathbb{R}^{n+1}$ es un Simple Turning Point, si se cumple que u_0 es un punto singular de (1.1) y $dim(Kernel(G_u)) = 1$.

Observaciones:

- Note que, si $dim(Kernel(G_u)) = 1$ entonces se debe cumplir que $G_\alpha \notin Im(G_x)$.

- Los *Simple Turning Points* son los puntos en los que la curva solución cambia de sentido en la dirección del parámetro α.

- Existen diferentes tipos de *Turning Points*, la definición 1.2.3 coincide con las definiciones de *Fold point* o *Simple Turning Point* dadas en [**2, 3, 6, 13**].
- Note que, aunque un punto u sea *Simple Turning Point*, sigue siendo una solución regular de (1.1).

DEFINICIÓN 1.2.4. Sea $h : I \subset \mathbb{R} \longrightarrow \mathbb{R}^{n+1}$ una curva diferenciable. Diremos que h es una curva solución del sistema (1.1) si se cumple que $G(h(t)) = \vec{0} \quad \forall\, t \in I$.

Ahora daremos una definición de *Bifurcation Points*. En estos puntos se intersectan dos curvas solución del sistema de ecuaciones.

DEFINICIÓN 1.2.5. Sean $h_i : I_i \subset \mathbb{R} \longrightarrow \mathbb{R}^{n+1}$ para $i \in \{1, 2\}$ dos curvas solución de (1.1) que se intersectan en $u_0 \in \mathbb{R}^{n+1}$, es decir, existen $t_1 \in I_1$ y $t_2 \in I_2$ tales que $h_1(t_1) = h_2(t_2) = u_0$. Diremos que u_0 es un Bifurcation Point si existen $U_1 \subset I_1$ y $U_2 \subset I_2$ entornos de t_1 y t_2 respectivamente, tales que $Im(h_1|_{u_1}) \cap Im(h_2|_{u_2}) = \{u_0\}$.

La condición $Im(h_1|_{u_1}) \cap Im(h_2|_{u_2}) = \{u_0\}$ de la definición 1.2.5 permite garantizar que sean dos curvas diferentes que pasan por u_0, y no dos parametrizaciones de la misma curva. La figura 1.3 muestra un ejemplo de un *Bifurcation Point*.

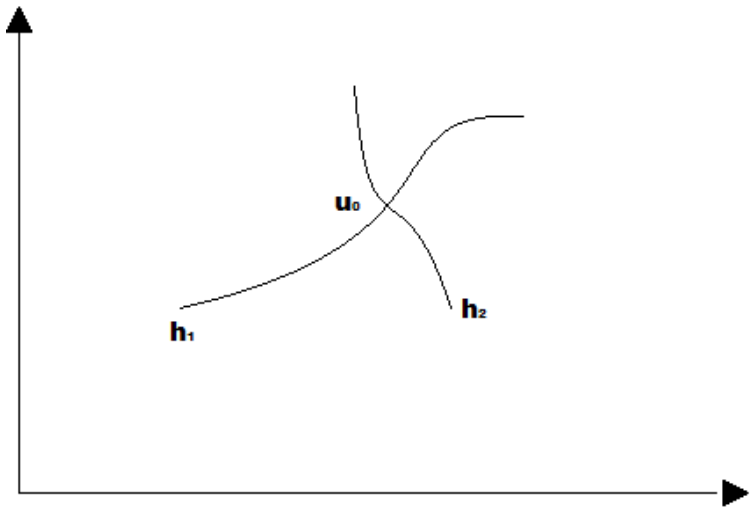

FIGURA 1.3. Bifurcation Point

Observaciones:

- A pesar de que solo se requieren dos curvas para que un punto sea *Bifurcation Point*, en general se debe considerar que pueden pasar mas de dos curvas por el mismo punto.

- Si u_0 es un *Bifurcation Point* y los vectores tangentes $h_1'(t_1)$ y $h_2'(t_2)$ son linealmente independientes, por regla de la cadena:

$$G(h_i(t)) = \vec{0} \quad \Rightarrow \quad G_u(h_i(t))h_i'(t) = \vec{0}$$

De manera que, evaluado en t_1 y t_2

$$G_u(u_0)h_1'(t_1) = \vec{0} \quad y \quad G_u(u_0)h_2(t_2) = \vec{0}$$

Esto implica que $h_1'(t_1), h_2'(t_2) \in \mathbf{Kernel}(G_u(u_0))$, siendo estos vectores linealmente independientes, tenemos que $\mathbf{dim}(\mathbf{Kernel}(G_u(u_0))) \geq 2$. Así, concluimos que: si u_0 es un *Bifurcation Point* y los vectores $h_1'(t_1)$ y $h_2'(t_2)$ son linealmente independientes, entonces u_0 es un punto singular de (1.1).

DEFINICIÓN 1.2.6. Sea $u_0 = (\vec{x}_0, \alpha_0) \in \mathbb{R}^{n+1}$ un *Bifurcation Point*, diremos que u_0 es un *Simple Bifurcation Point* si se cumple lo siguiente:

(1) $G(u_0) = \vec{0}$
(2) $\mathbf{dim}(G_x) = n - 1$
(3) $G_\alpha \in \mathrm{Im}(G_x)$

Las condiciones (2) y (3) implican que la matriz jacobiana G_{u_0} debe tener rango $n - 1$. Cuando un punto u_0 es singular, entonces puede ocurrir que este punto sea un *Simple Turning Point* o un *Simple Bifurcation Point* o incluso ambos. Dependiendo del método de continuación numérica, se usan diferentes maneras para diferenciar entre ambos y poder detectarlos. Ahora, veremos una definición de *Hoft Point*.

DEFINICIÓN 1.2.7. Sea u_0 un punto de equilibrio de (1.1), diremos que u_0 es un Hoft Point si la matriz G_x tiene un autovalor imaginario puro.

Cuando una curva solución tiene un *Hoft Point*, entonces se sabe que la curva tendrá soluciones periódicas.

1.3 Métodos de Continuación. Los Métodos de Continuación están basados en un esquema predictor-corrector, el cual consiste en dos pasos: Generar una aproximación a la solución utilizando información previa (paso predictor) y luego, usar esta predicción como aproximación inicial para algún método que resuelva un sistema de ecuaciones no lineal (paso corrector).

A continuación, se estudian varios métodos de continuación. En estos métodos consideraremos que se conoce un punto de equilibrio $(\vec{x}_0, \alpha_0)$ de (1.1).

1.3.1 Continuación de Orden Cero. En este método, el paso predictor consiste en fijar el parámetro $\alpha_i = \alpha_{i-1} + \Delta\alpha$ con $i = 1, \cdots, N$ (siendo N el número de soluciones que se desea obtener) y entonces definimos el predictor $p_i = (\vec{x}_{i-1}, \alpha_i)$. Luego, en el paso corrector se resuelve el sistema $G(p_i) = G(\vec{x_i}, \alpha_i) = \vec{0}$. Note que, fijando el parámetro α_i, este sistema es cuadrado, por lo que se puede utilizar el método de Newton para resolverlo. Ahora, presentamos un algoritmo de continuación de orden cero tomado de [**10**]:

Algoritmo 2 Continuación de Orden Cero

Datos de Entrada: $(\vec{x}_0, \alpha_0)$, $\Delta\alpha$, N.

Datos de Salida: $\{(\vec{x}_i, \alpha_i)\}_{i=1,\cdots,N}$

1: **Para** $i = 1, 2, \cdots, N$ **hacer**

2: $\quad \alpha_i = \alpha_{i-1} + \Delta\alpha$

3: $\quad p_i = (\vec{x}_{i-1}, \alpha_i)$

4: $\quad$ Resolver $G(\vec{x}_i, \alpha_i) = \vec{0}$ $\quad$ usando p_i

5: **Fin Para**

Este método puede presentar dos complicaciones: si se utiliza el método de Newton o algún método similar en el paso 3, no se podrá obtener solución en los puntos singulares del sistema. El otro problema es que, a pesar de que el método de Newton tiene una convergencia cuadrática, se necesita que el iterado inicial esté muy cerca de la solución, por lo que se necesitaría un mejor vector predictor para que el método sea eficiente.

1.3.2 Continuación de Orden 1. Este método permite construir un predictor que está más cerca de la solución, utilizando el *Teorema de la Función Implicita* sobre la función G.

TEOREMA 1.3.1 (Función Implícita). *Sea* $D \subset \mathbb{R}^{n+1}$ *un abierto y* $G : D \longrightarrow \mathbb{R}^n$ *una función de clase* C^1. *Si para algún* $\vec{x}_0 \in \mathbb{R}^n$ *y algún* $\alpha_0 \in \mathbb{R}$ *se tiene que*

- $G(\vec{x}_0, \alpha_0) = \vec{0}$
- $G_x(\vec{x}_0, \alpha_0)$ *tiene inversa*

Entonces existe un entorno I *de* α_0, *un entorno* V *de* $(\vec{x}_0, \alpha_0)$ *y una función* $g : I \longrightarrow \mathbb{R}^n$ *de clase* C^1 *tal que:*

(a) $g(\alpha_0) = \vec{x}_0$

(b) $(g(\alpha), \alpha) \in D \quad \forall\, \alpha \in I$

(c) $G(g(\alpha), \alpha) = \vec{0} \quad \forall\, \alpha \in I$

(d) *Si* $\alpha \in I$ *y* $(\vec{x}, \alpha) \in V$, *entonces el sistema* $G(\vec{x}, \alpha) = \vec{0}$ *tiene solución única y está dada por* $\vec{x} = g(\alpha)$. *Además:*

$$g'(\alpha) = -(G_{\vec{x}}(g(\alpha), \alpha))^{-1} G_\alpha(g(\alpha), \alpha) \tag{1.2}$$

Observaciones:

- El resultado (c) garantiza que existe una curva solución del sistema (1.1) y esta curva viene dada por $(g(\alpha), \alpha)$.
- El resultado (d) nos dice que la curva $(g(\alpha), \alpha)$ es la única solución de (1.1) en el entorno V.
- La ecuación (1.2) nos permite calcular el vector tangente a g, el cual nos permite construir una mejor aproximación para el vector predictor resolviendo el siguiente el sistema:

$$G_{\vec{x}}(g(\alpha), \alpha) g'(\alpha) = -G_\alpha(g(\alpha), \alpha) \tag{1.3}$$

En la práctica se debe resolver el sistema (1.3) en lugar de calcular la inversa de la matriz G_x, ya que este cálculo puede ser muy complicado a medida de que se aumente el orden de la matriz.

Para realizar continuación de orden 1, en el paso predictor se define $\alpha_i = \alpha_{i-1} + \Delta\alpha$ de la misma forma que en continuación de orden cero y definimos $\vec{x}_i = \vec{x}_{i-1} + (\Delta\alpha)g'(\alpha_{i-1})$ donde $g'(\alpha_{i-1})$ es el vector solución del sistema (1.3). Luego, nuestro vector predictor es $p_i = (\vec{x}_i, \alpha)$. De la misma forma que en continuación de orden cero, se fija la variable α_i, lo cual permite resolver un sistema de ecuaciones cuadrado en el paso corrector. Ahora, presentamos un algoritmo de continuación de orden 1:

Algoritmo 3 Continuación de Orden uno

Datos de Entrada: $(\vec{x}_0, \alpha_0)$, $\Delta\alpha$, N.

