5
2

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES
"ACATLAN"

APLICACIONES DE ANALISIS NUMERICOS EN EL ESTUDIO DE SISTEMAS DINAMICOS NO LINEALES

T E S I S

QUE PARA OBTENER EL TITULO DE

LICENCIADA EN MATEMATICAS APLICADAS Y COMPUTACION

P R E S E N T A

MINERVA CAMACHO TORRES

ASESOR:
FIS. MANUEL VALADEZ RODRIGUEZ

ACATLAN, EDO. DE MEX. 1996

TESIS CON
FALLA DE ORIGEN

TESIS

COMPLETA

VNIVERSIDAD NACIONAL AVFNOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES "ACATLAN"

DIVISION DE MATEMATICAS E INGENIERIA
PROGRAMA DE ACTUARIA Y M.A.C.

SRITA. MINERVA CAMACHO TORRES
Alumna de la carrera de M.A.C.
P r e s e n t e .

De acuerdo a su solicitud presentada con fecha 25 de enero de 1996, me complace notificarle que esta Jefatura tuvo a bien asignarle el siguiente tema de tesis: "APLICACIONES DE ANALISIS NUMERICO EN EL-ESTUDIOS DE SISTEMAS DINAMICOS NO LINEALES", el cual se desarrollará como sigue:

INTRODUCICON.
CAP. I Teoría cualitativa de sistemas dinámicos no lineales.
CAP. II Métodos numéricos para la solución de ecuaciones diferenciales ordinarias.
CAP. III Localización de equilibrios, determinación de estabilidad y trazo de ceroclinas.
CAP. IV Un algoritmo para obtener diagramas de bifurcación.
CAP. V Programación orientada a objetos y el programa BIFURCA para el análisis de sistemas dinámicos.
CONCLUSIONES.
BIBLIOGRAFIA.

Asimismo, fué designado como Asesor de Tesis: FIS. MANUEL VALADEZ RODRIGUEZ, Profesor de esta Escuela.

Ruego a usted tomar nota que en cumplimiento de lo especificado en la Ley de Profesiones, deberá presentar servicio social durante un tiempo mínimo de seis meses como requisito básico para sustentar - examen profesional, así como de la disposición de la Coordinación de la Administración Escolar en el sentido de que se imprima en lugar visible de los ejemplares de la tesis el título del trabajo - realizado. Esta comunicación deberá imprimirse en el interior de - la misma.

A T E N T A M E N T E
"POR MI RAZA HABLARA EL ESPIRITU"
Acatlán, Edo. Méx. agosto 26 de 1996.

ACT. LAURA MA. [illegible] BECERRA
Jefe del Programa de Actuaría
y M.A.C.

cg'

AGRADECIMIENTOS

A punto estuve de dejar el romanticismo de lado, pero por obra y gracia de sabe Dios qué condenada necesidad lo vertiré en este único espacio que me fué permitido. El sentimentalismo no sigue un orden, así que también me permitiré el privilegio de expresarlo tal como viene a mi mente.

Como todo ególatra ser humano, siempre tuve el íntimo y secreto deseo de que este trabajo fuera digno de no quedar en el olvido y de hecho nunca pedí demasiado, pues me bastaba con que sirviera a los estudiantes de la Licenciatura en Matemáticas Aplicadas y Computación que se imparte en la UNAM, ya como referencia bibliográfica, ya los programas desarrollados como base para construir nuevas aplicaciones.

Curiosamente, quizás porque ha pasado tanto tiempo desde que inicié este trabajo, aquel deseo de persistir se convirtió en un verdadero anhelo por verlo concluido con las imperfecciones que fuesen aunque nadie (excepto yo) se detuviera a mirarlo. A estas alturas, lo veo y me siento más feliz que una mujer recién parida a la que le basta con amar a su retoño aunque para el mundo éste pase desapercibido. En efecto, me basta con saber que el trabajo está concluido y que puse en él todo mi esfuerzo. Es esta la primera vez que me atrevo a confesar que algo me costó mucho trabajo sin importarme si éste es considerado bueno o malo ya que el éxito es a menudo azaroso; mucho esfuerzo a veces redunda en un pequeño o nulo éxito; otras veces estamos tan brillantes que cualquier cosa que hagamos será un gran éxito. Claro está que cuando se conjugan brillantez y esfuerzo, los resultados son los mejores. Además, en gran medida, el éxito se lleva dentro por cuan satisfechos estemos de lo que hacemos.

Lo difícil es elegir entre afanarte toda una vida para lograr algo "grande" arriesgándote a morir con la pena de no haber terminado tu "gran misión", o vivir a base de "pequeños" estímulos por los "pequeños logros" que obtienes con tus "pequeños y cortos esfuerzos" cuando luchas por tus "pequeñas" metas. Me parece que esa es la dicotomía de la vida o cuando menos de la mía y de la de algunas personas que andan por allí. Dicotomía porque para cualquiera de las dos opciones que se escoja, dolerá hondo dejar la otra. Quiero creer que algún día tendré la capacidad para ver grande lo que ahora es pequeño para mí porque tal vez no he

aprendido el arte de darle valor a las cosas que realmente merecen la pena, como dicen los viejos hacer.

Quisiera ahora expresar mi agradecimiento en general a todas aquellas personas que me han brindado su amistad y que de una forma u otra han puesto su huella a lo largo de mi vida. En particular, quisiera expresárselo a las siguientes personas:

A Ofelia por ser mi madre y por ganar tantas bendiciones de Dios para mí. A Agustín, mi padre, por darme tanta fortaleza y por amarnos tanto. A mi Hermano y Hermanas por compartir su vida conmigo y por recordarme ahora que estoy lejos. A Manuel Valadez por ser mi mejor amigo, por darme tanto apoyo, y porque es alguien de quien he aprendido mucho. A Joaquín, esa persona con quien compartí tan gratas experiencias, tan dulces momentos y de quien también aprendí mucho, por su colaboración con el desarrollo de un módulo de graficación en 3 dimensiones que fué incluido como parte del programa. A Humberto por su confianza y apoyo.

Quisiera también agradecer de manera muy especial a los profesores que componen el jurado de este trabajo por su comprensión y porque me siento muy honrada por el hecho de que profesores de su excelencia sean quienes aprobaron mi trabajo escrito. Muy especialmente quisiera expresarle mi gratitud al Maestro Victor Palencia por las agradables notas que envió con la aprobación del trabajo, así como mi más respetuoso afecto por el apoyo que siempre me brindó en el terreno profesional.

Minerva Camacho,
Agosto de 1996

ÍNDICE

RESUMEN

Una apasionante área de las Matemáticas es aquella que se encarga del estudio de sistemas dinámicos, llamada por muchos *Simulación Determinística*. El objetivo de dicha área es el estudio de los sistemas del mundo real cambiantes en el tiempo y cuyo comportamiento puede modelarse determinísticamente para analizar y predecir su comportamiento futuro. Ejemplos de modelos de sistemas dinámicos son los que modelan el comportamiento de las redes neuronales, la dinámica del crecimiento poblacional de dos especies que interactúan en distintas condiciones, y el movimiento del péndulo.

La importancia de estudiar sistemas dinámicos reside en que pueden realizarse predicciones sobre su comportamiento futuro y analizarse qué tan sensible es su dinámica a cambios en sus condiciones iniciales. Esto último es importante de conocerse en casi cualquier sistema, ya que en el mundo real generalmente no trabajamos con medidas y/o datos exactos; siempre habrá algún márgen de error dado, sobre todo, por la imposibilidad de considerar números con una cantidad infinita de dígitos decimales.

Existen sistemas dinámicos discretos y continuos. Los segundos son modelados mediante sistemas de ecuaciones diferenciales ordinarias y constituyen el foco de atención de este trabajo. Ahora, el análisis de un sistema dinámico de este tipo, involucra:

- La localización de su solución general.
- La localización de sus soluciones especiales: soluciones de equilibrio u órbitas periódicas.
- El estudio del comportamiento de las demás soluciones alrededor de las soluciones especiales.
- Cuándo las demás soluciones tienden a la especial a lo largo del tiempo y qué tan sensible es el comportamiento de las demás soluciones de las soluciones alrededor de puntos con características especiales.
- El tópico menos usual es el que estudia cómo cambian de ubicación y de comportamiento las soluciones especiales cuando se modifica el valor de los parámetros del sistema. A éste tipo de estudio se le conoce como Análisis de bifurcaciones y actualmente existe una robusta teoría estructurada para ello.

La *Teoría de Bifurcaciones.*

Cuando un sistema dinámico es lineal, su estudio por métodos puramente analíticos es relativamente simple. Sin embargo, es muy raro encontrar sistemas cuyo comportamiento puede modelarse linealmente, así que es ilusorio pensar que la teoría existente es general. Cualquier lector, sabe el grado de dificultad que incorpora la no linealidad en cualquier estudio. En el caso de los sistemas dinámicos la no linealidad impide generalizar una teoría para su estudio, así que, más bien se dispone de una metodología cualitativa muy dependiente de las herramientas numéricas y de los procesos de cálculo por computadora. Por tanto, en el caso no lineal (que es el más común) se utiliza una amplia gama de métodos numéricos.

La pretensión de este trabajo es instrumentar los métodos numéricos necesarios para estudiar sistemas dinámicos no lineales de dimensión arbitraria. Para cumplir nuestro objetivo es necesaria una amplia gama de métodos numéricos, pues se requieren métodos numéricos para solución de ecuaciones diferenciales ordinarias, métodos para encontrar ceros de funciones en 1 y en varias dimensiones, otros para localizar eigenvalores de una matriz, otros más para solución de sistemas de ecuaciones lineales y, finalmente, algunos algoritmos que involucran distintos tipos de métodos.

No queriendo que los programas desarrollados quedaran aislados y dependientes del programador, se construyó una aplicación de software con el nombre de BIFURCA y, aún más, con la esperanza de que sus módulos pudieran ser reutilizados, se siguió una metodología de desarrollo orientada a objetos.

En este documento se exponen todos aquellos tópicos necesarios que nos llevan a la construcción de la aplicación de software. En el primer capítulo se incluye un resúmen de la teoría cualitativa para el estudio de sistemas dinámicos. En el capítulo 2 se exponen y justifican los métodos numéricos para solución de sistemas de ecuaciones diferenciales. Los algoritmos de tales métodos no se presentan porque pueden obtenerse muy fácilmente de las ecuaciones recursivas que los definen, además de que el lector dispone del disco con los programas de todos los métodos mencionados. El capítulo 3 contiene los métodos numéricos referentes a la localización de soluciones de equilibrio y determinación de su estabilidad. Como una aportación de este trabajo, este capítulo incluye un algoritmo para obtener el espacio en que se anula cada una de las ecuaciones del sistema, llamado *ceroclina* en el caso de dimensión 2. Otro algoritmo más elaborado, también

producto de este trabajo, constituye el capítulo 4 y su objetivo es obtener diagramas de continuación de equilibrios.

Finalmente, en el capítulo 5 se presentan los conceptos de la metodología orientada a objetos para el desarrollo de software, y su ejemplificación con el modelo de objetos de BIFURCA.

Capítulo 1

TEORÍA CUALITATIVA DE SISTEMAS DINÁMICOS

INTRODUCCIÓN

En las ciencias naturales, se llama sistema dinámico a un sistema o porción del universo que está caracterizado por una colección finita de magnitudes cambiantes en el tiempo y que puede representarse mediante un sistema de ecuaciones diferenciales. Dado que todo a nuestro alrededor y dentro de nosotros mismos siempre está cambiando, el estudio de los sistemas dinámicos constituye una importante rama de las matemáticas.

Así como nada es estático, muy pocos fenómenos se comportan linealmente. Como sabemos, el comportamiento lineal se puede tipificar. De hecho existe una teoría matemática muy completa para el análisis de este tipo de sistemas dinámicos. En el caso no lineal, nos enfrentamos a un universo que no ha podido ser objeto de una clasificación total por ser tan rico en cantidad y diversidad. Por ésto, el estudio cualitativo de los sistemas dinámicos no lineales requiere de diversas técnicas muy formales de matemáticas, de métodos numéricos y un buen tanto de eurística e intuición.

En este capítulo se expondrán los conceptos básicos que requiere el estudio cualitativo de sistemas dinámicos no lineales. Se darán algunos resultados de la teoría clásica de ecuaciones diferenciales como el teorema de existencia y unicidad, dependencia de soluciones de condiciones iniciales, etc. Las últimas secciones son referentes a oscilaciones de relajación y bifurcaciones.

1.1 Sistemas Dinámicos y Forma Canónica de los Sistemas de Ecuaciones Diferenciales Ordinarias

Las magnitudes que caracterizan un sistema dinámico, sus variables de estado, pueden ser agrupadas en un vector de estado $x = (x_1, x_2, \cdots, x_n) \in \Re^n$. Ese vector de estado evoluciona en un espacio n-dimensional llamado *espacio de estados o espacio de fases* (vea figura 1.1). Supóngase que la evolución de dicho vector está determinada por una función vectorial f. Generalmente, un sistema dinámico puede modelarse mediante el sistema de ecuaciones diferenciales

$$\frac{dx}{dt} = \dot{x}(t) = f(t, x), \quad \text{donde} \quad t \in \Re, \tag{1.1}$$

$f: G \subset \Re \times \Re^n \to \Re^n$. El problema de valores iniciales consiste en encontrar la solución de (1.1) que satisface la condición inicial $x(t_0) = x_0$. La expresión $x(t; t_0, x_0)$ denota la solución de (1.1) que satisface la condición inicial $x(t_0) = x_0$.

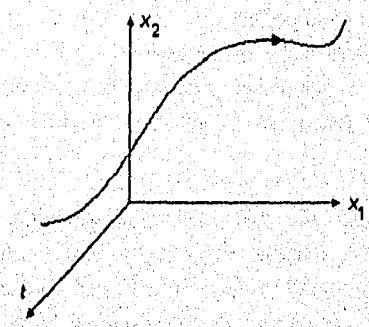


Figura 1.1. Evolución de un sistema en el tiempo

Por una solución de (1.1) entendemos un conjunto de funciones $x_1(t), x_2(t), \ldots, x_n(t)$ tal que al ser sustituidas éstas en (1.1), la igualdad se verifica. Así, podemos pensar que cada solución del sistema es una función $\phi: I \subset \Re \to \Re^n$, con regla de correspondencia $\phi(t) = (x_1(t), x_2(t), \ldots, x_n(t))$, tal que

$$\dot{\phi}(t) = f(t, \phi(t))$$

para todo $t \in \Re$, .

En general, toda ecuación diferencial de orden m de la forma

$$x^{(m)} = f(t, x), \tag{1.2}$$

puede ser expresada como un sistema de m ecuaciones de primer orden mediante un cambio adecuado de variables. Por ejemplo, si introducimos nuevas variables $x_1, x_2, \ldots, x_m$, tales que $x_1 = x$, $x_2 = \dot{x}, \ldots, x_m = x^{(m-1)}$, resulta el sistema equivalente:

$$\begin{aligned} \frac{dx_1}{dt} &= x_2, \\ \frac{dx_2}{dt} &= x_3, \\ &\vdots \\ \frac{dx_m}{dt} &= f(t, x_1, x_2, \cdots, x_m). \end{aligned} \tag{1.3}$$

Si este sistema es resuelto de alguna manera, entonces $x_1(t)$ será solución de (1.2). Recíprocamente, si x es solución de (1.2), entonces $x_1 = x$, $x_j = \dfrac{dx^{j-1}}{dt}$, $j = 2, \cdots, m$, es la solución del sistema (1.3).

Ejemplo 1.1. La ecuación del péndulo

$$\frac{d^2\theta}{dt^2} + \omega^2 \operatorname{sen}\theta = 0,$$

es equivalente al sistema de dos ecuaciones diferenciales de primer orden

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\omega^2 \operatorname{sen}(x_1); \end{aligned}$$

introduciendo las nuevas variables $x_1 = \theta$ y $x_2 = \dfrac{d\theta}{dt}$.

1.2 Existencia y Unicidad de Soluciones

Dado un problema de valores iniciales, antes de buscar la solución, primero es necesario saber si ésta existe y de ser así, que se pueda garantizar que está bien determinada por sus condiciones iniciales.

Para discutir este tema, necesitamos algunas definiciones y algunos resultados de la teoría de ecuaciones diferenciales.

Definición 1.1. Sea $D \subset \Re^n$ y considérese $f(t,x)$ definida en $G=[t_0-a,\ t_0+a]\times D$. Se dice que $f(t,x)$ satisface la condición de Lipschitz en G respecto a x si existe una constante L tal que

$$\|f(t,x_1)-f(t,x_2)\| \le L\|x_1-x_2\| \tag{1.4}$$

para todo $x_1, x_2 \in D$. L se conoce como la *constante de Lipschitz*.

En lugar de decir que $f(t,x)$ satisface la condición de Lipschitz a menudo se usa la expresión: "$f(t,x)$ es Lipschitz continua en x". Note que la Lipschitz continuidad en x implica la continuidad uniforme de f en x, sin embargo ésto no implica que también sea continua respecto a t. Por otra parte, si f es continuamente diferenciable respecto a x en un conjunto compacto, entonces es Lipschitz-continua en dicho conjunto.

Teorema 1.1 (Teorema de Existencia y Unicidad). Sea $G=[t_0-a,\ t_0+a]\times D$ donde $D=\{x\in\Re^n \mid \|x-x_0\|\le d\}$ y x_0 es un punto fijo de $\Re^n$. Si $f(t,x)$ es continua en G y es Lipschitz-continua respecto a x en G, entonces el problema de valor inicial

$$\dot{x}=f(t,x),\quad x(t_0)=x_0\ ,$$

tiene una y solo una solución definida para $|t-t_0|\le \inf(a,\frac{d}{M})$ con

$$M=\sup_G\|f\|\ .$$

Las condiciones suficientes (pero no necesarias) para la existencia y unicidad de una solución son que f sea continua en G y que sea continuamente diferenciable en D con

respecto a x. En la figura 1.2 se hace una representación gráfica del teorema en el caso en que $\Re^n=\Re$; es decir, cuando $n=1$

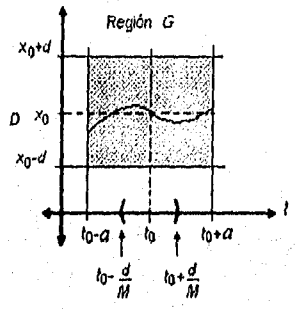


Figura 1.2. Representación de las condiciones del teorema con n=1

1.3 Sistemas Autónomos

La evolución de un sistema dinámico a partir de ciertas condiciones dadas, es distinta si estas condiciones se establecen en tiempos distintos. Es decir, dadas $x(t_0)=x_0$ y $x(t_1)=x_0$, se asume que $x(t;t_0;x_0)\neq x(t;t_1;x_0)$: las traslaciones en el tiempo de las soluciones de un sistema dinámico no tienen por qué ser también soluciones.

Ejemplo 1.2. La ecuación $\dot{x}=t$ tiene la solución general $x(t)=\frac{t^2}{2}+c$, donde el valor de c está determinado por una condición inicial. Por ejemplo si $x(t_0)=x_0$ tenemos que, $c=x_0-\frac{t_0^2}{2}$. De igual forma, para $x(t_1)=x_0$, $c=x_0-\frac{t_1^2}{2}$. De ahí que si $t_1\neq t_0$, entonces $x(t;t_1;x_0)\neq x(t;t_0;x_0)$. Lo anterior se puede apreciar mejor en la figura 1.3.

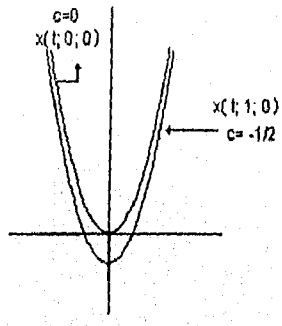


Figura 1.3. Soluciones para x_0=0, t_0=0, t_1=1.

Un caso especial de la ecuación (1.1) resulta muy importante en aplicaciones. Muchos procesos naturales tienen la propiedad de que su comportamiento no depende del tiempo en que se inició el proceso sino sólo de sus condiciones iniciales. Más formalmente, para dichos sistemas se tiene

$$x(t;t_1;x_0) = x\big(t+(t_0-t_1);t_0;x_0\big). \tag{1.5}$$

A este tipo de sistemas se les llama *sistemas autónomos* y la ecuación que los representa es de la forma

$$\frac{dx}{dt} = \dot{x}(t) = f(x), \tag{1.6}$$

donde la función f que determina el miembro derecho de la ecuación no depende explícitamente de t.

Cualquier sistema n-dimensional de la forma (1.1) es equivalente a un sistema autónomo de dimensión $n+1$. Ésto se logra introduciendo una nueva variable x_{n+1} que sustituya al tiempo en el lado derecho de la ecuación y agregando la ecuación $\frac{dx_{n+1}}{dt} = 1$.

En adelante, restringiremos la discusión al caso autónomo sin riesgo de pérdida de generalidad. La expresión (1.6) es la forma canónica de un sistema dinámico autónomo de dimensión n. Los sistemas de la forma (1.1) son llamados *no autónomos*.

1.4 Espacio de Fases y Campo Vectorial

El conjunto abierto $D \subset \Re^n$ en el cual evoluciona el vector de variables de estado x, se llama *espacio de fases*. Consideremos la ecuación (1.6)

$$\frac{dx}{dt} = \dot{x}(t) = f(x)$$

definida para $x = (x_1, x_2, \ldots, x_n) \in D \subset \Re^n$, entonces el conjunto D es el espacio de fases. Las imágenes de las soluciones constituyen curvas en este espacio llamadas *curvas integrales* o *trayectorias en el espacio de fases del sistema*. Nótese que la gráfica de las soluciones, entendiendo las soluciones como funciones de $\Re \to \Re^n$, se encuentra en el espacio $\Re \times \Re^n$ (espacio producto entre el tiempo y el espacio de fases).

Pensemos ahora en las consecuencias que el teorema de existencia y unicidad tiene sobre los sistemas autónomos.

Geométricamente, en los sistemas no autónomos el teorema implica que si en una región G un sistema cumple con las hipótesis impuestas por éste, entonces por cada punto de dicha región pasa una y solo una gráfica de solución del sistema. En los sistemas autónomos, se tienen dos implicaciones:

1) Las proyecciones en el espacio de fases de las gráficas de las diferentes soluciones son curvas que no se cortan en ningún punto. Si las curvas proyección de dos soluciones coinciden en un punto, entonces coinciden en todos sus puntos.

2) La proyección en el espacio de fases de la gráfica de una solución del sistema autónomo, o no se corta o es cerrada. Esto significa que si la proyección se corta, entonces $\varphi(t) = \varphi(t+T)$, lo cual coincide con la definición de solución periódica mencionada más adelante.

Ejemplo 1.3. Consideremos la ecuación del péndulo matemático sin rozamiento $\ddot{x} + \operatorname{sen} x = 0$, donde $-\pi \le x \le \pi$, $x \in \Re$. Haciendo el cambio a un sistema de dos ecuaciones tenemos,

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\operatorname{sen} x_1 \end{aligned} \tag{1.7}$$

Su espacio de fases es el conjunto de $\Re^2$ para el que están definidas las parejas $(x(t),\dot{x}(t))$ cuyas trayectorias se ilustran en la figura 1.4. La gráfica de las soluciones de este sistema se encuentra en $\Re^3$.

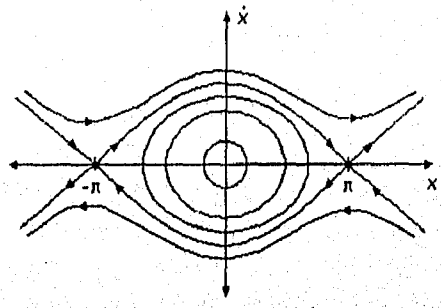


Figura 1.4. Plano Fase del Péndulo Matemático

Desde el punto de vista de las aplicaciones, es conveniente pensar en las trayectorias del sistema en el espacio de fases, porque si el sistema (1.6) representa algún sistema físico caracterizado por las variables de estado $x_1,x_2,\ldots,x_n$, entonces cada punto $(x_1,x_2,\ldots,x_n)$ del espacio de fases representa un estado del sistema y en cada tiempo t al correspondiente estado del sistema le corresponde un punto de este espacio. Si nos fijamos en el punto del espacio de fases que representa al sistema en distintos tiempos consecutivos, veremos que la evolución del sistema quedará representada por una curva, la correspondiente trayectoria del sistema.

La ecuación (1.6) determina un campo de vectores que son tangentes a las órbitas de la ecuación diferencial

Para dar una interpretación geométrica del problema de resolver el sistema autónomo (1.6), pensemos en que para cada punto del espacio de fases, f define un "vector derivada" de la solución en dicho punto. Así, f determina un campo vectorial en $\Re^n$.