Datos de Salida: $\{(\vec{x}_i, \alpha_i)\}_{i=1,\cdots,N}$

1: **Para** $i = 1, 2, \cdots, N$ **hacer**

2: $\alpha_i = \alpha_{i-1} + \Delta\alpha$

3: Resolver $G_{\vec{x}}(\vec{x}_{i-1}, \alpha_{i-1})g'(\alpha_i) = -G_\alpha(\vec{x}_{i-1}, \alpha_{i-1})$

4: $\vec{x}_i = \vec{x}_{i-1} + (\Delta\alpha)g'(\alpha_i)$

5: $p_i = (\vec{x_i}, \alpha_i)$

6: Resolver $G(\vec{x}_i, \alpha_i) = \vec{0}$

7: **Fin Para**

Este método presenta una mejora con respecto a continuación de orden 0, ya que permite calcular un predictor más cerca de la solución y por esto, el paso corrector será más eficiente. Sin embargo, en los puntos singulares del sistema (1.3) se pueden presentar complicaciones en el predictor y en el corrector, ya que se debe resolver un sistema de ecuaciones lineales donde la matriz de los coeficientes es G_x.

1.3.3 Continuación por Pseudo-Longitud de Arco. En los métodos que hemos estudiado hasta ahora, se varía el parámetro α una cierta distancia $\Delta\alpha$, para movernos sobre la curva solución. Este método de continuación presenta una ventaja sobre los anteriores, ya

que la manera en la que se avanza sobre la curva es mediante una estimación de la longitud de arco del siguiente punto. Esto permite que cuando se encuentre en la curva un *Simple Turning Points*, aún se pueda ejecutar el paso corrector sobre una matriz que no sea singular. Por esto, definimos la siguiente función,

$$\begin{aligned} &F : \mathbb{R}^{n+2} \longrightarrow \mathbb{R}^{n+1} \\ &F(\vec{x}, \alpha, s) = (G(\vec{x}, \alpha), \Gamma(\vec{x}, \alpha, s)), \end{aligned} \tag{1.4}$$

donde $\Gamma : \mathbb{R}^{n+2} \longrightarrow \mathbb{R}$ es una función diferenciable y el parámetro s denota la longitud de arco. Tenemos que, si $F(\vec{x}_0, \alpha_0, s_0) = \vec{0}$, entonces $G(\vec{x}_0, \alpha_0) = \vec{0}$. Se agrega esta función Γ de manera tal que, si se encuentra un *Simple Turning Point* $u_0 = (\vec{x}_0, \alpha)$ a lo largo de la curva, la matriz jacobiana de F tenga rango n+1. Luego, de la misma manera que en los casos anteriores, fijando el parámetro longitud de arco, s, se puede resolver un sistema de ecuaciones usando la matriz F_u.

Existen diferentes funciones Γ que se pueden usar para este caso, y la elección de la misma, depende de las prioridades que se tenga en el momento de ejecutar un código de continuación.

Ahora, veremos como contruir el predictor para este tipo de continuación. Siempre que G_u tenga rango n, se pueden reordenar las columnas de la matriz G_u de manera que se cumplan las hipótesis del teorema de la función implícita (sin importar el parámetro). Esto implica que existe una curva $g : I \subset \mathbb{R} \longrightarrow \mathbb{R}^{n+1}$ tal que $G(g(t)) = \vec{0}\ \forall\ t \in I$. Sin pérdida de generalidad, supongamos que la curva está parametrizada por longitud de arco, por lo que $I = (0, l)$, donde l es la longitud de la curva. Luego:

$$G(g(s)) = \vec{0} \Rightarrow G_u g'(s) = \vec{0} \quad \forall\ s \in I. \tag{1.5}$$

El sistema (1.5) tiene solución distinta de cero, ya que la matriz G_u tiene rango n. El vector solución de (1.5) define el vector tangente de la curva solución g, salvo la magnitud y el sentido. Después de que se haya realizado una iteración y se tenga un vector tangente en

un punto anterior, al sistema (1.5) se le puede agregar la ecuación $< g'(s_i), g'(s_{i-1}) >= 1$ por lo que tenemos:

$$\begin{cases} G_u g'(s_i) = \vec{0}, \\ < g'(s_i), g'(s_{i-1}) >= 1. \end{cases} \tag{1.6}$$

Esta ecuación permite que se mantenga la orientación con la que se está recorriendo la curva, y además, se tiene un sistema de $n+1$ ecuaciones y $n+1$ incógnitas. Para poder definir un predictor usando el vector obtenido en el sistema (1.6), se debe definir el valor del parámetro longitud de arco (la distancia en la que se quiere avanzar en la curva). Esto es debido a que, si se usa una magnitud muy pequeña, cuando la curva sea lo suficientemente estable se estaría realizando cálculos innecesarios, ya que se puede obtener la información necesaria sin muchas iteraciones. Pero también puede ocurrir lo contrario, si la magnitud es muy grande se puede perder información valiosa en el proceso. En este trabajo, mantendremos este parámetro fijo (stepsize), estudiando principalmente la detección de los puntos singulares que mencionamos en la sección anterior.

Ahora, una vez definido el *stepsize* Δs , podemos definir el predictor de la forma $p_i = \Delta s \frac{g'(s_i)}{\|g'(s_i)\|} + (\vec{x}_{i-1}, \alpha_{i-1})$, donde $g'(s_i)$ es el vector solución del sistema (1.6).

Algoritmo 4 Algoritmo de Continuación de Pseudo-Longitud de Arco

Datos de Entrada: $(\vec{x}_0, \alpha_0)$, N.

Datos de Salida: $\{(\vec{x}_i, \alpha_i)\}_{i=1,\cdots,N}$

1: **Para** $i = 1, 2, \cdots, N$ **hacer**
2: Definir el vector g'_i resolviendo el sistema (1.6)
3: Definir el Δs (*stepsize*)
4: $p_i = \Delta s \frac{g'_{s_i}}{\|g'_{s_i}\|} + (\vec{x}_{i-1}, \alpha_{i-1})$
5: Resolver $F(\vec{x}_i, \alpha_i, \Delta s) = \vec{0}$
6: **Fin Para**

Observaciones:

- En el paso 1, cuando i=1, se tendrá que resolver un sistema sobredeterminado ya que no se tendrá un vector tangente de la iteración anterior.
- En el paso 4, al haber fijado el *stepsize*, se puede resolver un sistema de n+1 ecuaciones y n+1 incógnitas, donde además la matriz F_u no es singular para ningún *Simple Turning Point*.

En el siguiente capítulo, daremos algunos preliminares sobre los autovalores y autovectores, que facilitarán el estudio de nuestro método de continuación.

2. Autovalores y Autovectores

Uno de los principales aportes de este trabajo es definir un módulo predictor que produzca un buen iterado inicial para el paso corrector. Adicionalmente, deseamos detectar en este módulo predictor la prescencia de puntos de interés, tarea que puede facilitarse conociendo parte del espectro de una matriz definida a partir de la matriz jacobiana. En este capítulo, se desarrollan los conceptos teóricos de autovalores y autovectores y también algunos métodos para calcularlos, que nos permitirán entender las bases de la propuesta para el módulo predictor.

Primero, veremos los conceptos necesarios para entender lo que significan los autovalores y autovectores de una matriz, y su importancia en este trabajo. Supondremos que $A \in \mathbb{C}^{n\times n}$ es una matriz, $x \in \mathbb{C}^n$ un vector distinto de cero y sea $\lambda \in \mathbb{C}$.

Definición: x es un autovector de la matriz A y λ es su autovalor correspondiente, si se cumple que:

$$Ax = \lambda x$$

Al par (x, λ) que safisface $Ax = \lambda x$ se le denomina autopar. El espectro de una matriz A es el conjunto formado por todos sus autovalores. Este conjunto también puede ser descrito como:

$$\Lambda(A) = \{z \in \mathbb{C} : \|(zI - A)\| = \infty\} = \{z \in \mathbb{C} : (zI - A) \text{ es singular}\}$$

Algunas propiedades importantes de los autovalores y autovectores son las siguientes:

- Si x es un autovector de A y λ es su autovalor correspondiente, entonces αx es un autovector asociado a λ, donde $\alpha \in \mathbb{C}$ y $\alpha \neq 0$

- $\mathtt{det}(A) = \lambda_1, \cdots, \lambda_n$, donde $\lambda_1, \cdots, \lambda_n$ son todos los autovalores de la matriz A, y $\mathtt{det}(A)$ representa el determinante de A.

- $\mathtt{traza}(A) = \sum_{i=1}^{n} \lambda_i$, donde $\lambda_1, \cdots, \lambda_n$ son todos los autovalores de la matriz A, y $\mathtt{traza}(A)$ representa la traza de la matriz A.

- La matriz A es singular si y solo si A tiene un autovalor igual a cero. Si la matriz A es no singular y (x, λ) es un autopar de A, entonces $(x, \frac{1}{\lambda})$ es autopar de la matriz A^{-1}.

En diferentes problemas de la ciencia y la ingenieria, se utiliza la información de los autovalores y autovectores de una matriz para describir el fenómeno que se está estudiando. En nuestro trabajo, los autovalores cercanos a cero de una matriz, definida a partir de la matriz jacobiana de G, nos permitirá detectar si nos estamos acercando a un punto de interés. También, los autovectores asociados a estos autovalores nos permitirán definir el vector tangente a la curva solución y como diferenciar entre los puntos *STP* y *SBP*. Supongamos que $\lambda = 0$ es un autovalor asociado a un autovector x de la matriz A, entonces:

$$Ax = \lambda x = 0x = \vec{0}$$

Esto implica que el vevtor x se encuentra contenido en el $\mathtt{Kernel}$ de la matriz A. Por esto, al aproximar el autopar (x, λ) donde λ está cercano a cero es similar a aproximar el $\mathtt{Kernel}(A)$ y la dimensión del mismo.

Observación:

- Si $\mathtt{dim}(\mathtt{Kernel}(A)) \geq 0$ entonces existe un autovalor de la matriz A igual a cero. Si existe un autovalor igual a cero, entonces $\mathtt{dim}(\mathtt{Kernel}(A)) \geq 1$. Sin embargo, la dimensión del $\mathtt{Kernel}$ no representa la multiplicidad del autovalor cero.

DEFINICIÓN Sea $p_A(z) = \det(A - zI)$, al polinomio p se le denomina *polinomio característico* de matriz A.

Note que, si $p_A(z) = 0$, entonces $\det(A-zI) = 0$. Por la propiedad 2, se tiene que $A-zI$ es singular, lo cual implica que existe $v \in \text{Kernel}(A-zI)$ tal que $(A-zI)v = \vec{0}$ y luego, $Av = zv$. Recíprocamente, si z es autovalor de A, entonces $(A - zI)v = \vec{0}$, con $v \in \text{Kernel}(A - zI)$, lo cual implica que la matriz $A-zI$ es singular, y por esto $\det(A-zI) = 0$. Hemos demostrado que, podemos detectar los autovalores de la matriz A, calculando las raices del polinomio característico. Luego, por el teorema fundamental del algebra $p_A(z) = (\lambda_1 - z)\cdots(\lambda_n - z)$, donde $\lambda_1, \cdots, \lambda_n$ son los autovalores de la matriz A.

2.1 Calculo de Autovalores y Autovectores. En esta sección, se estudián varios métodos para calcular los autovalores y autovectores de una matriz. Primero, veremos el método de las potencias, el cual no fue utilizado en este trabajo, pero sigue siendo un método importante cuando se necesita calcular un solo autopar de la matriz. Luego, veremos el algoritmo QR, el cual nos permite calcular todo el espectro de una matriz cuadrada. Este método fue utilizado en nuestro trabajo mediante la función `eig` que se encuentra en el ambiente *Matlab*. Y por último, veremos el método *Iram*, el cual permite calcular un número determinado de autovalores y autovectores de la matriz. Este método fue utilizado en nuestro trabajo mediante la función `eigs` del ambiente *Matlab*.