Se puede decir entonces que: resolver el sistema (1.6) es lo mismo que encontrar la familia de curvas tangentes al campo vectorial determinado por f.

1.5 Soluciones de Equilibrio

Cuando se analiza el comportamiento de un sistema dinámico, es importante saber si existen soluciones especiales que puedan caracterizar, de alguna forma, el comportamiento general de dicho sistema. Consideremos dos ejemplos de tales soluciones:

a) Soluciones de equilibrio.
b) Soluciones Periódicas.

En esta sección se tratarán los aspectos más relevantes referentes a puntos de equilibrio. Más adelante nos enfocaremos a las soluciones periódicas.

Definición 1.2. Sea D el espacio de fases del problema (1.6). Un *punto de equilibrio* $x = x_{eq} \in D$ de dicho sistema, es un punto tal que $f(x_{eq}) = 0$.

Si x_{eq} es un punto de equilibrio del sistema (1.6), entonces la función $\varphi(t) = x_{eq}$ es también una solución. De hecho se considera como una órbita degenerada.

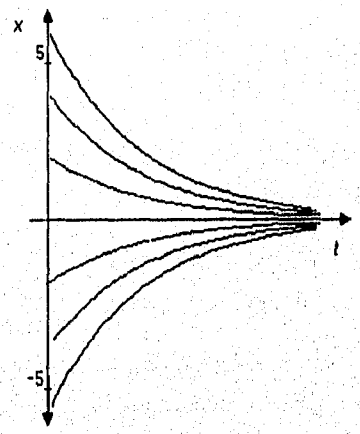
Definición 1.3. Un punto de equilibrio $x = x_{eq}$ de la ecuación (1.6), es un *atractor positivo* si existe una vecindad $\Omega_{x_{eq}} \subset \Re^n$ tal que para $x(t)$ se cumple $\lim_{t\to\infty} x(t) = x_{eq}$. Si un punto de equilibrio tiene esta propiedad cuando $t \to -\infty$, entonces es un *atractor negativo*.

Nótese que el teorema de existencia y unicidad garantiza que, aunque converjan a ésta, ninguna órbita podrá alcanzar la solución de equilibrio en un tiempo finito, de otra forma habría soluciones intersectándose. Es decir, una solución que parte de un punto que no es de equilibrio, no puede alcanzar la solución de equilibrio en un tiempo finito. De este modo, si x_0 es un punto de equilibrio y una trayectoria correspondiente a una solución se aproxima a x_0, entonces, necesariamente, $t \to \infty$. Recuérdese además que por cualquier punto x_0 del espacio de fases existe una y sólo una trayectoria del sistema (1.6) y que una trayectoria se traslapa consigo misma sólo si es cerrada (periódica).

La importancia de estas tres afirmaciones es que: si una solución parte de un punto que no es de equilibrio, entonces se mueve sobre la misma trayectoria sin importar en qué instante comience, nunca puede regresar a su punto inicial a menos que su movimiento sea periódico, nunca puede cruzar otra trayectoria y sólo puede alcanzar

un punto de equilibrio en el límite, cuando $t \to \infty$. Esto sugiere que tal solución se aproxima a un punto de equilibrio, se mueve sobre una trayectoria cerrada o tiende a una trayectoria cerrada, cuando $t \to \infty$, o bien se aleja infinitamente de dicho punto. Así, para sistemas autónomos, un estudio de los puntos de equilibrio y las soluciones periódicas tiene una importancia fundamental ya que alrededor de estos es donde se desarrolla la actividad más compleja y relevante del sistema.

Ejemplo 1.4. Para la ecuación $\dot{x} = -x, \quad t \geq 0$, las soluciones describen trayectorias como se muestra en la figura 1.5. Evidentemente, este es un atractor positivo.



$x=0$ es un punto de equilibrio; $x(t)=0$ es la solución de equilibrio. Dado que las soluciones son $x(t) = x_0 e^{-t}$, entonces $\lim_{t \to \infty} x(t) = 0$

Figura 1.5

Si en el ejemplo no se tuviera una ecuación explícita para $x(t)$, pero se supiera que 0 es el punto de equilibrio y que es un atractor positivo, podríamos esbozar un comportamiento aproximado de las soluciones.

Para encontrar los puntos de equilibrio es necesario encontrar las soluciones del sistema de ecuaciones $f(x) = 0$. Para sistemas lineales, existe una vasta teoría basada en el álgebra lineal. Pero para un sistema no lineal, obtener la solución no es una tarea fácil. Si disponemos de un equipo de cómputo, entonces podemos recurrir a métodos de solución numérica.

Expresando el sistema (1.6) en términos de sus componentes,

$$\frac{dx_j}{dt} = f_j(x_1, x_2, \ldots, x_j), \qquad j = 1,2,\ldots,n$$

tendremos que el punto de equilibrio x_{eq}, es el punto donde se intersectan las curvas $f_1(x)=0, f_2(x)=0,\ldots, f_n(x)=0$ a las que se les llama *ceroclinas*.

1.6 Soluciones Periódicas

Definición 1.4. Supóngase que $x=\phi(t)$ es una solución de la ecuación autónoma $\dot{x}=f(x)$, $x\in D\subset\Re^n$, y supóngase que $T>0$ es el menor número positivo para el cual se verifica la igualdad $\phi(t+T)=\phi(t)$ para todo $t\in\Re^n$. Entonces, se dice que $\phi(t)$ es una *solución periódica* de la ecuación con *periódo T*, o simplemente que es una *solución T-periódica*.

Es claro que $\phi(t)$ también tiene periodos $2T$, $3T$,..., kT. Dado que $\phi(t)$ vuelve a tomar los mismos valores a partir de $t=T$, es fácil ver que en el espacio de fases se tendrá una órbita cerrada. Mas aún, una órbita cerrada necesariamente corresponde a una solución periódica.

Lema 1.1. A toda solución periódica de la ecuación (3.1) le corresponde una órbita cerrada en el espacio de fases y, a toda órbita cerrada, le corresponde una solución periódica.

Prueba.

La primera parte es directa. Para probar que a toda órbita cerrada C le corresponde una solución periódica, consideremos un punto $x_0\in C$. La solución $\phi(t)$ inicia para $t=0$ en x_0. Debido a la unicidad de soluciones, C no puede contener puntos de equilibrio, es decir $\|f(x)\|\geq x_{eq}>0$ para cada $x\in C$. De ahí que $\|\dot{x}(t)\|\geq x_{eq}>0$. Entonces, después de un cierto tiempo T habremos regresado a x_0. Nuestro siguiente paso es probar que $\phi(t+T)=\phi(t)$ para todo $t\in\Re$.

Hagamos $t=nT+t_1$ con n un número entero positivo y $0<t_1<T$. De la propiedad de traslación, se sigue que si $\phi(t)$ es la solución de la ecuación (3.1) con condición inicial $\phi(t_1)=x_1$; entonces $\phi(t-nT)$ es la solución para la condición inicial $\phi(t_1+nT)=x_1$. De ahí que $\phi(t_1)=\phi(t_1+nT)$ y como $t_1\in(0,T)$, tenemos que $\phi(t)$ es T-periódica.

Una curva cerrada en el plano fase que tenga curvas no cerradas moviéndose en espiral hacia ella, desde adentro o desde afuera, cuando $t\to\infty$ se llama *ciclo límite*.

En el ejemplo 1.3 observamos que el péndulo matemático tiene soluciones periódicas aunque ninguna es un ciclo límite.

Ejemplo 1.5. En la figura 1.6 se puede observar el plano fase del sistema de Fitzhugh-Nagumo (FHN) generalizado cuyas ecuaciones son:

$$\frac{dv}{dt} = -F(v) - w - I,$$
$$\frac{dw}{dt} = b\left(w_\infty(v) - w\right),$$

donde

$$F(v) = v(v-1)(v-a)$$
$$w_\infty(v) = \frac{1}{2}\left[1 + tanh(v - v_1)/v_2\right]$$

En el ejemplo se aprecia la existencia de 2 ciclos límite. Nótese además que, análogo a como ocurre con los puntos de equilibrio, uno de estos ciclos atrae a las soluciones que están fuera y dentro de él. El otro ciclo repele a las soluciones que están fuera de él.

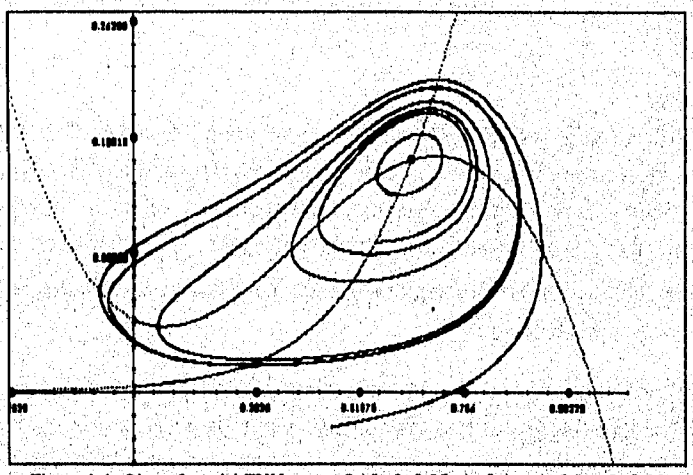


Figura 1.6 Plano fase del FHN con a=0.15, I=0.05, b=0.1, v1=0.9, v2=0.32

Para entender la dinámica global de un sistema , es muy importante saber si éste tiene soluciones periódicas, lo cual no es una tarea sencilla ni aún en el espacio $\Re^2$ para el que se cuenta con cierta teoría desarrollada como el Teorema de Poincaré-Bendixon. La localización de órbitas periódicas se tiene que llevar a cabo combinando métodos numéricos con analíticos.

1.7 Definición de Estabilidad

Mediante los sistemas dinámicos generalmente se pueden modelar sistemaas físicos que ocurren en la realidad. Cada solución de dichos sistemas representa una forma posible de comportamiento del sistema físico.

Cuando se estudian sistemas como los mencionados, es muy común pretender analizar su comportamiento para una solución particular. En tal caso, es pertinente hacer las siguientes reflexiones:

a) Teóricamente, para nuestro análisis sólo necesitaríamos establecer la condición inicial adecuada. Sin embargo, en la práctica no es posible poner exactamente al sistema físico en un estado dado, entre otras cosas porque esto implica mediciones y éstas siempre involucran algún error.

b) Teóricamente también tenemos que cuando un sistema evoluciona de acuerdo a una solución, su comportamiento debería estar regido por ésta. En la práctica, es casi imposible modelar un sistema tomando en consideración todos los factores que intervienen en él. Por ello podría haber perturbaciones que hicieran que el sistema cambiara de una solución a otra con el paso del tiempo.

Por estas razones, si queremos analizar el comportamiento del sistema a lo largo de la solución x_1, también es relevante saber cómo cambia dicho comportamiento si el sistema evoluciona a lo largo de x_2, una solución que inicialmente se encontraba cercana a x_1. A este tipo de análisis se le conoce como estudio de la estabilidad de la solución x_1.

Como se explicó anteriormente, el comportamiento más relevante de un sistema dinámico se encuentra alrededor de las soluciones de equilibrio y de las órbitas periódicas, así que nos concentraremos al análisis de estabilidad de este tipo de soluciones.

Definición 1.5. Considérese la solución $\phi(t)$ del problema (1.6). Se dice que $\phi(t)$ es *Liapunov-estable*, o *estable* solamente, si dado $\varepsilon>0$ podemos encontrar un $\delta=\delta(\varepsilon)>0$ tal que para cualquier otra solución $\varphi(t)$ que cumpla con $|\varphi(t_0)-\phi(t_0)|<\delta$, se tiene $|\varphi(t)-\phi(t)|<\varepsilon$ para $t\in[t_0,\infty)$. Si $\phi(t)$ no es estable, entonces se dice que es *inestable*.

De acuerdo a la definición, que $\phi(t)$ sea estable significa que para una vecindad alrededor del punto $\phi(t_0)$ con un radio $\varepsilon>0$ arbitrario, podemos encontrar otra vecindad

alrededor del punto $\phi(t_0)$ con radio δ de tal suerte que cualquier solución correspondiente a condiciones iniciales dadas dentro de dicha vecindad no se separará de $\phi(t)$ más de una distancia ε a medida que crezca t. Las figuras 1.7a y 1.7b ilustran la estabilidad de Liapunov.