2.1.1 Método de las Potencias. Este método permite calcular el autovalor λ de mayor módulo de la matriz A. Se garantiza la convergencia del método cuando existen n autovectores linealmente independientes de la matriz A.

Este método solo nos permite calcular un solo autopar (x, λ) de la matriz A. En la práctica, considerando que un autovalor puede tener multiplicidad mayor que uno, es recomendable utilizar un método que permita hallar más de un autovalor.

Algoritmo 5 Método de las Potencias

Datos de Entrada: x_0 iterado inicial distinto de cero, $Nmax$ número máximo de iteraciones, A matriz con n autovectores linealmente independientes.

Datos de Salida: x autovector de la matriz A, λ autovalor asociado a x

1: $x = \frac{x_0}{\|x_0\|}$
2: $i = 1$
3: **Mientras** $i < Nmax$ o $\|Ax - \lambda x\| > tol$ **hacer**
4: $x = Ax$
5: **Si** $\|x\| = \vec{0}$ **Parar** **Fin Si**
6: $x = \frac{x}{\|x\|}$
7: $\lambda = x^t Ax$
8: $i = i + 1$
9: **Fin Mientras**

A pesar de que la convergencia de este método es muy lenta, sigue siendo utilizado en algunos casos donde se requiera calcular el autovalor de mayor módulo para matrices dispersas, o también, matrices que este bien condicionadas.

2.1.2 Algoritmo QR. Este Algoritmo permite el cálculo de los autovalores de la matriz A, usando un método de factorización QR. Esta factorización permite encontrar dos matrices Q y R tales que $A = QR$, siendo Q una matriz ortogonal y R una matriz triangular superior. En nuestro trabajo se utilizó la función `eig` de `Matlab`, la cual utiliza el algoritmo QR para el calculo de autovalores de una matriz cuadrada.

Este algoritmo converge a una matriz triangular superior que es similar a A, ese decir, tiene los mismos autovalores de A. Para un estudio más detallado se puede consultar [**8**].

Algoritmo 6 Algoritmo QR

Datos de Entrada: Matriz A cuadrada de orden n.

Datos de Salida: $\{\lambda_i\}_{i=1}^{n}$ autovalores de la matriz A.

1: $A_1 = A$

2: $i = 1$

3: **Para** $i = 1, \cdots,$ hasta tener convergencia **hacer**

4: $\quad A_i = Q_i R_i$ Factorización QR de la matriz A_i

5: $\quad A_i = R_i Q_i$

6: **Fin Para**

2.1.3 Método Iram. Este método se encuentra basado en la factorización de *Arnoldi*, el cual es un método de proyecciones ortogonales sobre un espacio de Krylov. Este proceso consiste en hallar dos matrices V_m y H_m tales que:

$$AV_m \approx V_m H_m$$

donde V_m es una matriz cuyos m vectores columnas son ortogonales y forman una base del espacio de Krylov de dimensión $m << n$; y H_m es una matriz de Hessenberg. Este método es ampliamente utilizado cuando la matriz es de grandes dimensiones y dispersa; y además se quiere hallar solo unos cuantos autovalores en una región determinada.

En este trabajo, utilizamos *eigs*, versión de *Iram* en *Matlab*, solicitando algunos autovalores cercanos a cero. El algoritmo del método de *iram* pérmite hallar los autovalores con alguna característica buscada, en nuestro caso, se utilizó el caracter "sr"para obtener los autovalores con parte real menor que cierta tolerancia.

En el siguiente capítulo, veremos las justificaciones teóricas para el método de continuación numérica por pseudo-longitud de arco que se propone en este trabajo.

Algoritmo 7 Método Iram

Datos de Entrada: Matriz A, m número de autovalores buscados, "sr"tipo de autovalores buscados.

Datos de Salida: $\{\lambda_i\}_{i=1}^{m}$ autovalores.

1: $k = 2m$.
2: $AV_k = V_k H_k$ Factorización de Arnoldi.
3: Calcular $\{\lambda_1, \cdots, \lambda_m\}$ autovalores de H_m que satisfagan la condición "sr".
4: **Mientras** $k \leq n$ y $\{\lambda_1, \cdots, \lambda_m\}$ no satisfagan la condición "sr". **hacer**
5: $k = k + 1$.
6: $AV_k = V_k H_k$ Factorización de Arnoldi.
7: Calcular $\{\lambda_1, \cdots, \lambda_m\}$ autovalores de H_m que satisfagan la condición "sr".
8: **Fin Mientras**

3. Propuesta de Esquema Predictor-Corrector

En este capítulo, presentamos un algoritmo de continuación por pseudo-lomgitud de arco con el que se podrá calcular el vector tangente y detectar puntos singulares mediante el cálculo de autovalores y autovectores de una matriz J_E, definida a partir de la matriz Jacobiana de G. Primero, estudiaremos los cálculos necesarios para obtener el predictor.

Ahora, justificamos los cálculos necesarios para construir un predictor de segundo orden. Para ello necesitaremos encontrar los vectores h' y h'' de la curva solución h. Recordemos que $G : \mathbb{R}^n \times \mathbb{R} \longrightarrow \mathbb{R}^n$ es una función diferenciable y $u_0 = (\vec{x}_0, \alpha_0)$ es un punto de equilibrio de G.

3.1 Cálculo de vectores h' y h''. En la sección anterior vimos que, cuando la matriz G_{u_0} tiene rango n, entonces G_{u_0} tiene n columnas linealmente independientes. Luego de reordenar las columnas de la matriz, se cumplen las hipótesis del Teorema de la Función Implícita, por lo que exite una curva solución del sistema definido por G. Sea $h : (0, l) \subset \mathbb{R}^+ \longrightarrow \mathbb{R}^{n+1}$ la reparametrización por longitud de arco de la curva solución y además $s_0 \in (0, l)$ tal que $h(s_0) = u_0$. Entonces:

$$G_u h'(s) = \vec{0} \qquad \forall s \in (0, l).$$

A partir de este sistema, tenemos que el vector tangente h' está contenido en el Kernel de la matriz G_{u_0} y también, los vectores contenidos en el Kernel de G_{u_0} difieren del vector h' solamente en sentido y en magnitud.

En esta parte, omitiremos el parámetro longitud de arco, es decir, consideraremos que las funciones están evaluadas en $s_0 \in (0,1)$. Recordemos que $g_i,\ \ i \in \{1,\cdots,n\}$, son las funciones coordenadas de la función G. Consideremos la matriz:

$$J_E = \begin{pmatrix} \frac{\partial g_1}{\partial x_1} & \cdots & \frac{\partial g_1}{\partial x_n} & \frac{\partial g_1}{\partial x_{n+1}} \\ \vdots & \ddots & \vdots & \vdots \\ \frac{\partial g_n}{\partial x_1} & \cdots & \frac{\partial g_n}{\partial x_n} & \frac{\partial g_n}{\partial x_{n+1}} \\ 0 & \cdots & 0 & 0 \end{pmatrix}$$

PROPOSICIÓN 3.1.1. *El Kernel de la matriz* G_{u_0} *es igual al Kernel de la matriz* J_E*, es decir,* $\mathrm{Kernel}(G_{u_0}) = \mathrm{Kernel}(J_E)$

DEMOSTRACIÓN. Sea $v \in \mathbb{R}^{n+1}$, entonces se cumplen las siguientes implicaciones:

$$\vec{v} \in \mathrm{Kernel}(J_E) \Longleftrightarrow J_E \cdot \vec{v} = \vec{0} \Longleftrightarrow < \nabla g_i, \vec{v} >= 0 \quad \forall i \in \{1,...,n\} \Longleftrightarrow \vec{v} \in \mathrm{Kernel}(G_{u_0})$$

❑

Sabemos que la matriz J_E debe tener rango n siempre que la matriz G_{u_0} tenga rango n. Esto implica que 0 es autovalor de J_E con multiplicidad 1. Ahora bien, el autovector v asociado al autovalor 0 está contenido en el Kernel de J_E, por la proposición 3.1.1, tenemos que $\frac{v}{\|v\|} = h'(s)$. Así, el cálculo del autovector v asociado al autovalor 0 nos permite encontrar el vector tangente a la curva solución h.

En esta parte, supongamos que se conocen las segundas derivadas de las funciones coordenadas de G. Sin embargo, cuando presentemos el algoritmo de continuación estimaremos el vector h'' mediante la resta de vectores h'. Esto es debido a que, en la práctica, es muy difícil que se tengan las segundas derivadas.

De la ecuación (1.5) tenemos:

$$G_u \cdot h' = \vec{0} \Rightarrow \begin{pmatrix} \frac{\partial g_1}{\partial x_1} & \cdots & \frac{\partial g_1}{\partial x_n} & \frac{\partial g_1}{\partial x_{n+1}} \\ \vdots & \ddots & \vdots & \vdots \\ \frac{\partial g_n}{\partial x_1} & \cdots & \frac{\partial g_n}{\partial x_n} & \frac{\partial g_n}{\partial x_{n+1}} \end{pmatrix} \begin{pmatrix} h_1' \\ \vdots \\ h_{n+1}' \end{pmatrix} = \vec{0}$$

.

Así, para la fila i tenemos:

$$< \left(\frac{\partial g_i}{\partial x_1}, \cdots, \frac{\partial g_i}{\partial x_{n+1}} \right), (h'_1, \cdots, h'_{n+1}) >= 0 \quad \Rightarrow \quad < \bigtriangledown g_i, h' >= 0$$

.

Derivando a ambos lados:

$$< (\bigtriangledown g_i)', h' > + < \bigtriangledown g_i, h'' >= 0 \tag{3.1}$$

.

Realizaremos el cálculo de $(\bigtriangledown g_i)'$

$$(\bigtriangledown g_i)' = \left(\frac{\partial g_i}{\partial x_1}', \cdots, \frac{\partial g_i}{\partial x_{n+1}}' \right)$$

.

Por regla de la cadena

$$(\bigtriangledown g_i)' = \left(\frac{\partial^2 g_i}{\partial x_i^2} h'_1 + \cdots + \frac{\partial^2 g_i}{\partial x_1 \partial x_{n+1}} h'_{n+1}, \cdots, \frac{\partial^2 g_i}{\partial x_{n+1} \partial x_1} h'_1 + \cdots + \frac{\partial^2 g_i}{\partial^2 x_{n+1}} h'_{n+1} \right)$$

,

$$(\bigtriangledown g_i)' = \left(\sum_{k=1}^{n+1} \frac{\partial^2 g_i}{\partial x_1 \partial x_k} h'_k, \cdots, \sum_{k=1}^{n+1} \frac{\partial^2 g_i}{\partial x_{n+1} \partial x_k} h'_k \right)$$

.

Sustituyendo este resultado en la ecuación (3.1), nos queda:

$$< \left(\sum_{k=1}^{n+1} \frac{\partial^2 g_i}{\partial x_1 \partial x_k} h'_k, \cdots, \sum_{k=1}^{n+1} \frac{\partial^2 g_i}{\partial x_{n+1} \partial x_k} h'_k \right), h' > + < \bigtriangledown g_i, h'' >= 0$$

,

$$\sum_{k=1}^{n+1} \frac{\partial^2 g_i}{\partial x_1 \partial x_k} h'_k h'_1 + \cdots + \sum_{k=1}^{n+1} \frac{\partial^2 g_i}{\partial x_{n+1} \partial x_k} h'_k h'_{n+1} + < \bigtriangledown g_i, h'' >= 0$$

,

$$\sum_{j=1}^{n+1} \left(\sum_{k=1}^{n+1} \frac{\partial^2 g_i}{\partial x_j \partial x_k} h'_k h'_j \right) + < \bigtriangledown g_i, h'' >= 0, \quad \text{haciendo } c_i = -\sum_{j=1}^{n+1} \left(\sum_{k=1}^{n+1} \frac{\partial^2 g_i}{\partial x_j \partial x_k} h'_k h'_j \right)$$

.

Para cada fila nos queda:

$$\begin{cases} < \bigtriangledown g_1, h'' >= c_1 \\ \vdots \\ < \bigtriangledown g_n, h'' >= c_n \end{cases}$$

.

Así, tenemos el sistema:

$$G_{u_0} h''(s_0) = \vec{c} \quad donde \quad \vec{c} = \begin{bmatrix} c_1 \\ \vdots \\ c_n \end{bmatrix}$$

.

Tenemos $n+1$ incógnitas y n ecuaciones. Usando que el vector tangente h' y el vector h'' deben ser ortogonales, tenemos que $< h'(s_0), h''(s_0) >= 0$. Finalmente, obtenemos el sistema:

$$\begin{cases} G_{u_0} h''(s_0) = \vec{c} \\ < h'(s_0), h''(s_0) >= 0 \end{cases} \tag{3.2}$$

.

El sistema (3.2) tiene $n+1$ ecuaciones y $n+1$ incógnitas. En la sección 3.4 veremos que si a la matriz G_{u_0} se le agrega el vector h' como una fila, entonces la matriz resultante tiene rango completo. Esto implica que el sistema (3.2) tiene solución única.

3.2 Definición del Predictor. Hemos encontrado los vectores $h'(s_0)$ y $h''(s_0)$, ahora calcularemos un vector predictor usando estos vectores. Haciendo el desarrollo de Taylor de grado 2 de la curva h centrado en el punto s_0, tenemos:

$$h(s) \approx h(s_0) + \sum_{k=1}^{2} \frac{h^{(k)}(s_0)(s-s_0)^k}{k!} = h(s_0) + h'(s_0)(s-s_0) + \frac{h''(s_0)(s-s_0)^2}{2}$$

Ahora bien, se desea obtener el predictor a una distancia $\lambda > 0$ del punto s_0, por lo que evaluaremos la expresión anterior en el punto $s_0 + \lambda$. Denotando por p al predictor:

$$p = h(s_0) + h'(s_0)(s_0 + \lambda - s_0) + \frac{h''(s_0)(s_0 + \lambda - s_0)^2}{2}$$

Y así, definimos el predictor:

$$p = h(s_0) + \lambda h'(s_0) + \frac{\lambda^2 h''(s_0)}{2} \tag{3.3}$$

Este predictor de segundo orden ha sido utilizado en [**3**], ya que permite reducir el número de iteraciones del método de Newton en el paso corrector. También, en [**7**], proponen construir un predictor de mayor orden, dependiendo de cuantas soluciones se hayan calculado del problema.

3.3 Detección de Puntos de Interés. Ahora, describimos la manera en la que detectaremos algunos puntos de interés sobre la curva solución. Con este método se pueden detectar tres tipos de puntos: los *Simple Bifurcation Points* (*SBP*), los *Simple Turning Points* (*STP*) y los *Hoft Points* (*HP*).

Primero, veremos que podemos detectar los puntos singulares *SBP* y *STP* sobre la curva solución detectando los autovalores cercanos a cero de la matriz J_E. La manera en la que

diferenciaremos entre ellos será mediante los autovectores asociados a los autovalores cercanos a cero.

Sabemos que la matriz J_E tiene por lo menos un autovalor igual a cero, esto implica que su polinomio característico debe ser de la forma:

$$P_c(J_E) = \det(J_E - \lambda I) = \lambda P(\lambda)$$

Siendo P un polinomio de grado n.

PROPOSICIÓN 3.3.1. *Sea u_0 un punto de equilibrio del sistema de ecuaciones G, entonces:*

$$\frac{\det(J_E - \lambda I)}{\lambda} = \det(G_x - \lambda I)$$

DEMOSTRACIÓN. Tenemos que:

$$G_x - \lambda I = \begin{pmatrix} \frac{\partial g_1}{\partial x_1} - \lambda & \cdots & \frac{\partial g_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_n}{\partial x_1} & \cdots & \frac{\partial g_n}{\partial x_n} - \lambda \end{pmatrix}$$

Luego, usando la regla de Laplace para el cálculo del determinante:

$$\det(G_x - \lambda I) = (\frac{\partial g_1}{\partial x_1} - \lambda)A_{11} + \sum_{k=2}^{n} \frac{\partial g_1}{\partial x_k} A_{1k} \tag{3.4}$$

Donde A_{1k} es el adjunto del elemento de la fila 1 y la columna k, es decir, el determinante de la matriz resultante de eliminar la fila 1 y la columna k multiplicado por $(-1)^{1+k}$.

Por otra parte, tenemos que:

$$J_E - \lambda I = \begin{pmatrix} \frac{\partial g_1}{\partial x_1} - \lambda & \cdots & \frac{\partial g_1}{\partial x_n} & \frac{\partial g_1}{\partial x_{n+1}} \\ \vdots & \ddots & \vdots & \vdots \\ \frac{\partial g_n}{\partial x_1} & \cdots & \frac{\partial g_n}{\partial x_n} - \lambda & \frac{\partial g_n}{\partial x_{n+1}} \\ 0 & \cdots & 0 & -\lambda \end{pmatrix}$$

Luego:

$$\frac{\det(J_E - \lambda I)}{\lambda} = \frac{(\frac{\partial g_1}{\partial x_1} - \lambda)\dot{A}_{11} + \sum_{k=2}^{n} \frac{\partial g_1}{\partial x_k}\dot{A}_{1k}}{\lambda} \tag{3.5}$$

Siendo $\dot{A}_{1k}$ el adjunto correspondiente al elemento de la fila 1 y columna k de la matriz $J_E - \lambda I$. Luego, usando la última fila de cada adjunto $\dot{A}_{1k}$ y usando nuevamente la regla de Laplace:

$$\dot{A}_{1k} = -\lambda A_{ik} \tag{3.6}$$

Así, sustituyendo $\dot{A}_{1k}$ en la ecuación (3.5), y por la ecuación (3.4) tenemos:

$$\frac{\det(J_E - \lambda I)}{\lambda} = (\frac{\partial g_1}{\partial x_1} - \lambda)A_{11} + \sum_{k=2}^{n} \frac{\partial g_1}{\partial x_k}A_{1k} = \det(G_x - \lambda I)$$

$$\frac{\det(J_E - \lambda I)}{\lambda} = \det(G_x - \lambda I)$$

❑

PROPOSICIÓN 3.3.2. *Sea u_0 un punto singular del sistema de ecuaciones G, entonces:*

$$\lim_{\lambda \to 0} \frac{\det(J_E - \lambda I)}{\lambda} = 0$$

DEMOSTRACIÓN. Si el punto u_0 es un punto singular, entonces se debe cumplir que:

$$\lim_{\lambda \to 0} \det(G_x - \lambda I) = 0 \tag{3.7}$$

Usando la proposición (3.3.1) y tomando el límite cuando λ tiende a cero, obtenemos el resultado:

$$\lim_{\lambda \to 0} \frac{\det(J_E - \lambda I)}{\lambda} = \lim_{\lambda \to 0} \det(G_x - \lambda I) = 0$$

❑

Obsevaciones:

- Por la proposición 3.3.1, podemos detectar los *Hoft Points* calculando los autovalores imaginarios puros de la matriz J_E.

- Por la proposición 3.3.2, tenemos que los autovalores distintos a cero de las matrices G_x y J_E son los mismos. En nuestro código de continuación, en cada iteración se buscarán los autovalores de la matriz J_E tales que $|\lambda| < tol$, siendo tol una tolerancia determinada por el usuario, para así poder detectar los puntos singulares. Cuando se tenga más de un autovalor cercano a cero, nos estaremos acercando a un punto singular.

PROPOSICIÓN 3.3.3. *Sea u_0 un punto de equilibrio. Si u_0 es un SBP, entonces se debe cumplir que $\dim(\text{Kernel}(J_E)) = 2$.*

DEMOSTRACIÓN. Como vimos en la sección (1.2), si un punto u_0 es un *SBP*, entonces necesariamente se debe cumplir que $\text{Rg}(G_{u_0}) = n - 1$. Por el teorema de la dimensión, esto implica que $\dim(\text{Kernel}(G_{u_0})) = 2$, y por la proposición 3.1.1, tenemos que $\dim(\text{Kernel}(J_E)) = 2$.

❑

Obsevaciones:

- El Recíproco de la proposición 3.3.3 no es cierto. Consideremos la siguiente función:

$$f : \mathbb{R}^3 \longrightarrow \mathbb{R}^2, \quad \text{dada por}$$
$$f(x, y, z) = (x + y + z, x + y + z)$$

La jacobiana de f viene dada por:

$$f_u = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Note que, cualquier punto sobre el plano determinado por la ecuación $x+y+z = 0$ es un cero de la función f, y además, cualquier curva sobre este plano es una curva solución del sistema. Sin embargo, $\dim(\text{Kernel}(J_E)) = 2$ en todo punto.

- Hemos demostrado que se pueden detectar los puntos singulares del sistema G calculando los autovalores cercanos a cero de la matriz J_E. Cuando se encuentre un punto singular, este podrá ser un *STP* o un *SBP*. Para poder distinguir entre ellos, calcularemos los autovectores asociados a los autovalores cercanos a cero cerca del punto singular. Si estos vectores difieren mucho entre sí (con una cierta tolerancia), entonces sera un *SBP*, ya que estos son los 2 vectores tangentes de las dos curvas que pasan por el punto. En caso contrario, será un *STP*.

- Si un punto u_0 se encuentra cerca de un *SBP* o de un *STP*, se espera que existan por lo menos 2 autovalores J_E que satisfagan la condición $|\lambda| < tol$. Si esta condición se satisface para más de 2 autovalores de J_E, entonces se tendrá que estudiar los autovectores asociados a estos autovalores. Si el espacio generado por estos autovectores es de dimensión mayor que 2, se tendrá un un tipo de singularidad diferente que a la que se estudió en este trabajo.

- Este método se puede generalizar para mas de dos curvas que se intersectan en un punto, pero se debe tomar en cuenta que solo puede detectar aquellos puntos donde los vectores tangentes de las curvas solución sean linealmente independientes.

Recordemos que $u_0 \in \mathbb{R}^{n+1}$ es una solución regular si $G(u_0) = \vec{0}$ y $Rg(G_u) = n$.

3.4 Paso Corrector. Como vimos en el capítulo 1.3.3, en los métodos de pseudo-longitud de arco se debe definir una función F que permita avanzar a lo largo de la curva solución mediante la longitud de arco y por esto, permitiendo encontrar una solución en todos los puntos regulares de la curva. En esta parte, definimos nuestro sistema F y además, demostramos que, en efecto, no se tendrán dificultades en ninguna solución regular.