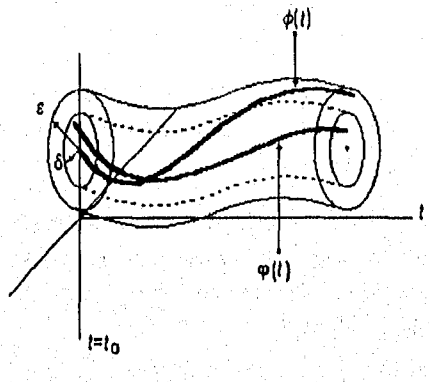


Figura 1.7a. Estabilidad de Liapunov

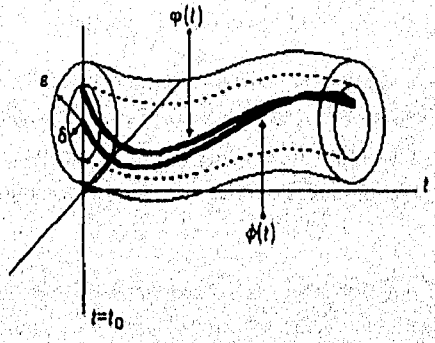


Figura 1.7b. Estabilidad Asintótica

Adicionalmente, en la figura 1.7b observamos que la distancia no sólo está acotada, sino que además tiende a cero cuando $t\to\infty$. En tal caso se dice que $\phi(t)$ es asintóticamente estable.

Definición 1.6. **Se dice que $\phi(t)$ es *asintóticamente estable* si es Liapunov-estable y además existe $\delta_1>0$ tal que si $|\varphi(t_0)-\phi(t_0)|<\delta_1$, entonces $\lim_{t\to\infty}|\varphi(t)-\phi(t)|=0$.**

Por ejemplo, se puede observar que la solución de equilibrio de (0,0) del péndulo matemático (1.7) es estable en el sentido de Liapunov. Pero las soluciones de equilibrio $(-\pi, 0)$ y $(\pi, 0)$ no lo son, ya que para cualquier vecindad alrededor de estos puntos siempre existen trayectorias que se alejan de dichos puntos.

Nótese que una solución periódica estable en el sentido de Liapunov es un caso muy especial. Tenemos que las soluciones que están cerca de de la solución periódica o son atraídas por ésta o bien son también soluciones periódicas y están sincronizadas (tienen el mismo periodo). En el caso del péndulo matemático, cuando el desplazamiento es pequeño el sistema puede ser considerado como un oscilador armónico cuyas soluciones son todas del mismo periodo. Conforme el desplazamiento es mayor, el periodo crece y ya no se podrá garantizar que soluciones inicialmente cercanas permanezcan así al transcurrir el tiempo. Es decir, las soluciones periódicas dejarán de ser estables.

En la práctica estaremos interesados en determinar la estabilidad de las soluciones de equilibrio y de las soluciones periódicas. Nótese que una solución de equilibrio asintóticamente estable es un atractor. Las soluciones periódicas asintóticamente estables se corresponden con ciclos límite. Las afirmaciones inversas no son válidas.

1.8 Estabilidad de las Soluciones de Equilibrio

Una vez encontrados los puntos de equilibrio, uno quisiera poder determinar su estabilidad. Por simplicidad supondremos que tenemos un punto de equilibrio aislado, x_{eq} (no existe ningún otro punto de equilibrio en una vecindad $\Omega_{x_{eq}}$) y que ha sido trasladado al origen del espacio de fases mediante la transformación $y = x - x_{eq}$.

Para el caso lineal se dispone de una metodología establecida para encontrar las soluciones del sistema y estudiar su comportamiento general. No así para el caso no lineal, por lo que se recurre a un análisis local alrededor del punto de equilibrio. Dicho análisis consiste en linealizar el sistema alrededor del punto de equilibrio y obtener la información necesaria a partir del sistema lineal resultante. En lo que resta de esta sección se dará una idea precisa de esta manera de proceder.

Las Soluciones del Sistema Lineal

Considérese el sistema lineal autónomo,

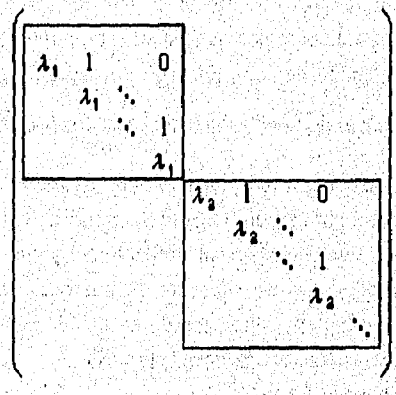
$$\frac{dx}{dt} = \mathbf{A}x\,, \tag{1.8}$$

donde $\mathbf{A} \in \Re^{n\times n}$ es una matriz constante con elementos $(\mathbf{A})_{ij}=a_{ij}$. Para resolver dicho sistema, se busca una transformación **T** no singular tal que $x=\mathbf{T}z$ haga que el sistema $\dot{z} = \mathbf{T}^{-1}\mathbf{A}\mathbf{T}z$ sea lo más simple posible. La primera intención es que $\mathbf{T}^{-1}\mathbf{A}\mathbf{T}$ sea una matriz diagonal, para ello se busca un conjunto, $\mathrm{B}=\{v_1, v_2, \cdots, v_n\}$, de vectores linealmente independientes que diagonalizen a **A**. Se sabe que si dicho conjunto está compuesto de eigenvectores, entonces $\mathbf{T}^{-1}\mathbf{A}\mathbf{T}$ es una matriz diagonal con $a_{ii}=\lambda_i$ para cada $i=1,2,...$ y donde cada λ_i es un eigenvalor de **A**, que tiene asociado al vector propio v_i. El sistema resultante

$$\dot{z} = \mathbf{T}^{-1}\mathbf{A}\mathbf{T}z \tag{1.9}$$

está desacoplado por lo que cada componente de z puede obtenerse separadamente y de forma casi directa. Las columnas de **T** son los vectores de B.

Si hay algunos valores propios repetidos que impidan obtener n eigenvectores linealmente independientes, entonces se obtienen los vectores necesarios para completar una base de Jordan. En este caso, las columnas de **T** están formadas por los vectores de dicha base y la matriz $\mathbf{T}^{-1}\mathbf{A}\mathbf{T}$ es una matriz triangular superior llamada forma normal de Jordan y que tiene la forma:



Aunque $\mathbf{T}^{-1}\mathbf{A}\mathbf{T}$ no es un sistema desacoplado, es mucho más sencillo de resolverse que el original.

Cuando $\mathbf{T}^{-1}\mathbf{AT}$ es diagonal, la solución general tiene la forma $x = \sum_{k=1}^{n} c_k v_k e^{\lambda_k t}$ donde las c_k son constantes que dependen de las condiciones iniciales y cada v_k es un eigenvector asociado al eigenvalor λ_k. En otro caso, la solución toma formas distintas dependiendo de la multiplicidad de los valores propios y de los eigenvectores linealmente independientes que se les puedan asociar. Por ejemplo, para un sistema de dimensión dos con un eigenvalor de multiplicidad 2 que únicamente tiene asociado un eigenvector, tendremos una solución general de la forma

$$x = v_1 e^{\lambda t} + v_1 t e^{\lambda t} + v_2 e^{\lambda t},$$

donde $\{v_1, v_2\}$ es la base de Jordán y λ el valor propio único.

En cualquier caso, se distingue que los valores propios son los que determinan el comportamiento principal de las soluciones del sistema.

El Sistema Lineal de Dimensión 2

Para ilustrar la manera en que los valores propios determinan el comportamien de las soluciones de un sistema dinámico, tomaremos el caso bidimensional.

Cuando $\mathbf{A} \in \Re^{2\times 2}$, las soluciones del sistema tienen sólo unos cuantos comportamientos posibles. Estos se discutirán brevemente, suponiendo que el cambio de coordenadas $y=\mathbf{T}z$ ha sido llevado a cabo. Si $\lambda_1 \neq \lambda_2$, entonces existe una base de eigenvectores linealmente independientes y $\mathbf{T}^{-1}\mathbf{AT}$ tiene la forma,

$$\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}.$$

Si sólo existe un eigenvalor y tiene una base de eigenvectores asociados, la forma para $\mathbf{T}^{-1}\mathbf{AT}$ será

$$\begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix}.$$

Cuando no se pueden obtener dos eigenvectores linealmente independientes a partir de un único eigenvalor existente, $\mathbf{T}^{-1}\mathbf{AT}$ corresponde a la forma normal de Jordan

$$\begin{pmatrix} \lambda & 1 \\ 0 & \lambda \end{pmatrix}.$$

En cada caso, la forma de la solución general para $z(t)$ es

$$z(t) = \begin{pmatrix} c_1 e^{\lambda_1 t} \\ c_2 e^{\lambda_2 t} \end{pmatrix} \qquad \text{para el primer caso,} \tag{1.10a}$$

$$z(t) = \begin{pmatrix} c_1 e^{\lambda t} \\ c_2 e^{\lambda t} \end{pmatrix} \qquad \text{para el segundo caso,} \tag{1.10b}$$

$$z(t) = \begin{pmatrix} c_1 e^{\lambda t} + c_2 e^{\lambda t} \\ c_2 e^{\lambda t} \end{pmatrix} \quad \text{para el tercer caso.} \tag{1.10c}$$

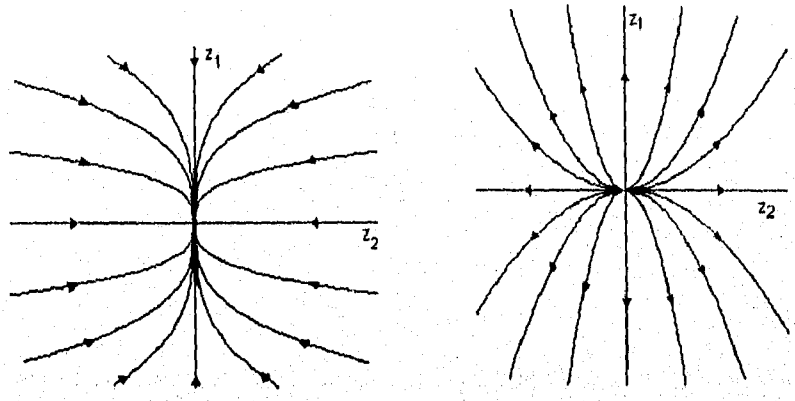
con c_1 y c_2 constantes arbitrarias. El comportamiento de las soluciones representadas por las ecuaciones (1.10), cambia de acuerdo a los valores de λ_1 y λ_2, cuyos casos se analizan a continuación.

a. El nodo

Si los valores propios son reales y de signos iguales, entonces tenemos que el punto de equilibrio es un **nodo.**

Si $\lambda_1 \neq \lambda_2$, entonces tenemos soluciones de la forma (1.10a). Eliminando t se tiene que $|z_1| = c|z_2|^{\lambda_1/\lambda_2}$. De ahí que las soluciones son parábolas similares a las que se muestran en la figura 1.8. Si $\lambda_1 < 0$ y $\lambda_2 < 0$, entonces $(0,0)$ es un atractor positivo; en otro caso es un atractor negativo.

Si $\lambda_1 = \lambda_2$ se pueden tener los casos (1.10b) o (1.10c). Puede demostrarse que cuando la soluciones tengan la forma (1.10b), las soluciones son rectas a través del origen. La figura 1.9 ilustra el comportamiento de las soluciones de la forma (1.10c). Si $\lambda < 0$, las soluciones se aproximan al punto de equilibrio cuando $t \to \infty$.



$\lambda_1=-1,\ \lambda_2=-3$ $\lambda_1=2,\ \lambda_2=1$

Figura 1.8. Ejemplo de nodos cuando $\lambda 1 \neq \lambda 2$

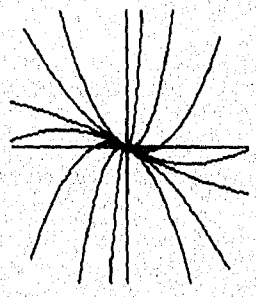


Figura 1.9. Nodo cuando $\lambda 1=\lambda 2$

b. El punto silla

Si los eigenvalores λ_1 y λ_2 son reales y tienen diferente signo, el punto de equilibrio $(0,0)$ es un punto silla. Las soluciones de $z(t)$ son de la forma (1.10a) y las órbitas del plano fase están determinadas por la ecuación $|z_1| = c|z_2|^{-|\lambda_1/\lambda_2|}$, por lo que su comportamiento es hiperbólico. De (1.10a) se puede ver que un punto silla no puede ser un atractor ya que uno de los valores propios es positivo y la función $e^{\lambda t} \to \infty$ cuando $t \to \infty$ para $\lambda > 0$. Nótese que los ejes de coordenadas corresponden a cinco soluciones diferentes: el punto de equilibrio y los cuatro semiejes. Para dos de dichas soluciones se tiene que $z(t) \to (0,0)$ cuando $t \to \infty$ y, para dos más, ésto sucede cuando $t \to -\infty$. El comportamiento descrito se muestra en la figura 1.10.