Sea u_0 una solución regular del sistema de ecuaciones (1.1), sea $v = (v_1, \cdots, v_{n+1})$ el vector tangente a la curva solución en el punto u_0 y sea p el predictor definido mediante la ecuación (3.3). Primero, definimos la función Γ:

$$\begin{aligned} &\Gamma : \mathbb{R}^{n+1} \longrightarrow \mathbb{R} \\ &\Gamma(\vec{x}, \alpha) = < v, (\vec{x}, \alpha) > - < v, p, > \end{aligned}$$

Note que, en nuestro método consideraremos el parámetro longitud de arco constante (la longitud con la que avanzamos sobre la curva es constante), por esto, el dominio de nuestra función Γ es $\mathbb{R}^{n+1}$. Ahora definimos;

$$\begin{aligned} &F : \mathbb{R}^{n+1} \longrightarrow \mathbb{R}^{n+1} \quad \text{mediante} \\ &F(\vec{x}, \alpha) = (G(\vec{x}, \alpha), \Gamma(\vec{x}, \alpha)) \end{aligned} \tag{3.8}$$

La Jacobiana de F en el punto u_0 viene dada por:

$$F_{u_0} = \begin{pmatrix} \frac{\partial g_1}{\partial x_1} & \cdots & \frac{\partial g_1}{\partial x_n} & \frac{\partial g_1}{\partial \alpha} \\ \vdots & \ddots & \vdots & \vdots \\ \frac{\partial g_n}{\partial x_1} & \cdots & \frac{\partial g_n}{\partial x_n} & \frac{\partial g_n}{\partial \alpha} \\ v_1 & \cdots & v_n & v_{n+1} \end{pmatrix}$$

PROPOSICIÓN 3.4.1. *La matriz F_{u_0} es inyectiva.*

DEMOSTRACIÓN. Sea $\vec{w} \in \mathrm{Kernel}(F_{u_0})$, entonces:

$$F_{u_0} \cdot \vec{w} = \begin{pmatrix} \frac{\partial g_1}{\partial x_1} & \cdots & \frac{\partial g_1}{\partial x_n} & \frac{\partial g_1}{\partial \alpha} \\ \vdots & \ddots & \vdots & \vdots \\ \frac{\partial g_n}{\partial x_1} & \cdots & \frac{\partial g_n}{\partial x_n} & \frac{\partial g_n}{\partial \alpha} \\ \vec{v}_1 & \cdots & \vec{v}_n & \vec{v}_{n+1} \end{pmatrix} \cdot \vec{w} = \vec{0}$$

Luego:

$$\begin{cases} < \nabla g_i, \vec{w} >= 0 \quad \forall\, i = 1, \cdots, n \\ < \vec{v}, \vec{w} >= 0 \end{cases}$$

Esto implica que $\vec{w} \in \mathsf{Kernel}(G_{u_0})$, de manera que $\vec{w}$ es un vector tangente a la curva en el punto u_0. Luego $\vec{w} = c\vec{v}$, con $c \in \mathbb{R}$. Como $< \vec{v}, \vec{w} >= 0$, entonces necesariamente $\vec{w} = \vec{0}$. Así, la matriz F_{u_0} es inyectiva.

❑

Observaciones:

- Como la matriz F_{u_0} es inyectiva, entonces no es singular. Esto implica que, siempre que u_0 sea una solución regular, se podrá resolver el sistema (3.8) mediante el método de Newton o alguno semejante.

- El método de Newton necesita de un buen iterado inicial para poder aprovechar que tiene una convergencia cuadrática. Al utilizar el predictor p como iterado inicial para la función F, se tendrá que:

$$F(p) = (G(p), \Gamma(p)) = (G(p), 0) \approx \vec{0}$$

  El error que tendrá el predictor será el error que obtiene por el desarrollo de Taylor de grado 2 de la curva solución.

4. Implementación

En este capítulo, presentamos el algoritmo de continuación con el cual se utiliza el predictor de segundo orden y la función F que definimos antes en el paso corrector. Denotaremos por $J_E(u)$ a la matriz J_E que definimos en la sección anterior, evaluada en el punto $u = (\vec{x}, \alpha) \in \mathbb{R}^{n+1}$. Las variables TP, BP y HP representan los *Simple Turning Points*, *Simple Bifurcation Points* y *Hoft Points* que se encuentran en la curva.

Algoritmo 8 Propuesta

Datos de Entrada: u_0, N, tol_1, tol_2, Δs.

Datos de Salida: $\{u_i\}_{i=1\cdots,N}$, TP, BP, HP

1: **Para** $i = 1, \cdots, N$ **hacer**
2: Calcular los autovalores de la matriz $J_E(u_{i-1})$ y sus autovectores asociados.
3: **Si** Existe un autovalor de $J_E(u_{i-1})$ que sea imaginario puro **entonces**
4: u_{i-1} está cerca de un *Hoft Point*.
5: **Fin Si**
6: **Si** $J_E(u_{i-1})$ tiene más de 1 autovalor cercano a cero **entonces**
7: Determinar si el punto singular es *STP* o un *SBP* usando tol_1.
8: **Fin Si**
9: Definir el vector tangente h'_{i-1} con el autovector correspondiente..
10: Calcular el vector h''_{i-1} resolviendo el sistema (3.2) o mediante la resta de vectores tangentes previamente calculados.
11: Definir el predictor $p_i = u_{i-1} + (\Delta s)h'_i + \frac{(\Delta s)^2 h''_i}{2}$.
12: Realizar el paso corrector: $F(u_i) = \vec{0}$ usando el predictor p_i y la tolerancia tol_2
13: **Fin Para**

Cuando se va recorriendo la curva solución es muy poco probable que se logre detectar el lugar exacto donde se encuentra un punto de interés, porque siempre existe un cierto error de aproximación. Al realizar el código definiremos como puntos de interés a aquellos puntos dentro del entorno que cumplan más fuertemente una condición específica. Por ejemplo, si tenemos varios puntos en un entorno de un *Hoft Point*, definiremos como HP al punto cuyo autovalor asociado tenga su parte imaginaria más grande. En las siguientes secciones, estudiaremos la manera de realizar los procedimientos del algoritmo anterior.

4.1 Cálculo de Autovalores y Autovectores. Se realizaron dos códigos de continuación del algoritmo 1, usando dos maneras diferentes de realizar el paso 1. La primera forma fue usando la función **eig** y la segunda forma fue usando la función **eigs**. Estas funciones vienen incorporadas en el ambiente *Matlab*.

La función **eig** tiene como datos de entrada la matriz a la cual se le quieren calcular los autovalores y autovectores (en nuestro caso, la matriz $\mathsf{J_E}$) y tiene como datos de salida dos matrices que denominaremos V y D. Las columnas de la matriz V son los autovectores de la matriz $\mathsf{J_E}$ y la matriz D tiene ceros en todas sus entradas, menos en la diagonal. El elemento $\mathsf{D_{i,i}}$ de la diagonal de D es el autovalor asociado a la columna i de la matriz V.

La función **eigs** tiene como datos de entrada una matriz cuadrada (en nuestro caso es la matriz $\mathsf{J_E}$), un número entero $\mathfrak{m}$ (el número de autovalores que se quiere obtener), y un caracter que indica los autovalores de la matriz $\mathsf{J_E}$ que se desean calcular. En nuestro caso, se definió el caracter "sr"(*Small Real*) para que se busquen los autovalores de módulo más pequeño de $\mathsf{J_E}$.

Como datos de salida, la función **eigs** tiene dos matrices $\mathsf{V_m}$ y $\mathsf{D_m}$, donde las columnas de la matriz $\mathsf{V_m}$ son los autovectores de la matriz $\mathsf{J_E}$ y la matriz $\mathsf{D_m}$ tiene ceros en todas sus entradas, menos en la diagonal. La diferencia es que estas matrices son de orden $\mathfrak{m}$ (en nuestro caso se utilizó $\mathfrak{m} = 3$).

La función *eig* cálcula todo el espectro de la matriz $\mathsf{J_E}$ utilizando el método de factorizaciones QR, se puede consultar este tema en [**1**]. La función *eigs* cálcula solo una parte del

espectro utilizando el método de Arnoldi, el cual consiste en realizar proyecciones ortogonales sobre un espacio vectorial que se denomina *Espacio de Krylov*. Este método es comúnmente utilizado para resolver sistemas de ecuaciones lineales. Para un estudio más detallado de los *Espacios de Krylov* o sobre el método de Arnoldi se puede consultar [**12**]. Al realizar la factorización de Arnoldi de la matriz J_E, se obtendrán dos matrices

$$J_E \cdot Q_m = Q_m \cdot H_m \tag{4.1}$$

donde la matriz Q_m es ortogonal y la matriz H_m es una matriz *Hessenberg*. El entero m representa el número de columnas de la Q_m y también el orden de la matriz H_m (la matriz de *Hessenber* es cuadrada).

La ventaja que presenta el método de Arnoldi es que realiza la factorización de manera tal que los autovalores buscados son los autovalores de la matriz H_m. Así, si se elige un m tal que $m << n+1$ ($n+1$ es el orden de la matriz J_E) se podrán obtener los autovalores deseados de la matriz J_E con muchas menos iteraciones.

4.2 Cálculo del Vector Tangente y Detección de Puntos Singulares. En la sección 3.3, vimos que podemos detectar los *Simple Bifurcation Points* y los *Simple Turning Points* mediante los autovalores cercanos a cero de la matriz J_E. Sabemos que cerca de un punto singular la matriz J_E tendrá mas de un autovalor igual a cero; por esto, durante la ejecución del código se monitorearán los autovalores más pequeños de J_E. En esta sección, estudiaremos los procedimientos necesarios para realizar los pasos 7,9 y 10 del algoritmo de la propuesta.

En la ejecución del código se tendrán dos tolerancias que se involucran en la detección de los puntos singulares:

- La primera tolerancia, denotada por $\mathtt{tol}_1$, permitirá detectar cuando nos estemos acercando a un punto singular. Esta tolerancia representa que tan cercanos a cero se considerarán los autovalores de J_E como puntos singulares verificando que se satisfaga la condición $|\lambda| < \mathtt{tol}_1$. Si se encuentra un solo autovalor que cumpla la condición anterior, entonces este punto no estará cerca de un punto singular; si se encuentran dos autovalores que cumplan la condición anterior, entonces el punto estará cerca de un *Simple Bifurcation Point* o un *Simple Turning Point* y si encuentran mas de dos autovalores que satisfagan la condición anterior; se deberá verificar la dimensión del espacio generado por los autovectores asociados: si la dimensión es menor o igual a 2 el punto será un *STP* o un *SBP*, y si la dimensión es mayor que 2 se tendrá un tipo de singularidad que no se estudió en este trabajo.

  De esta forma, cuando nos estemos acercando a un *SBP* o a un *STP*, estaremos definiendo un entorno sobre el punto singular y cuando se deje de cumplir la condición $|\lambda| < \mathtt{tol}_1$ significará que no estaremos dentro del entorno. Los puntos de la curva dentro de este entorno tendrán por lo menos 2 autovalores cercanos a cero, se guardará el valor del segundo autovalor más pequeño (el autovalor más pequeño será el asociado al vector tangente) y luego, definiremos como punto singular al punto asociado al segundo autovalor más pequeño.

- Si un punto u_i es un *SBP*, entonces los autovectores asociados a los autovalores cercanos a cero son linealmente independientes. En nuestro código determinaremos

si los vectores son linealmente independientes mediante el ángulo que se forma entre ellos. Para ello, al tener dos autovalores que satisfacen $|\lambda| < tol_1$ cuyos autovectores v_1 y v_2 cumplen que $(1 - < v_1, v_2 >) > 10^{-1}$, entonces el punto es un *SBP*. En caso contrario, se tendrá que el punto es un *STP*.