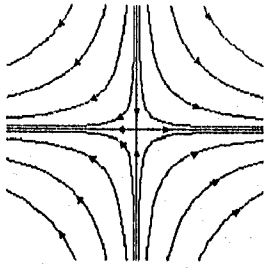


Figura 1.10. Punto silla

c. El foco

Un foco existe cuando los eigenvalores son números complejos y tanto su parte real como su parte imaginaria son distintas de 0. De hecho λ_1 es el complejo conjugado de λ_2 y viceversa. Si $\lambda_1 = u + iw$, el sistema tiene soluciones de la forma $e^{(u \pm iw)t}$. Aplicando la identidad $e^{i\theta} = \cos\theta + i\,\mathrm{sen}\theta$ y realizando algunas combinaciones lineales, obtenemos soluciones reales de la forma

$$z(t) = e^{ut}\begin{pmatrix}\cos(wt)\\ \mathrm{sen}(wt)\end{pmatrix}, \quad \text{o bien,} \quad z(t) = e^{ut}\begin{pmatrix}\mathrm{sen}(wt)\\ \cos(wt)\end{pmatrix}$$

De donde las órbitas son espirales hacia adentro del origen si $u<0$ y hacia afuera si $u>0$.

d. El centro

Cuando los eigenvalores son imaginarios puros, entonces las soluciones pueden expresarse como combinaciones de cos(wt) y sen(wt) y el punto de equilibrio corresponde a un centro ya que las órbitas en el plano fase son círculos alrededor de éste. Evidentemente, éste punto de equilibrio no es asintóticamente estable sino solamente estable.

Figura 1.11. Un centro y un foco

Linealización

La idea fundamental en este método es que si tenemos o suponemos que una o varias variables $u,v,...$ son pequeñas, entonces todas sus combinaciones de potencias y productos u^2, v^3, uv, u^4v,... resultan mucho más pequeñas y como una primera aproximación, pueden despreciarse. Por ejemplo, si tenemos el sistema

$$\begin{aligned}\dot{x}_1 &= 3x_1 - x_1^2 - x_1x_2\\ \dot{x}_2 &= -x_1 + 2x_2 + x_1x_2\end{aligned} \tag{1.11}$$

y suponemos que x_1 y x_2 son pequeñas, el sistema puede aproximarse con:

$$\begin{aligned}\dot{x}_1 &= 3x_1\\ \dot{x}_2 &= -x_1 + 2x_2\end{aligned} \tag{1.12}$$

El nuevo sistema (1.12) será una excelente aproximación en una vecindad muy pequeña del origen.

En general, queremos linealizar un sistema alrededor de un punto de equilibrio a. Esto se hace trasladando el origen al punto de equilibrio mediante la sustitución $x=x_{eq}+y$ donde y es el vector de movimiento y suponiendo que las nuevas variables son pequeñas. Aún cuando esta idea es muy útil conceptualmente, en la práctica puede resultar bastante tedioso seguir este proceso de sustituir las variables para cada punto de equilibrio, especialmente si tenemos varios de ellos. Esto puede simplificarse enormemente como se muestra a continuación.

Sea x_{eq} un punto de equilibrio. Suponiendo que $f(x)$ tiene un desarrollo en serie de Taylor alrededor de x_{eq}, y que es posible dividirlo en la parte lineal mas los términos de orden superior, linealizar significa despreciar esos términos de orden superior. En el caso de (1.6), tenemos

$$\frac{dx}{dt} = f(x_{eq}) + \left.\frac{\partial f}{\partial x}\right|_{x_{eq}} (x - x_{eq}) + \text{términos de orden superior} \tag{1.13}$$

donde $f(x_{eq})=0$. Haciendo $y=x-x_{eq}$ y despreciando los términos de orden superior obtenemos

$$\frac{dy}{dt} = \left.\frac{\partial f}{\partial y}\right|_{x_{eq}} y. \tag{1.14}$$

La matriz

$$\frac{\partial f}{\partial y} = \begin{pmatrix} \frac{\partial f_1}{\partial y_1} & \cdots & \frac{\partial f_1}{\partial y_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial y_1} & \cdots & \frac{\partial f_n}{\partial y_n} \end{pmatrix}$$

es la *matriz Jacobiana* de f. La evaluación de dicha matriz en x_{eq} produce un sistema lineal de la forma (1.8) y su análisis se efectúa de la manera en que ya se señaló. Cabe recordar que el estudio de las soluciones del sistema linealizado solamente es válido para el sistema original en una vecindad $\Omega_{x_{eq}}$ del punto de equilibrio.

Determinación de la Estabilidad de un Punto de equilibrio

En general para el sistema lineal (1.8), $\frac{dx}{dt} = Ax$, tenemos el siguiente resultado respecto a la estabilidad de sus soluciones de equilibrio.

Teorema 1.2. Considérese la ecuación (1.8) con $A \in \Re^{n\times n}$ no singular y eigenvalores $\lambda_1, \lambda_2, \ldots, \lambda_n$.

a) Si $Re\lambda_k < 0$, $k=1,\ldots,n$, entonces para cada $x(t_0)=x_0 \in \Re^n$ y constantes C y μ elegidas adecuadamente tenemos

$$\|x(t)\| \le C\|x_0\|e^{-\mu} \quad \text{y} \quad \lim_{t\to\infty} x(t) = 0.$$

b) Si $Re\lambda_k \le 0$, $k=1,\ldots,n$, donde los eigenvalores con $Re\lambda_k = 0$ son distintos entre sí, entonces $x(t)$ está acotada para $t \ge t_0$. Explícitamente

$$\|x(t)\| \le C\|x_0\|e^{-\mu}$$

con C una constante positiva.

c) Si existe un eigenvalor λ_k tal que $Re\lambda_k > 0$, entonces en cada vecindad de $x=0$ existen valores iniciales para cuyas soluciones se cumple que

$$\lim_{t\to\infty}\|x(t)\| = +\infty .$$

En el caso *a*), la solución $x=0$ es asintóticamente estable, en el caso *b*) $x=0$ es Lyapunov-estable y, en el caso *c*) es inestable.

A continuación veremos cómo se pueden aplicar éstos resultados a sistemas no lineales.

El método de linealización es útil únicamente cuando produce una buena representación del sistema no lineal. Existen resultados que especifican qué tan buena es dicha representación. A continuación se enuncian algunos de ellos referentes al sistema

$$\frac{dx}{dt} = \mathbf{A}x + g(x) = f(x), \tag{1.15}$$

tal que

$$\lim_{\|x\|\to 0} \frac{\|g(x)\|}{\|x\|} = 0.$$

Esta última condición es bastante general, se satisface por ejemplo cuando $f(x)$ es continuamente diferenciable en una vecindad de $x=0$ y se anula en este punto.

Un primer resultado es el siguiente: si $x=0$ es un atractor positivo (negativo) del sistema (1.15) linealizado, entonces $x=0$ también será un atractor positivo (negativo) del sistema no lineal (1.15).

Por otra parte, si el punto de equilibrio del sistema lineal resulta ser un punto silla, entonces éste no puede ser un atractor del sistema no lineal. Más en general, para cualquier dimensión: si la matriz **A** de (1.15) posee un eigenvalor con parte real positiva, entonces el punto de equilibrio $x=x_{eq}$ no puede ser un atractor de la ecuación no lineal (1.15); éste es inestable.

Cuando se habló de las características del punto silla en un sistema lineal, se mencionaron dos tipos de soluciones particulares: aquellas para las cuales $z(t)\to 0$ cuando $t\to\infty$ y aquellas para las cuales $z(t)\to 0$ cuando $t\to -\infty$. Estas conforman respectivamente lo que se conoce como *variedad estable* y *variedad inestable*. Para el caso de una silla de dimensión 2, son curvas, es decir, variedades unidimensionales. Dichas variedades existen cuando el punto de equilibrio es un punto hiperbólico, y son objetos que pertenecen al sistema no lineal.

La caracterización más precisa es la siguiente: si para el sistema lineal (1.8) los eigenvalores de **A** tienen parte real diferente de cero tales que n_1 de ellos tienen parte real positiva y n_2 tienen parte real negativa ($n_1+n_2=n$), entonces hay dos subespacios de $\Re^n$, E_e y E_i con las siguientes propiedades:

i) E_e y E_i tienen dimensiones n_2 y n_1 respectivamente.

ii) E_e y E_i son invariantes bajo la ecuación diferencial (1.14): toda condición inicial sobre estos subespacios da lugar a soluciones que se mantienen sobre ellos para todo tiempo.

iii) $x_0 \in E_e$ implica que $x(t;x_0) \to 0$ cuando $t \to \infty$; $x_0 \in E_e$ implica que $x(t;x_0) \to 0$ cuando $t \to -\infty$.

Por otro lado, el concepto de equilibrio hiperbólico puede generalizarse a los sistemas no lineales. Un equilibrio del sistema (1.6) es llamado hiperbólico si los eigenvalores de la matriz jacobiana de f tienen partes reales diferentes de cero. Así $x=0$ es un equilibrio hiperbólico del sistema (1.15) si esta condición se cumple para la matriz **A**. Bajo este supuesto se puede demostrar que existen dos variedades, W_e y W_i, con las siguientes propiedades:

i) W_e y W_i son invariantes bajo la ecuación diferencial (1.15).

ii) W_e y W_i son tangentes en el origen a los subespacios E_e y E_i del sistema lineal asociado (1.14).

iii) El origen atrae las trayectorias sobre la variedad W_e: $x_0 \in W_e$ implica que $x(t;x_0) \to 0$ cuando $t \to \infty$. El origen repele las trayectorias sobre W_i: $x_0 \in W_e$ implica que $x(t;x_0) \to 0$ cuando $t \to -\infty$.

Este último resultado es importante porque permite extrapolar información del sistema linealizado. Además cuando $n=2$ suele suceder que W_e y W_i separan al plano fase en dos regiones en las cuales los comportamientos de las órbitas son cualitativamente diferentes. En este caso a dichas variedades se les llama *separatrices*.

La linealización también es útil para obtener conclusiones sobre la estabilidad de un punto de equilibrio a del sistema (1.15), en resumen:

i) Si ningún eigenvalor de A tiene parte real positiva y los que tienen parte real cero son simples, entonces $x=x_{eq}$ es estable en el sentido de Liapunov.

ii) Si algún eigenvalor tiene parte real positiva entonces $x=x_{eq}$ es inestable.

iii) Si todos los eigenvalores tienen parte real negativa entonces $x=x_{eq}$ es asintóticamente estable.

Los resultados comentados en esta sección validan varios aspectos del método de linealización. Esta aproximación permite extraer información local sobre la dinámica del sistema de ecuaciones. Para obtener una idea global sobre su comportamiento, se debe realizar dicho análisis sobre cada punto de equilibrio y de hecho, podría ser más efectivo el utilizar métodos numéricos y de graficación para esbozar las soluciones de un sistema no lineal.

1.9 Estabilidad Orbital

Considérese el espacio de fases del péndulo que contiene órbitas cerradas alrededor del origen, correspondientes a las soluciones periódicas. En la figura (1.11) se han dibujado 2 órbitas con condiciones iniciales A y B. Como, a diferencia del oscilador armónico, la frecuencia del péndulo sí depende de la amplitud de oscilación, la órbita que comienza en b se mueve más lentamente que la que comienza en A. Esto implica que, para algún tiempo $t>t_0$, sucederá que los puntos que recorren sendas órbitas se encontrarán en posiciones diametralmente opuestas (C y D) con respecto al origen. Evidentemente la distancia entre ambas soluciones, $|CD|$, no depende continuamente de la cercanía de las condiciones iniciales, así que estas soluciones son inestables en términos de la definición de Liapounov.

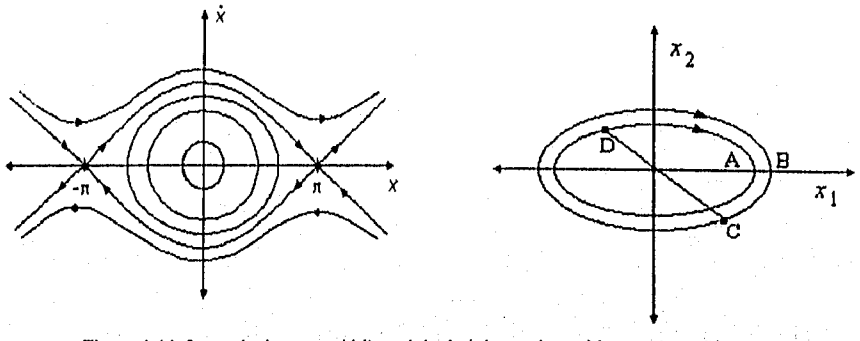


Figura 1.11 Las soluciones periódicas del péndulo son inestables en el sentido de Liapounov, debido a que soluciones inicialmente cercanas terminan separándose.