Note que, considerando que los vectores v_1 y v_2 están normalizados, entonces $< v_1, v_2 >$ representa el coseno del ángulo formado entre estos vectores. Si los vectores son linealmente dependientes, entonces $< v_1, v_2 >= 1$. (Se debe asegurar que los vectores se encuentren orientados de la misma manera, para evitar que el producto escalar sea negativo).

Ahora, en el siguiente algoritmo mostramos el procedimiento para los pasos (6) y (9) del algoritmo de la propuesta en la iteración $i = 1, \ldots, N$. Sean $|D_1| < \cdots < |D_j|$ los autovalores de $J_E(u_{i-1})$ que cumplen la condición $|D_j| < tol_1$ y sean $v_1, \cdots, v_j$ los autovectores asociados a los D_j.

Algoritmo 9

1: **Si** $dim(span(\{v_1, \cdots, v_n\}))$, **Parar** **Fin Si**
2: **Si** $j = 2$ **entonces**
3: $cos_1 =< v_1, h_{i-1} >$
4: **Si** $cos_1 < 0$ **entonces** $cos_1 = -cos_1$ y $v_1 = -v_1$ **Fin Si**
5: $cos_2 =< v_2, h_{i-1} >$
6: **Si** $cos_2 < 0$ **entonces** $cos_2 = -cos_2$ y $v_1 = -v_1$ **Fin Si**
7: **Si** $cos_1 > cos_2$ **entonces** $h_i = v_1$
8: **Si no** $h_i = v_2$ **Fin Si**
9: **Si** $(1 - < v_1, v_2 >) > 10^{-1}$ **entonces** $bp = 1$
10: **Si no** $tp = 1$ **Fin Si**
11: **Fin Si**
12: **Si** $j = 1$ **entonces**
13: $h_i = v_1$
14: **Si**$(< h_i, h_{i-1} >) < 0$, **entonces** $h_i = -h_i$ **Fin Sin**
15: **Fin Si**

4.3 Paso Corrector. En esta sección, se estudia la manera en la que realizaremos el paso corrector en el código de continuación. Como vimos en el capítulo anterior, la matriz jacobiana de la función $F : \mathbb{R}^{n+1} \longrightarrow \mathbb{R}^{n+1}$, definida por $F(\vec{x}, \alpha) = (G(\vec{x}, \alpha), \Gamma(\vec{x}, \alpha))$ no es singular en una solución regular u_0. Recordemos que la función Γ viene definida como $\Gamma(\vec{x}, \alpha) \longmapsto < v, \vec{x}, \alpha > - < v, p_i >$, donde p_i es el predictor obtenido en la iteración i del algoritmo de la propuesta. La matriz jacobiana de F evaluada en u_0 viene dada por:

$$F_{u_0} = \begin{pmatrix} \frac{\partial g_1}{\partial x_1} & \cdots & \frac{\partial g_1}{\partial x_n} & \frac{\partial g_1}{\partial \alpha} \\ \vdots & \ddots & \vdots & \vdots \\ \frac{\partial g_n}{\partial x_1} & \cdots & \frac{\partial g_n}{\partial x_n} & \frac{\partial g_n}{\partial \alpha} \\ v_1 & \cdots & v_n & v_{n+1} \end{pmatrix}$$

En el siguiente algoritmo, se utilizó el método de Newton sobre la función F para realizar el paso corrector del algoritmo de la propuesta. Supondremos que se está ejecutando la iteración i, donde p_i es el predictor obtenido en la iteración i y denotaremos por xs_i a la solución obtenida por el corrector. Omitiremos el parámetro α en el siguiente algoritmo, consideraremos que la última coordenada de xs_i corresponde a la coordenada del parámetro α.

Algoritmo 10 Paso Corrector de la Propuesta

Datos de Entrada: tol_1, h'_i, p_i.

Datos de Salida: xs_i

1: $xs_i = p_i$
2: $j = 1$
3: **Mientras** $j \leq Nmax$ y $\|F(xs_i)\| > tol_1$ **hacer**
4: $x = xs_i$
5: Resolver el sistema $F_x * z = F(x)$
6: $xs_i = x - z$
7: **Fin Mientras**

Observaciones:

- En el paso 3, se espera que se pueda satisfacer la condición $\|\mathtt{F}(\mathtt{xs}_\mathtt{i})\| > \mathtt{tol}_1$ antes de 20 iteraciones. Recordemos que el error que tiene el punto inicial (el predictor) es el error de realizar el desarrollo de Taylor de grado 2 y también, el error de estimar la segunda derivada mediante la diferencia de los vectores tangentes anteriores.

- En lugar de usar el método de Newton, en el paso corrector se puede utilizar cualquier método que resuelva ecuaciones no lineales que requiera de un iterado inicial.

5. Pruebas Numéricas

En este capítulo, mostramos los resultados de ejecutar los dos códigos de continuación sobre algunos ejemplos, y también realizamos unas pruebas de tiempo sobre los códigos que mostrarán en cuales casos es preferible usar alguno de los dos métodos.

Problema 1. La figura 5.1 representa un circuito eléctrico compuesto por un amplificador, dos diodos y un conjunto de nodos y resistencias. En este módelo, tomado de [**13**], se estudia como varía el voltaje de salida en el nodo x_6 para distintos voltajes de entrada α.

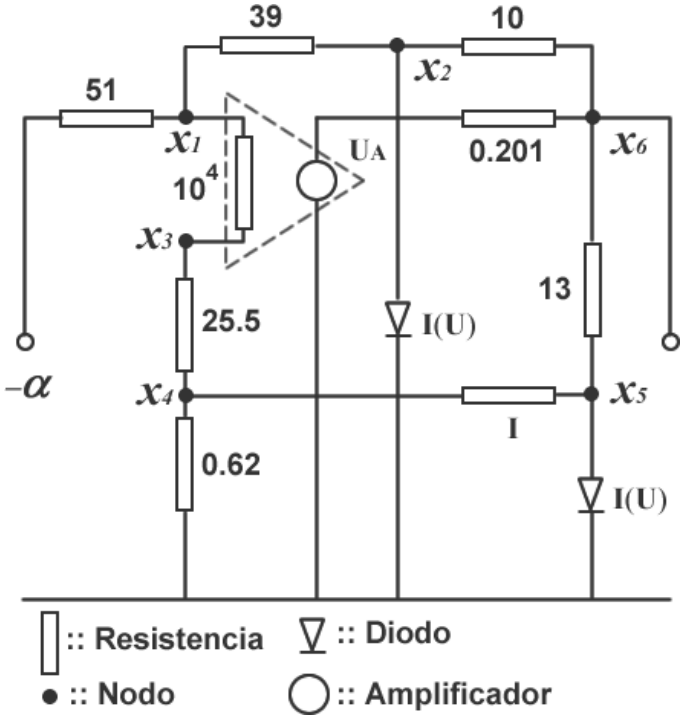

FIGURA 5.1. Diagrama del Circuito

El amplificador y los diodos son modelados por las siguientes ecuaciones $U_A(U) = 7,64\ \arctan(192\ U)$ y $I(U) = 5,6 \times 10^{-8}(e^{25U} - 1)$, respectivamente. Aplicando la primera ley de Kirchoff ($\sum_V = \sum_{IR}$) se construye el sistema de ecuaciones del circuito:

$$
\begin{aligned}
g_1(\vec{x},\alpha) &= (x_1 - x_3)\frac{1}{10000} + (x_1 - x_2)\frac{1}{39} + (x_1 - \alpha)\frac{1}{51} = 0 \\
g_2(\vec{x},\alpha) &= (x_2 - x_6)\frac{1}{10} + I(x^2) + (x_2 - x_1)\frac{1}{39} = 0 \\
g_3(\vec{x},\alpha) &= (x_3 - x_1)\frac{1}{10000} + (x_3 - x_4)\frac{1}{25,5} = 0 \\
g_4(\vec{x},\alpha) &= (x_4 - x_3)\frac{1}{25,5} + (x_4)\frac{1}{0,62} + (x_4 - x_5) - 0,62 = 0 \\
g_5(\vec{x},\alpha) &= (x_5 - x_6)\frac{1}{13} + x_5 - x_4 + I(x_5) = 0 \\
g_6(\vec{x},\alpha) &= (x_6 - x_2)\frac{1}{10} + (U_A(x_3 - x_1) - x_6)\frac{1}{0,201} + (x_6 - x_5)\frac{1}{13} = 0
\end{aligned}
$$

Usando los códigos de continuación, obtenemos dos conjuntos $\{(\vec{x}_i, \alpha_i)\}_{i=1}^{n}$ de soluciones de $G(\vec{x},\alpha) = (g_1, \cdots, g_6)$ para valores de $\alpha_i \in [0,1] \;\; \forall\, i = 1, \cdots, n$. La figura 5.2 muestra un diagrama de bifurcación realizado con los pares ordenados $(\alpha_i, [\vec{x}_i])$ con $[\vec{x}_i] = x_6^{(i)}$ obtenidos con el código que usa *eig*.

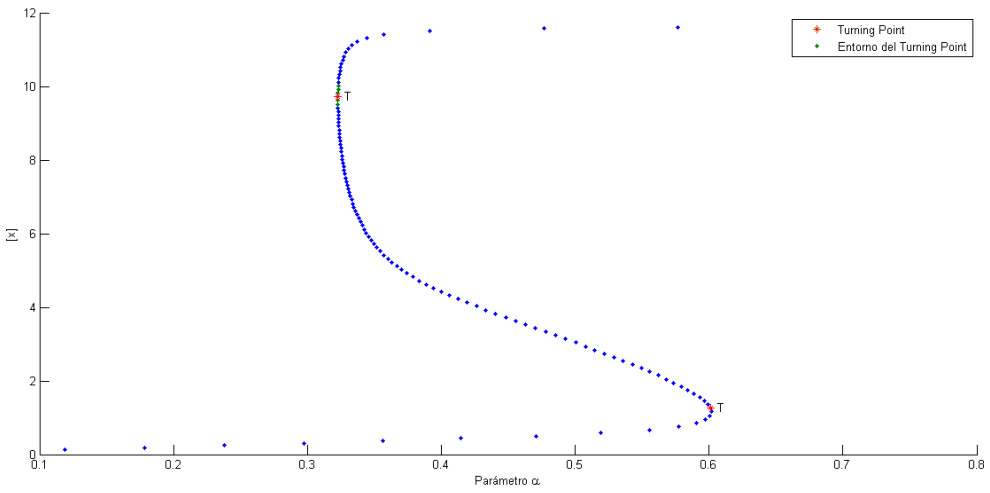

FIGURA 5.2. Diagrama de Bifurcación del Circuito (código *eig*)

En la figura 5.3 vemos un diagrama de bifurcación realizado con los pares ordenados $(\alpha_i, [\vec{x}_i])$ con $[\vec{x}_i] = x_6^{(i)}$ obtenidos con el código que usa *eigs*.