Sin embargo, no sería muy razonable calificar esta situación de inestable; una trayectoria en el espacio de fases y otra que puede verse como una pequeña perturbación de la primera, permanecen cercanas *como conjuntos de puntos.* Esta observación ha dado lugar a otra definición de estabilidad, que se aplica principalmente a soluciones periódicas:

Definición 1.7. Considérese una solución periódica $\phi(t; t_0, a)$ del problema (1.6). Se dice que ésta es *orbitalmente estable* si y sólo si para cada $\varepsilon>0$ (arbitrariamente pequeño) existen $\delta=\delta(\varepsilon)>0$ y una función $t_1(t)$ tales que

$$|x_0 - a| < \delta$$

implica que

$$|\phi(t;t_0,x_0) - \phi(t_1(t);t_0,a)| < \varepsilon, \qquad \forall\ t > t_0$$

Esta vez se ha construido un tubo alrededor de la solución $\phi(t; t_0, a)$ en el espacio de fases, no en el espacio producto entre el espacio de fases $\Re$ (espacio de fases+tiempo). La función $t_1(t)$ permite llevar a cabo la traslación necesaria para considerar tal vecindad.

Como se dijo antes, las soluciones periódicas también pueden ser atractoras. Análogamente al caso de los puntos críticos, una solución periódica es orbitalmente asintóticamente estable cuando es atractiva y además es orbitalmente estable. Ya se ha mencionado el nombre de una órbita periódica con estas características: ciclo límite.

1.10 Bifurcaciones

La ubicación y estabilidad de un punto de equilibrio de un sistema dinámico pueden cambiar al modificar los valores de sus parámetros. Por ejemplo, un equilibrio podría dejar de ser atrayente y dar lugar a un ciclo límite. Éste es un cambio cualitativo drástico en el comportamiento del sistema, y uno quisiera poder caracterizar las condiciones que deben satisfacer los parámetros para que tales cambios ocurran. Conociendo la variedad de cambios posibles y las circunstancias en que éstos suceden, se podría anticipar la evolución del sistema. De ésto se encarga la Teoría de Bifurcaciones.

El análisis de las órbitas en el espacio de fases permite entender los distintos comportamientos que puede exhibir el sistema al modificar las condiciones iniciales. El análisis de bifurcaciones también es importante, ya que el aspecto cualitativo del retrato de fases puede transformarse notoriamente al variar sus parámetros. Los valores para los cuales esto ocurre son llamados valores o puntos de bifurcación. Puede pensarse éntonces que, para cada valor de los parámetros, se tiene un retrato de fases distinto. Esta información se puede desplegar pictóricamente en una gráfica que codifica de manera sintética: el diagrama de bifurcación. El conocimiento de estos diagramas es crucial para entender el comportamiento del sistema y para poder controlar o diseñar una variante dinámica específica a partir de él.

Un ejemplo Unidimensional

La idea matemática de bifurcación está relacionada con el concepto de transición de fase en Física. Un punto de bifurcación es un punto crítico en el cual ocurren cambios drásticos. Por ejemplo, la ecuación

$$\frac{dx}{dt} = \mu x - x^2 \tag{1.16}$$

tiene a la solución trivial $x=0$ como punto de equilibrio. Otra solución de equilibrio de 1.16 es $x=\mu$. Linealizando alrededor de los puntos críticos es posible averiguar su estabilidad; se obtiene

$$x = 0 \quad \Rightarrow \quad \frac{dx}{dt} \equiv \mu x$$

$$x = \mu \quad \Rightarrow \quad \frac{dx}{dt} \equiv -\mu x$$

Así que, cuando $\mu<0$, $x=0$ es asintóticamente estable y $x=\mu$ es inestable. Cuando μ se hace positivo, las soluciones de equilibrio intercabian estabilidad: $x=0$ ahora es inestable y $x=\mu$ es la solución asintóticamente estable. Esta es una transición cualitativa bastante notoria, y $\mu=0$ es el punto en el cual ocurre, es decir, $\mu=0$ es el valor de bifurcación. Nótese que en $\mu=0$ la linealización produce el caso degenerado $dx/dt = 0$; para ese valor del parámetro el único equilibrio resulta ser inestable.

Es posible hacer un diagrama en el plano μx en el que se grafiquen los valores de x en el equilibrio, como función de μ, señalando simultáneamente su estabilidad. Una convención bastante común es unir las soluciones estables con lineas continuas y las inestables con lineas punteadas. La figura (1.12) es un diagrama de bifurcación elemental que muestra la información cualitativamente relevante sobre el sistema en función de los valores posibles del parámetro. Los puntos de equilibrio trazan dos curvas conforme μ varía: la recta horizontal $x=0$ y la recta identidad $x=\mu$. El intercambio de estabilidad se da en el punto de bifurcación $\mu=0$, en el cual existe un sólo punto crítico. Suele decirse que las soluciones de equilibrio "chocan" en el punto de bifurcación. Mediante tal diagrama de bifurcación es posible sintetizar la dinámica de (1.16) eficientemente.

En general, para obtener los puntos críticos en función de los valores de los parámetros se tienen que considerar ecuaciones del tipo

$$\mathbf{F}(\mu, \mathbf{x}) = 0$$

con $\mu \in \Re^m$, $\mathbf{x} \in \Re^n$. Se desea averiguar cuándo es que las soluciones de esta ecuación pueden bifurcarse. Sin pérdida de generalidad se puede suponer que se investiga la bifurcación de la solución trival $\mathbf{x}=0$. Así que

$$\mathbf{F}(\mu, \mathbf{x}) = 0$$

es la ecuación a estudiar. El valor del parámetro $\mu=\mu_c$ es el punto (o valor) de bifurcación si exlste una solución no trivial en toda vecindad de $(\mu_c, 0)$ en $\Re^m \times \Re^n$.

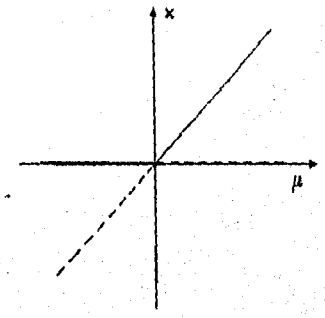


Figura 1.12. Diagrama de bifurcación de equilibrios para la ecuación (1.16). Las líneas punteadas indican puntos inestables y las continuas puntos estables

El teorema de Hopf

La teoría de bifurcaciones es abstracta y especializada. Como es de esperarse, en dimensión dos se tiene una clasificación bastante completa del tipo de bifurcaciones (transiciones) que pueden ocurrir en un sistema de ecuaciones. Sin embargo, cuando la dimensión es mayor que 2, el número y diversidad de casos crece demasiado.

Un resultado general muy útil en el análisis de modelos es el teorema de Hopf. Éste predice la aparición de una órbita periódica en las proximidades de un punto de equilibrio cuando éste cambia de estabilidad de una manera particular.

Teorema 1.3. *Teorema de Hopf.* Considérese el sistema

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}_{\mu}(\mathbf{x}) \tag{1.17}$$

donde $\mathbf{x} \in \Re^n$ y $\mu \in \Re$ es un parámetro. Supóngase que el sistema tiene un punto crítco $(\mathbf{x}_0, \mu_0)$ en el cual se satisfacen las siguientes condiciones:

h1) La matriz de derivadas **A**, dada por

$$\mathbf{A}_{ij} = \left.\frac{\partial f_i}{\partial x_j}\right|_{\mu=\mu_0},$$

tiene un único par (simple) de eigenvalores imaginarios puros.

h2) La parte real de los eigenvalores λ y λ^* (complejo conjugado) depende de μ de manera que

$$\left.\frac{d\mathrm{Re}(\lambda(\mu))}{d\mu}\right|_{\mu=\mu_0} = D \neq 0, \tag{1.18}$$

Entonces existe una única variedad de dimensión 3, llamada variedad centro, que pasa por $(\mathbf{x}_0, \mu_0)$ en $\Re^n \times \Re$, y un sistema de coordenadas para el cual la expansión en serie de Taylor de grado 3 en dicha variedad está dada por

$$\begin{aligned}\frac{dx}{dt} &= \left(Du + a\left(x^2+y^2\right)\right)x - \left(\omega + c\mu + b\left(x^2+y^2\right)\right)y \\ \frac{dy}{dt} &= \left(\omega + c\mu + b\left(x^2+y^2\right)\right)x + \left(Du + a\left(x^2+y^2\right)\right)y\end{aligned} \tag{1.19}$$

Si $a \neq 0$, entonces existe una superficie de soluciones periódicas sobre la variedad centro que concuerda hasta segundo orden con el paraboloide $\mu = -\frac{a}{D}\left(x^2+y^2\right)$. Cuando $a<0$ las soluciones periódicas son ciclos límite asintóticamente estables, mientras que si $a>0$, son repulsoras.

Discusión del caso bidimensional

En esta parte se dará una interpretación del teorema de Hopf de manera que resulte más transparente. Considérese, en primer lugar, el significado de las hipótesis: *(h1)* implica, dado que **A** evaluada en μ_0 es no singular, que existe una curva suave de equilibrios $(\mathbf{x}(\mu), \mu)$ con $\mathbf{x}(\mu_0)=\mathbf{x}_0$, y que los eigenvalores $\lambda(\mu)$ y $\lambda^*(\mu)$ de **A** (imaginarios puros en μ_0), varían suavemente con μ. Por otra parte, *(h2)* condiciona la forma en la que varían $\lambda(\mu)$ y $\lambda^*(\mu)$ con respecto a μ; el teorema requiere que estos crucen el eje imaginario con velocidad diferente de cero.

Ahora bien, ¿Cómo es que estas 2 condiciones implican la existencia de una familia de órbitas periódicas? Considérese el sistema de dimensión 2, $\mathbf{x}=(x,y)$. La clave del teorema de Hopf está en la llamada teoría de formas normales, que estudia las transformaciones de coordenadas que pueden realizarse sobre (1.17), y que

simplifican el sistema de ecuaciones al máximo. Mediante una serie adecuada de transformaciones y cambios de coordenadas, se llega a un sistema resultante con un desarrollo en serie de Taylor, con términos hasta orden 3, de la forma (1.19). En dicha expresión, los cambios de coordenadas han hecho que el valor de bifurcación sea $\mu_0=0$ y que el punto de equilibrio para μ_0 sea el origen, $x_0=0$.

La situación es considerablemente más clara si (1.19) se reescribe en coordenadas polares, con lo cual

$$\begin{aligned} \frac{dr}{dt} &= \left(D\mu + ar^2\right)r \\ \frac{d\theta}{dt} &= \omega + c\mu + br^2 \end{aligned} \tag{1.20}$$

Aquí se puede descubrir la naturaleza de la bifurcación: supóngase $D>0$ y $a>0$. La coordenada r se ha separado de θ, así que existen soluciones periódicas, que son círculos con r constante obtenidos cuando $\frac{dr}{dt}=0$ con $r\neq 0$. Entonces la amplitud r (siempre positiva), sufre una bifurcación en $\mu=0$; pasa de un único equilibrio, $r=0$ para $\mu>0$, a 2 equilibrios cuando $\mu<0$. Ésto corresponde a la aparición de una órbita periódica con amplitud $r=\left(-\frac{Du}{a}\right)^{1/2}$. Es decir, en el espacio producto $(r,\theta)\times\mu$, las soluciones se encuentran sobre la parábola $\mu=-\frac{ar^2}{D}$. Ahora es evidente la necesidad de que D sea diferente de cero ($\neq 2$); de lo contrario no puede ocurrir la bifurcación.

El teorema de Hopf afirma que las soluciones son cualitativamente las mismas cuando se agregan más términos a la serie de Taylor en (1.19). Además especifica la estabilidad del punto de equilibrio y de la órbita periódica, las cuales quedan determinadas por el coeficiente a. Ésto es fácil de ver si se linealiza (1.20) alrededor de las dos soluciones consideradas y analizando los 4 casos posibles según los signos de a y D.

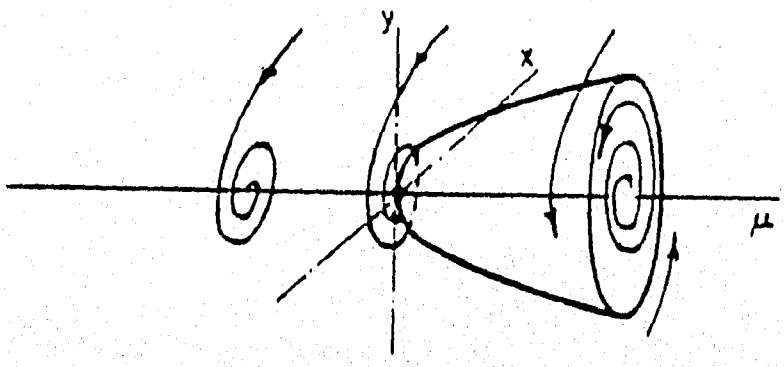


Figura 1.13. Escenario en el que ocurre la bifurcación de Hopf.

En la figura 1.13 se muestra el escenario en que ocurre la bifurcación de Hopf. En este caso, para $\mu<0$ las soluciones son espirales que se acercan al origen. Cuando $\mu>0$ las espirales se alejan de él, atraídas por el ciclo límite. Cerca del punto de bifurcación $\mu=\mu_0$, la superficie que contiene las órbitas periódicas es similar a un paraboloide.

BIBLIOGRAFÍA

[1] C. WILLIAM GEAR, *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall Inc., Engleewood Cliffs, N.J., 1971.

[2] FERDINAND VERHULST, *Nonlinear Differential Equations and Dynamical Systems*, Springer-Verlag, Berlín, 1990.

[3] HIRSCH, M.W., Y SMALE, S. *Differential Equations, Dynamical Systems and Linear Algebra*, Academic Press, New York, 1974.

[4] WILLIAM E. BOYCE, *Ecuaciones Diferenciales y problemas con valores en la frontera*, NORIEGA LIMUSA, Méx. D.F., 1991.

[5] GIORDANO WEIR, Differential Equations, a modelling approach, Addison Wesley, 1991.

[6] T.S. PARKER & L.O. CHUA, *Practical Numerical Algorithms for Chaotic Systems*, Springer-Verlag, New York Inc., 1989.

Capítulo 2

MÉTODOS NUMÉRICOS PARA LA SOLUCIÓN DE ECUACIONES DIFERENCIALES

INTRODUCCIÓN

En el campo de estudio de los sistemas de ecuaciones diferenciales ordinarias, se cuenta con métodos para encontrar soluciones analíticas cuando dichos sistemas son lineales. Sin embargo, el caso no lineal suele representar serios problemas para quien busca encontrar una expresión analítica de la solución. Es por eso que han sido creados distintos métodos numéricos para encontrar aproximaciones discretas de la solución llamados métodos de integración. En específico, nos referiremos a aquellos que construyen una ecuación iterativa llamada *ecuación en diferencias*. El carácter recursivo e iterativo de dichos métodos hace que se requieran bastantes cálculos aritméticos para encontrar las soluciones aproximadas. Así, se requiere de la computadora como una herramienta clave para el análisis de sistemas dinámicos, ya que además nos permite obtener una representación visual de dichos sistemas en un tiempo relativamente corto, siempre que se disponga del software apropiado.

2.1 Sistemas de Ecuaciones Diferenciales

Los métodos de integración que se discutirán, sirven para encontrar soluciones aproximadas de sistemas de ecuaciones diferenciales que tienen la forma

$$\frac{d\mathbf{x}}{dt} = \dot{\mathbf{x}}(t) = \mathbf{f}(t,\mathbf{x}), \tag{2.1}$$

donde $\mathbf{x} = (x_1, x_2, \ldots, x_n)^t$ es un vector columna de variables dependientes y $\mathbf{f}$ es una función vectorial de $\Re \times \Re^n \to \Re^n$.

Se buscarán soluciones que satisfagan la condición inicial $\mathbf{x}(t_0) = \mathbf{x}_0$, definidas en el intervalo $[a,b]$ que contiene a t_0. Para que dichas soluciones existan basta con que $\mathbf{f}$ sea de clase C^1 respecto a t o que satisfaga una condición de Lipschitz en $\mathbf{x}$.

Los métodos que trataremos son también útiles para resolver ecuaciones diferenciales de orden mayor. Generalmente, una ecuación diferencial de orden m puede transformarse en un sistema de m ecuaciones diferenciales de primer orden mediante un cambio adecuado de variables. Por ejemplo, como se vió en el ejemplo 1.1 del capítulo anterior, la ecuación del péndulo

$$\frac{d^2\theta}{dt^2} + \omega^2 \operatorname{sen}\theta = 0,$$

es equivalente al sistema de dos ecuaciones diferenciales de primer orden

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = -\omega^2 \operatorname{sen}(x_1);$$

introduciendo las nuevas variables $x_1 = \theta$ y $x_2 = \dfrac{d\theta}{dt}$.

Pensando que la ecuación diferencial (2.1) gobierna el comportamiento de algún sistema dinámico pensaremos que la variable independiente t es el tiempo.

En particular, se dará la teoría para resolver el problema de una variable dependiente

$$\frac{dx}{dt} = \dot{x}(t) = f(t,x), \tag{2.2}$$

con condición inicial $x(t_0) = x_0$. Donde $t_0 \le t \le b$ y $t_0 \in [a,b]$.

Asumiremos que f cumple una condición de Lipschitz en x, por lo que, para cada $t_0 \in [a,b]$, el problema (2.2) tiene una solución única definida en este intervalo.

2.2 Métodos de Integración

Como se mencionó, no siempre es posible encontrar soluciones analíticas del problema (2.2), por lo que recurriremos a los métodos numéricos para encontrar una solución aproximada. La solución aproximada que se construye no es continua, sino que consta de una sucesión discreta $\{x_i\}_{i=0}^{N}$ de aproximaciones de los valores de la solución en los puntos de otra sucesión $\{t_i\}_{i=0}^{N}$ llamados puntos de red, donde cada $t_i \in [a,b]$. A partir de la sucesión de aproximaciones, es posible obtener una aproximación continua de $x(t)$ utilizando alguna técnica de interpolación. Asociaremos a cada aproximación x_i el valor $x(t_i)$ que corresponde a la solución exacta de $x(t)$ en el punto t_i.

En gran parte de nuestra discusión asumiremos que los puntos de red están uniformemente distribuidos. Por ejemplo, la uniformidad de una sucesión de N puntos en el intervalo $[a,b]$ se tiene si hacemos $t_i = a + ih$ con $h = (b-a)/N$. La norma o distancia $h_i = |t_{i+1} - t_i|$ se llama *tamaño de paso* en la i-ésima iteración. Si los puntos de red están uniformemente distribuidos, el tamaño de paso es *fijo* y lo denotaremos por h. En otro caso, el tamaño de paso es *variable* y h se definirá como $h = max\{h_i\}$.

Los métodos de integración a que nos enfocaremos están construidos a partir de una ecuación recursiva llamada *ecuación en diferencias*. Para calcular la aproximación x_{i+1}, dicha ecuación puede depender sólo del punto anterior (t_i, x_i). Es decir, se puede expresar como $x_{i+1} = G(t_i, x_i)$, dando lugar a un *método de un paso*.

Por otra parte, el lado derecho de la ecuación en diferencias, podría estar en función de los puntos anteriores (t_i, x_i), $(t_{i-1}, x_{i-1}), \ldots, (t_{i-k}, x_{i-k})$, (para alguna constante $k>0$), caso en que tendríamos la ecuación que define a un *método multipaso*

$$x_{i+1} = G(t_i, t_{i-1}, \cdots t_{i-k}, x_i, x_{i-1}, \cdots x_{i-k}) .$$

Adicionalmente, la ecuación recursiva podría requerir también del punto (t_{i+1}, x_{i+1}) dando lugar a lo que se conoce como un *método implícito:*

$$x_{i+1} = G(t_{i+1}, t_i, \cdots t_{i-k}, x_{i+1}, x_i, \cdots x_{i-k}) .$$

Es evidente que en este último caso, es necesario utilizar algún método explícito para obtener un valor lo más cercano posible al x_{i+1} real.

2.3 Error en las Aproximaciones

El hecho de estar obteniendo una solución aproximada del problema (2.2), nos conduce a preguntarnos ¿Qué tan precisa es dicha aproximación con respecto a la solución exacta? o lo que es lo mismo, ¿Qué tan grande es la diferencia que hay entre x_i y $x(t_i)$ para algún i específico?. Tenemos que dicha diferencia representa el error en que incurrimos en el paso i y puede considerarse que está determinado principalmente por dos clases de error:

- ***El error de truncamiento.*** Es la diferencia que hay entre la solución exacta y la solución aproximada por la ecuación en diferencias que usemos. Más adelante nos daremos cuenta de que el nombre de este tipo de error obedece a que muchos de los métodos de integración están basados en el desarrollo en series de Taylor de $x(t)$ que necesariamente deben ser truncadas a partir de algún término.

- ***El error de redondeo.*** Representa la diferencia entre el valor teórico de la aproximación y el valor numérico que en realidad maneja la computadora, dado que ésta trabaja con una cantidad limitada de dígitos. Así pues, el tamaño del error de redondeo dependerá del número de dígitos que sea capaz de representar la computadora en que sean implantados los algoritmos matemáticos.

Dado que cada aproximación obtenida depende de una o más aproximaciones anteriores, es evidente que el error depende de los errores cometidos en pasos precedentes. Si consideramos que se tienen los valores exactos de los puntos anteriores requeridos para calcular $x(t_i)$, podemos definir el *error local de truncamiento* en el i-ésimo paso, $d_i(h)$, como la cantidad por la cual la solución exacta difiere de la proporcionada por el método de diferencias. Si no se asume que los

valores anteriores son exactos, a la diferencia teórica sin tomar en cuenta el error de redondeo, $e_i = x_i - x(t_i)$, le llamaremos *error global de truncamiento* hasta el i-ésimo paso.

2.4 Convergencia de un Método de Diferencias

Otra pregunta asociada al problema de resolver la ecuación diferencial (2.2) es: ¿Qué pasa cuando $h \to 0$? ¿Será entonces que las aproximaciones tienden a la solución exacta? Ésta es una condición deseable para un método de integración numérico, ya que, reduciendo el tamaño de paso h es posible reducir el error de truncamiento tanto como se desee.

Cuando el error que se comete en la aproximación tiende a cero con el tamaño de paso, se dice que el método es convergente. Más precisamente diremos que:

Definición 2.1 Se dice que un método de integración numérica de la ecuación (2.2) *es convergente* si

$$\lim_{h \to 0} \max_{1 \le i \le N} |x_i - x(t_i)| = 0,$$

donde $x(t_i)$ denota el valor exacto de la solución de la ecuación (2.2) en el punto t_i, y x_i es la aproximación teórica obtenida por el método en el i-ésimo paso.

Para medir la convergencia de un método se compara ésta con la rapidez de convergencia de otras funciones conocidas como las funciones de la forma

$$G(h) = h^p.$$

La rapidez de convergencia se establece usando la terminología siguiente:

Definición 2.2. Si $\lim_{h \to 0} x_i = x(t_i)$, se dice que la *convergencia del método es de orden* $O(G(h))$, si existe un número $K > 0$, independiente de h, tal que

$$\left| \frac{e_i}{G(h)} \right| = \left| \frac{x_i - x(t_i)}{G(h)} \right| \le K, \text{ para toda } h>0, \text{ en una vecindad del origen.}$$

Esta situación se expresa escribiendo $x_i = x(t_i) + O(\mathrm{G}(h))$ ó diciendo que $x_i \to x(t_i)$ con una rapidez de convergencia $O(\mathrm{G}(h))$. Adicionalmente, el método se llama de *orden p* si $\mathrm{G}(h) \approx h^p$.

Los métodos de integración numérica dependen de varios parámetros, por ejemplo del tamaño de paso h (o del $max\{h_i\}$ si el tamaño de paso es variable) o del número de términos en la expansión en series. Además de saber que el método usado es convergente, es importante obtener fórmulas que nos permitan escoger estos parámetros para lograr un grado de exactitud prefijado.

2.5 Error de Redondeo vs Error de Truncamiento

Aunque el error de redondeo no siempre es significativo, podría adquirir dimensiones importantes si no se escoge un tamaño de paso adecuado. Esto no quiere decir que sea muy difícil elegir h. Casi siempre, este tipo de errores se desarrolla mucho cuando el tamaño de paso que se usa es demasiado pequeño en términos de los dígitos que se manejen al efectuar los cálculos. Existe un h mínimo en donde se logra el menor error. Por ejemplo, para el Método de Euler (de un paso) tenemos una representación gráfica de los errores en la figura 2.1.