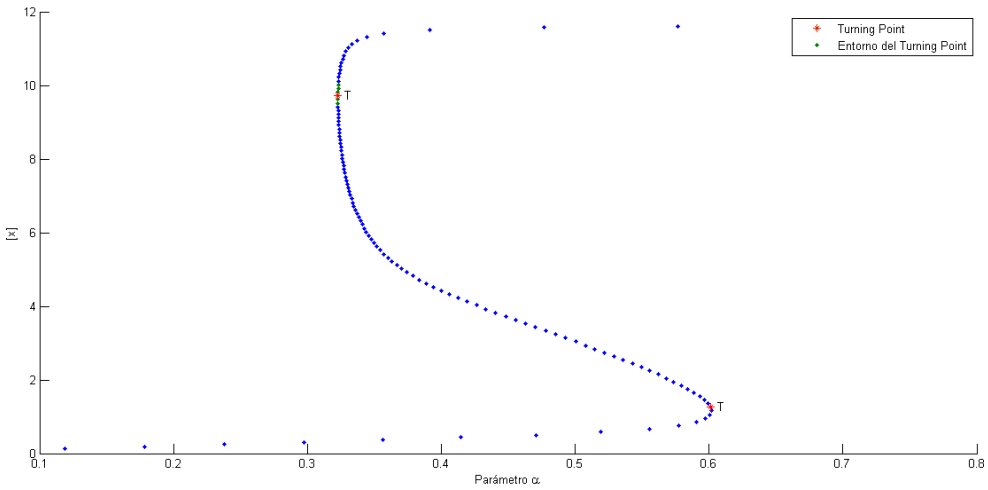

FIGURA 5.3. Diagrama de Bifurcación del Circuito (código *eigs*)

Las soluciones de los dos códigos satisfacen una tolerancia de 10^{-10}. El código que usó *eig* se demoró 19.7265 segundos en calcular las 120 soluciones de la gráfica 5.4, el código que utilizó *eigs* se demoró 19.4270 segundos en hallar las 120 soluciones. En la sección 5.1 veremos un ejemplo que muestra aproximadamente cuando se puede usar el código que usa *eig* o el que usa *eigs*.

Problema 2. El siguiente ejemplo, tomado de [**5**], es un sistema de ecuaciones llamado Brusselator, el cual modela la reacción conocida como Belusov-Zhabotinsky. Este sistema de reacción y difusión se genera a partir de la discretización de una ecuación en derivadas parciales (EDP), la cual se sabe que tiene comportamiento oscilatorio.

El sistema está formado por:

$$\begin{cases} \frac{\partial X}{\partial t} = \frac{D_x \partial^2 X}{L^2 \partial x^2} + A - (B-1)X + X^2 Y \\ \frac{\partial Y}{\partial t} = \frac{D_y \partial^2 Y}{L^2 \partial x^2} + BX - X^2 Y \end{cases} \tag{5.1}$$

donde $X, Y, A, B : [0,1] \times \mathbb{R} \longrightarrow \mathbb{R}^+$ representan las concentraciones de los reactivos, x es la variable espacial y t la variable temporal. L representa la longitud de los reactivos y D_x y D_y son constantes de difusión. En nuestro ejemplo, consideraremos los reactivos A y B constantes. En los bordes $x = 0$ y $x = 1$ se establecen las condiciones de Dirichlet:

$$\begin{cases} X(0,t) = X(1,t) = A \\ Y(0,t) = Y(1,t) = B/A \end{cases}$$

Nos interesa estudiar las soluciones de (5.1) que sean independientes del parámetro temporal. Es decir $X(x)$ y $Y(x)$, por lo que $\frac{\partial X}{\partial t} = \frac{\partial Y}{\partial t} = 0$. Ahora, hacemos una malla en el intervalo $[0, L]$ de N puntos equidistantes $\{x_1, \ldots, x_N\}$ y estimamos la derivada espacial de segundo orden de X y de Y mediante la fórmula: $\frac{\partial^2 f}{\partial x^2} = \frac{1}{h^2}(f_{i-1} + f_i + f_{i+1})$, donde $f_i = f(x_i)$ para $i = 1, \cdots, N-1$.

Ahora bien, sustituyendo estos resultados en (5.1), nos queda:

$$\begin{cases} \frac{D_x}{L^2 h^2}(X_{i-1} - 2X_i + X_{i+1}) + A - (B-1)X_i + X_i^2 Y_i = 0 \\ \frac{D_y}{L^2 h^2}(Y_{i-1} - 2Y_i + Y_{i+1}) + BX_i - X_i^2 Y_i = 0 \end{cases} \tag{5.2}$$

donde $i = 2, \ldots, N-1$ y $h = \frac{1}{N+1}$. Note que (5.2) es un sistema de ecuaciones no lineal cuyas incógnitas son $\{X_1, \ldots, X_N\}$, $\{Y_1, \ldots, Y_N\}$ y L. Asignando los valores iniciales $A = 2$, $B = 4{,}6$, $D_x = 0{,}0016$, $D_y = 0{,}08$ y $L = 0{,}06$, ejecutamos los códigos de Continuación que realizamos sobre el sistema (5.2) y de cada código se obtuvo un conjunto $\{(\vec{x}_i, \alpha_i)\}_{i=1}^n$ de soluciones.

Tomando como $[x] = x_{n+1}$, podemos ver en la figura 5.4 la gráfica de bifurcación para los puntos $([x], \alpha)$, donde $\alpha > 0{,}06$. Esta gráfica es del código donde se utilizó *eig* para el cálculo de los autovalores y autovectores.

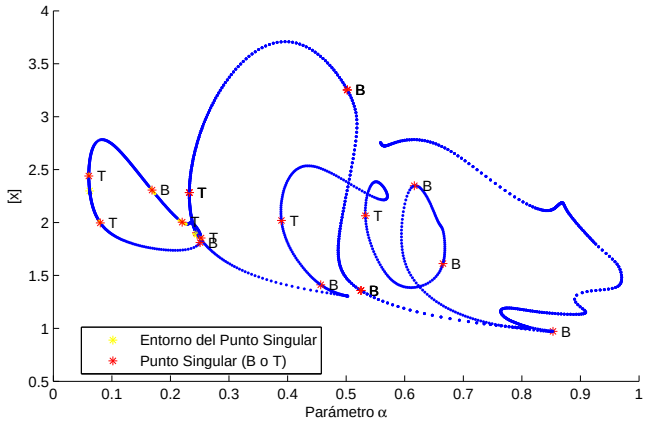

FIGURA 5.4. Diagrama de Bifurcación del Brusselator (código *eig*)

En la figura 5.5, vemos que se obtuvo el mismo resultado con el código que usa *eigs* para el cálculo de autovalores y autovectores, salvo por un punto en el extremo inferior derecho de la gráfica. Esta diferencia se debe a un error de aproximación debido a las tolerancias usadas, es decir, con cada código se obtuvieron aproximaciones de vectores tangentes lo suficientemente distintas como para que un código identificara el punto como *STP* y el otro código como *SBP*.

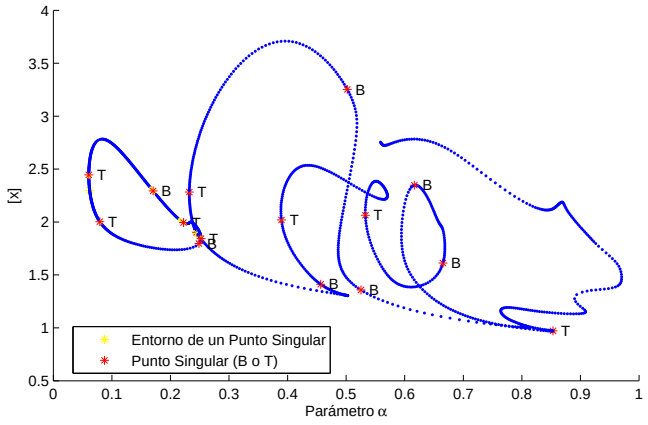

FIGURA 5.5. Diagrama de Bifurcación del Brusselator (código *eigs*)

Las soluciones de los dos códigos satisfacen una tolerancia de 10^{-8}. El código que usó *eig* se demoró 45,2740 segundos en calcular las 1500 soluciones de la figura 5.4, mientras que el código que utilizó *eigs* se demoró 158.7218 segundos en hallar las 1500 soluciones.

5.1 Pruebas de Tiempo. Como explicamos en el capítulo anterior, se realizaron dos códigos de continuación del algoritmo 1, en uno de ellos se utilizó la función *eig* y en el otro se utilizó la función *eigs*. La diferencia entre ambas funciones es que *eig* calcula todos los autovalores de una matriz, mientras que *eigs* permite calcular un número menor de autovalores que cumplan alguna propiedad particular; en nuestro caso, los autovalores más cercanos a cero.

Por esto, se esperaría que cuando el orden de la matriz que se quiera estudiar sea lo suficientemente grande, entonces *eigs* será más eficiente que *eig* en nuestro método de continuación; ya que solo necesitamos conocer los autovalores cercanos a cero de la matriz jacobiana para el cálculo de los vectores tangentes y para la detección de puntos singulares.

En la figura 5.6, podemos ver una prueba que realizamos con las funciones *eig* y *eigs*. Se crearon 20 matrices aleatorias cuadradas de rango completo usando la función *rand* de *Matlab*, cuyos órdenes variaban uniformemente desde 5 hasta 800. Luego, se midió el tiempo en que se tardaba obtener los autovalores y autovectores de las matrices usando *eigs* y usando *eig*.

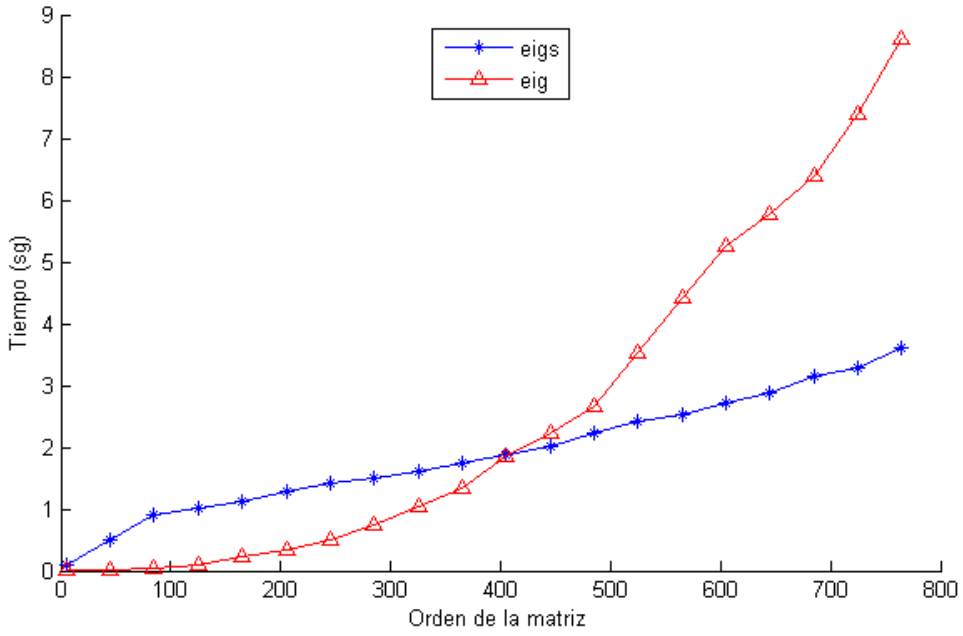

FIGURA 5.6. Comparación entre *eig* y *eigs*

Observaciones:

- Note que, cuando el orden de la matriz se acerca a 450 aproximadamente, entonces el tiempo que se toma en obtener los autovalores y autovectores de la matriz usando *eig* y usando *eigs* es casi el mismo. De ahí en adelante, es mas rápido usando *eigs*.

- Se debe tener en cuenta que el tiempo puede variar dependiendo de la forma de la matriz, y por esto, las curvas de la figura 5.6 se intersectarían en puntos distintos.

Problema 3. Consideremos la recta contenida en $\mathbb{R}^n$ que pasa por el origen, cuyo vector director es $\vec{u} \in \mathbb{R}^n$, entonces un punto $\vec{x} = (x_1, \cdots, x_n) \in \mathbb{R}^n$ que se encuentre sobre la recta deberá cumplir que:

$$\vec{x} = \alpha \cdot \vec{u} \quad \text{donde } \alpha \in \mathbb{R}^n \tag{5.3}$$

Luego:

$$(x_1, \cdots, x_n) = \alpha \cdot (u_1, \cdots, u_n) \Rightarrow (x_1 - \alpha \cdot u_1, \cdots, x_n - \alpha \cdot u_n) = \vec{0}$$

Ahora, sea $A \in M_{n \times n+1}(\mathbb{R})$ tal que, $\mathtt{dim}(\mathtt{Rg}(A)) = n$. Entonces tenemos que los vectores filas de A son linealmente independientes. Sea $i \in \{1, \cdots, n\}$, definimos la siguiente función:

$$\begin{aligned} &g_i : \mathbb{R}^{n+1} \longrightarrow \mathbb{R} \quad \text{definida por} \\ &g_i(\vec{x}, \alpha) = < A(i,:), (\vec{x}, \alpha) > \cdot(\alpha \cdot u_i - x_i) \end{aligned}$$

Así, definimos la función $G : \mathbb{R}^{n+1} \longrightarrow \mathbb{R}^n$ como $G(\vec{x}, \alpha) = (g_1(\vec{x}, \alpha), \cdots, g_n(\vec{x}, \alpha))$.

La Jacobiana de la función G viene dada por:

$$J_{ij} = \begin{cases} A_{ij} \cdot (\alpha \cdot u_i - x_i) & \text{Si} \quad i \neq j \quad \text{y} \quad j < n+1 \\ A_{ij} \cdot (\alpha \cdot u_i - x_i) - < A(i,:), (x, \alpha) > & \text{Si} \quad i = j \\ A_{ij} \cdot (\alpha \cdot u_i - x_i) + u_i < A(i,:), \alpha > & \text{Si} \quad j = n+1 \end{cases}$$

Observaciones:

- Note que $G = \vec{0}$ siempre que $(\vec{x}, \alpha) \perp A(i,:)$ y también, en todo punto sobre la recta definida por (5.3).

- Este ejemplo permite variar el orden de la matriz A, lo que permitirá realizar pruebas de tiempo sobre los métodos de continuación usando *eig* y usando *eigs*.

La figura 5.7 se realizó tomando $n = 200$, y generando la matriz A usando el comando $A = \mathtt{rand}(n, n+1)$ y $\vec{u} = \mathtt{rand}(n, 1)$ de *Matlab*. Se tomaron los puntos (x_1, x_{n+1}) para realizar la gráfica. Se utilizó el código de *eig* para esta gráfica.

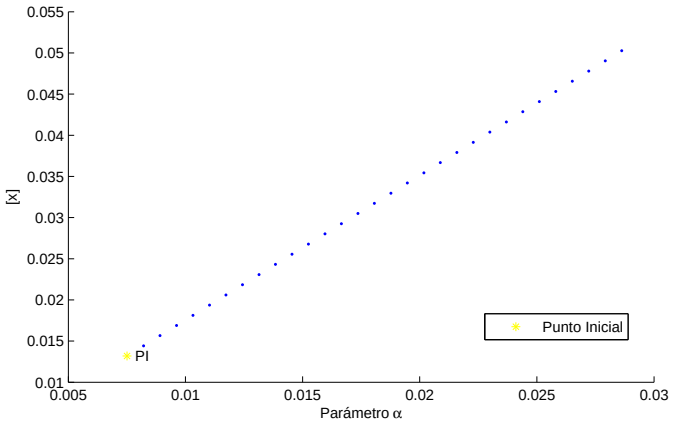

FIGURA 5.7. Diagrama de la recta

Análogamente, tenemos la gráfica obtenida con el código de *eigs* para los datos iniciales anteriores.

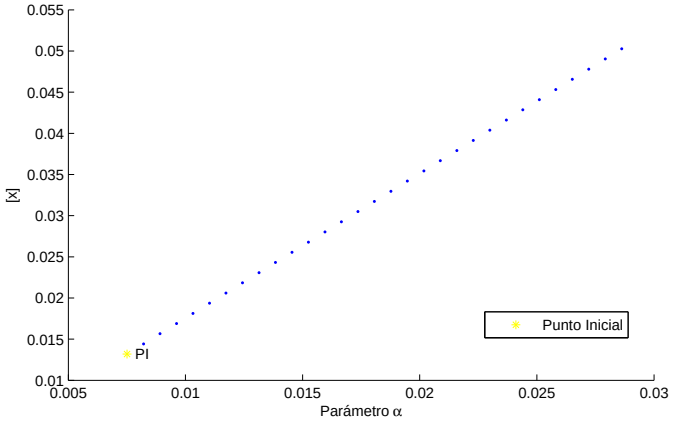

FIGURA 5.8. Diagrama de la recta

Usando *eig*, el código se demoró 19.2505 sg en obtener la solución, y usando *eigs* 20.9198 sg. Los conjuntos de soluciones obtenidas de los dos códigos satisfacen una tolerancia de 10^{-8}.

Como el parámetro n lo podemos modificar, creamos 20 matrices de n filas y $n+1$ columnas usando la función *rand* de *Matlab*, donde n varía uniformemente desde 5 hasta 2000. Luego se crearon los 20 vectores directores correspondientes al problema anterior, y medimos el tiempo que se tardan los códigos de continuación en realizar 5 iteraciones a cada uno. En la figura 5.9 gráficamos los resultados obtenidos.

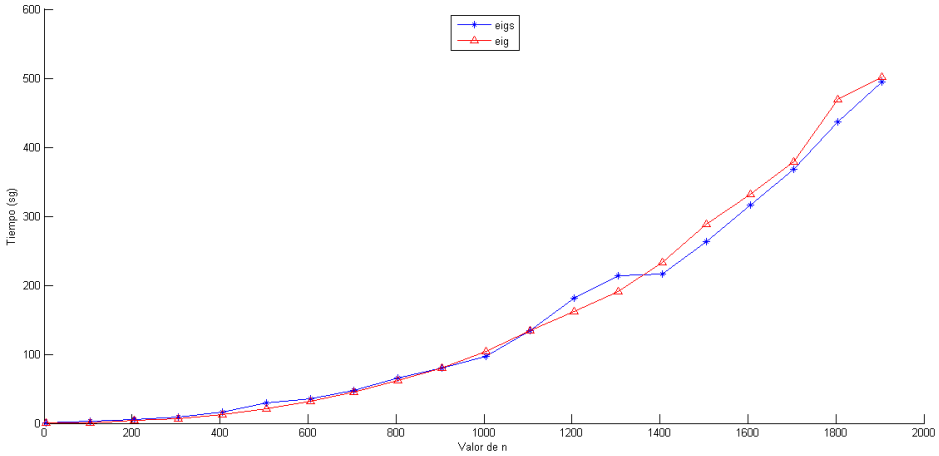

FIGURA 5.9. Prueba de Tiempo con Problema 3

Note que, cuando el valor de n es aproximadamente 1400 *eigs* empieza a ser más rápido que *eig*. Sin embargo, la diferencia de tiempo entre ellos es muy pequeña. Considerando que solo se realizaron 5 iteraciones por cada caso, mientras más soluciones se requieran de un sistema, más productivo será el código usando *eigs*.

6. Conclusiones

En este trabajo, se desarrolló un código de continuación numérica para una función $\mathsf{G} : \mathbb{R}^{\mathfrak{n}} \times \mathbb{R} \longrightarrow \mathbb{R}^{\mathfrak{n}}$ en el ambiente *Matlab*, usando dos maneras diferentes para el cálculo del espectro. En el primer código se utiliza la función *eig* y el segundo usa *eigs* para el cálculo de los autovalores y autovectores de la matriz Jacobiana de G. El código que usa *eigs* puede ser más rápido cuando el valor de $\mathfrak{n}$ es lo suficientemente grande. Se debe tomar en cuenta que la estructura de la matriz puede afectar la rapidez con que eigs calcula los autovalores y autovectores, por esto, sería conveniente realizar algunas pruebas de tiempo con una matriz que tenga la estructura del problema que se está estudiando, para determinar si es conveniente usar *eig* o *eigs*.

Otro factor importante es que los valores de las tolerancias iniciales afectan la efectividad de los códigos, por esto, se pueden variar los valores de las mismas en caso de que no se obtenga un resultado satisfactorio. Esto es debido a que, como en los métodos numéricos se trabaja con soluciones discretas, es posible que no se encuentre una solución lo suficientemente cerca de un punto singular para que sea detectado con una tolerancia específica. Esto influye para detectar los puntos singulares y también para diferenciar entre *Simple Turning Point* y *Simple Bifurcation Point*. Análogamente, mientras más grande sea el parámetro longitud de arco, menor presición tendrá el código, pero se recorrerá la curva solución más rápido. Una posible mejora que se le podría realizar al trabajo es implementar una función que varíe el parámetro longitud de arco en función de la curvatura de la curva, para tratar de avanzar una distancia óptima.

Bibliografía

[1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. D. Croz, A. Greenbaum, S. Hammarling, A. McKenney, , and D. Sorensen. *LAPACK Users' Guide*, third edition, 1999.

[2] Z. Castillo. *A New Algorithm for Continuation and Bifurcation Analysis of Large Scale Free Surface Flows*. PhD thesis, Rice University, Houston, 2004.

[3] T. F. Chan. Newton-like pseudo-arclength methods for computing simple turning points. Technical Report 233, Computer Science Department, Yale University, 1982.

[4] T. F. C. Chan. and H. B. Keller. Arc-length continuation and multi-grid techniques for nonlinear elliptic eigenvalue problems. *SIAM J. SCI. STAT. COMPUT.*, 3, 1982.

[5] A. Dhooge, W. Govaerts, Y. Kuznetsov, W. Mestrom, A. Riet, and B. Sautois. *MATCONT and CLMATCONT: Continuation Toolboxes in MATLAB*, 2006.

[6] K. I. Dickson, C. T. Kelley, I. C. F. Ipsen, and I. G. Kevrekidis. Condition estimates for pseudo-arclength continuation. *SIAM J. NUMBER. ANAL.*, 45, 2007.

[7] C. T. Kelley. Arclength continuation and bifurcation. Technical report, Center for Research in Scientific Computation, North Carolina State University, 2005.

[8] D. Kincaid and W. Cheney. *Numerical Analysis Mathematics of Scientific Computing*.

[9] P. V. Negron-Marrero. Nonlinear problems depending on a parameter: Continuation methods, limit points, simple bifurcation. Technical report, University of Puerto Rico, Department of Mathematics.

[10] A. Retali. Plataforma para el estudio y desarrollo de métodos de continuación. Trabajo Especial de Grado, Universidad Central de Venezuela, 2010.

[11] C. Robertson. Stepsize adaptation and turning point calculation for continuation methods. Master's thesis, Royal Institute of Technology, Sweden, 2004.

[12] Y. Saad. *Iterative Methods for Sparse Linear Systems*. 2 edition, 2003.

[13] R. Seydel. *Practical Bifurcation and Stability Analysis*. 2 edition.