En la práctica difícilmente encontraremos el valor de h que produzca el menor error. Sin embargo, veremos que sí es posible mantener el error dentro de una cota específica (diferente de cero) escogiendo un tamaño de paso adecuado, si es que el método es convergente.

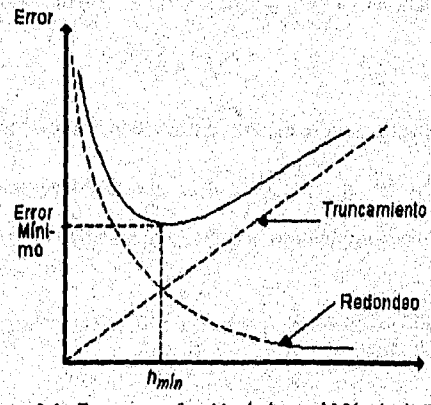


Figura 2.1. Error como función de h, en el Método de Euler

2.6 Estabilidad

Dado que cuando se construye una solución numérica se está sujeto a errores de redondeo en la condición inicial, es deseable que cambios pequeños en esta, produzcan cambios subsecuentemente pequeños en las aproximaciones. Para métodos de variable discreta podemos definir el concepto de *estabilidad absoluta* como,

Definición 2.3. Un método es *absolutamente estable* para un tamaño de paso h (o $max\{h_i\}$, i=1,2,... si el método es de paso variable) y una ecuación diferencial dada, si el cambio producido por una peturbación de tamaño δ en uno de los puntos x_i no es más grande que δ en todos los subsecuentes valores x_k, $k > i$.

Desafortunadamente esta definición es dependiente del problema, así que para analizar la estabilidad absoluta, se utiliza una "ecuación prueba" $\dot{x} = \lambda x$ donde λ es una constante compleja, y se dice que la región de estabilidad absoluta es aquel conjunto de valores de $h \geq 0$ y λ para los cuales una perturbación en un simple valor x_m producirá un cambio en valores subsecuentes x_{m+1}, x_{m+2},... que no se incrementa paso a paso.

Recordemos que una ecuación diferencial $\dot{x}(t) = f(t,x)$ tiene una familia de soluciones constituida por curvas que se corresponden biunívocamente con alguna condición inicial. Cuando el valor x_0 sufre una perturbación, la solución se mueve hacia otra curva de la familia de soluciones. En el caso de una ecuación en diferencias, sucede lo mismo. Que el método sea absolutamente estable significa que cada curva de la familia de soluciones a la que nos traslademos no se separa de la curva exacta en una cantidad mayor a la perturbación que ocurra en algún paso. Igualmente, asegura que los errores introducidos en cada paso tienen un pequeño efecto en las subsecuentes aproximaciones. Por otra parte, podría bastarnos con poder asegurar que la solución numérica no se separe de la solución exacta más allá de una cantidad fija no necesariamente menor o igual que la perturbación ocurrida en los valores iniciales. En ese caso, hablaremos de *estabilidad* "a secas".

Definición 2.4. Si existe un $h_0 > 0$ para una ecuación diferencial tal que un cambio fijo en los valores iniciales produce un cambio acotado en la solución numérica para todo $0 \leq h \leq h_0$, entonces el método es *estable*.

2.7 Métodos de un Paso

El método general de un paso

Son de un paso las técnicas que utilizan únicamente el punto (t_i, x_i) para calcular x_{i+1}. Más precisamente:

Definición 2.5. Los métodos de un paso $x_{i+1} = G(t_i, x_i)$, para aproximar las soluciones de la ecuación diferencial (2.3) se pueden siempre escribir de la forma

$$x_{i+1} = x_i + h\psi(t_i, x_i, h), \tag{2.3}$$

donde ψ está determinada por f y está dada por $\psi = \dfrac{G(t_i, x_i) - x_i}{h}$.

Definición 2.6. Se dice que el método (2.3) tiene orden p si p es el mayor entero para el cual se cumple:

$$d_{i+1}(h) = x(t_{i+1}) - x_{i+1} = x(t_{i+1}) - x(t_i) - h\psi(t_i, x(t_i), h) = O(h^{p+1}), \quad \forall i, \tag{2.4}$$

donde $x(t)$ es la solución exacta de la ecuación (2.2).

Convergencia y estabilidad de los métodos de un paso

Formalmente, podemos definir la convergencia para métodos de un paso como

Definición 2.7. El método de un paso (2.3) *es convergente* si $max|x_i - x(t_i)| \to 0$ cuando $h = max|t_{i+1} - t_i| \to 0$ para alguna ecuación diferencial $\dot{x} = f(t, x(t))$ la cual satisface una condición de Lipschitz en x.

En la definición anterior puede notarse que se permiten errores en el valor x_0 dejando lugar a la pérdida de dígitos que pudiéramos tener.

Definición 2.8. El método de un paso (2.3) es *estable* si para cada ecuación diferencial existen constantes positivas h_0 y K tales que la diferencia entre dos soluciones numéricas distintas x_i y $\hat{x}_i$ que satisfacen (2.3) es tal que

$$|x_i - \hat{x}_i| \le K|x_0 - \hat{x}_0| \quad \text{para todo} \quad h \in [0, h_0].$$

Los siguientes teoremas nos dan condiciones de convergencia y estabilidad.

Teorema 2.1. Si $\psi(t,x,h)$ es continua para $t \in [t_0,b]$, $h \in [0,h_0]$ y todo x y, si ésta satisface una condición de Lipschitz en x, en dicha región, una condición suficiente y necesaria para la convergencia es que

$$\psi(t,x;0) = f(t,x). \tag{2.5}$$

La demostración se puede encontrar en la referencia [1] pág. 57.

Teorema 2.2. Si $\psi(t,x,h)$ es continua para $t \in [t_0,b]$, $h \in [0,h_0]$ y todo x y, si ésta satisface una condición de Lipschitz en x en dicha región, entonces el método dado por (2.3) es estable.

Demostración

Sean $|e_{i+1}| = |x_{i+1} - \hat{x}_{i+1}|$ y L_x la constante de Lipschitz de ψ en x. Entonces,

$$\begin{aligned} |e_{i+1}| &= |x_i + h\psi(t_i,x_i,h) - \hat{x}_i - h\psi(t_i,\hat{x}_i,h)| \\ &\le |x_i - \hat{x}_i| + h|\psi(t_i,x_i,h) - \psi(t_i,\hat{x}_i,h)| \\ &\le |e_i| + hL_x|e_i| = (1+hL_x)|e_i| \\ &\le (1+hL_x)^{i+1}|e_0| \end{aligned}$$

luego entonces, $$|e_i| \le (1+hL_x)^i |e_0|.$$

Como $hL_x \ge 0$, expandiendo la función e^{hL_x} en su serie de Taylor alrededor de h=0, tenemos que $e^{ihL_x} \ge (1+hL_x)^i$.

Si $0 \le ih \le b-a$ entonces $|e_i| \le e^{L_x(b-a)}|e_0|$.

Como la expresión obtenida no depende explícitamente de h, entonces se cumple para cualquier tamaño de paso y el teorema queda demostrado.

Error en los métodos de un paso

Ya hemos visto que el error local de truncamiento en el paso x_{i+1}, denotado por d_{i+1}, está dado por la diferencia entre el valor de la solución exacta en el tiempo t_{i+1} y el valor de la aproximación en el paso i+1, es decir,

$$d_{i+i}(h) = x(t_{i+1}) - x_{i+1} = x(t_{i+1}) - x(t_i) - h\psi(t_i, x(t_i), h) = O(h^{p+1}), \quad \forall i .$$

En esta sección veremos que si un método de un paso cuya ψ cumple una condición de Lipschitz en x tiene un error local de orden $p+1$, entonces su error global será de orden p. Para esto, demostraremos el siguiente lema.

Lema 2.1. Si $|e_i|$ satisface

$$|e_i| \le |e_{i-1}|(1+hL) + D, \tag{2.6}$$

para dos constantes L y D mayores que cero, y $0 \le ih \le b - t_0$, entonces

$$\begin{aligned} |e_i| &\le D \cdot \frac{(1+hL)^i - 1}{hL} + (1+hL)^i |e_0| \\ &\le \frac{D}{hL}(e^{L(b-t_0)} - 1) + e^{L(b-t_0)} |e_0|. \end{aligned} \tag{2.7}$$

Demostración

La primera desigualdad se prueba por inducción. Evidentemente es verdadera para $i = 1$. Asumiendo que es válido cuando $i = k$, por (2.6) y (2.7) obtenemos para $i = k+1$,

$$\begin{aligned} |e_{k+1}| &\le |e_k|(1+hL) + D \\ &\le (1+hL)\left[D \cdot \frac{(1+hL)^k - 1}{hL} + (1+hL)^k |e_0| \right] \\ &\le D \cdot \frac{(1+hL)^{k+1} - 1}{hL} + (1+hL)^{k+1} |e_0|. \end{aligned}$$

De donde la primera desigualdad de (2.7) es verdadera. La segunda desigualdad se deduce de que para $hL \ge 0$ tenemos que $(1+hL) \le e^{hL}$; en particular $(1+hL)^i \le e^{ihL} \le e^{(b-t_0)L}$.

Procedamos ahora a buscar una cota para el error global

$$-e_{i+1} = x_{i+1} - x(t_{i+1})$$

para el cual se tiene,

$$-e_{i+1} = x_i - h\psi(t_i, x_i, h) - x(t_i) - h\psi(t_i, x(t_i), h) + d_{i+1}(h),$$
$$= e_i - h\left[\psi(t_i, x_i, h) - \psi(t_i, x(t_i), h)\right] + d_{i+1}(h).$$

Bajo el supuesto de que ψ cumple una condición de Lipschitz en x, tenemos

$$|e_{i+1}| \le |e_i| + hL_x|e_i| + Dh^{p+1},$$
$$|e_{i+1}| \le |e_i|(1 + hL_x) + Dh^{p+1}.$$

Por el lema 2.1 tenemos que las desigualdades anteriores equivalen a

$$|e_i| \le Dh^{p+1} \cdot \frac{(1+hL_x)^i - 1}{hL_x} + (1+hL_x)^i |e_0|,$$
$$|e_i| \le Dh^p \frac{(e^{(L_x(b-t_0)} - 1) - 1}{L_x} + e^{L_x(b-t_0)}|e_0| = O(h^p). \qquad (2.8)$$

De la desigualdad (2.8) vemos que cuando $e_0 \to 0$, el método cuyo error local es $O(h^{p+1})$ tendrá un error global $O(h^p)$.

Teorema 2.3. Si un método de un paso cuya ψ satisface una condición de Lipschitz con respecto a x, tiene un error local de orden $p+1$, entonces tiene un error global de orden p, acotado por

$$|e_i| \le Dh^p \frac{(e^{(L_x(b-t_0)} - 1) - 1}{L_x} + e^{L_x(b-t_0)}|e_0|.$$

Métodos de Taylor

Un ejemplo de métodos de un paso son los métodos de Taylor. De hecho constituyen la idea fundamental sobre la cual se construyen la mayoría de métodos de un paso y multipaso. Dado el problema de valor inicial (2.2), para encontrar la solución $x(t)$, supongamos que ésta tiene derivadas al menos de orden $n+1$. Entonces, la solución se puede expandir en términos de su polinomio de Taylor de n-ésimo grado alrededor de t_i:

$$x(t_{i+1}) = x(t_i) + h\dot{x}(t_i) + \frac{h^2}{2}\ddot{x}(t_i) + \cdots + \frac{h^n}{n!}x^{(n)}(t_i) + \frac{h^{n+1}}{(n+1)!}x^{(n+1)}(\xi_i) \qquad (2.9)$$

donde ξ_i es algún punto del intervalo (t_i, t_{i+1}). Un método de Taylor de orden p se obtiene despreciando el último término que involucra a ξ_i. Entonces, el algoritmo que determina un método de Taylor de orden m queda expresado como sigue:

$$x_0 = x(t_0),$$
$$x_{i+1} = x_i + h\dot{x}_i + \frac{h^2}{2}\ddot{x}_i + \cdots + \frac{h^p}{p!}x_i^{(p)}, \tag{2.10}$$

donde,
$$\dot{x}_i = f(t_i, x_i)$$
$$x_i^{(p)} = \left.\frac{d^{p-1}f(t,x)}{dt^{p-1}}\right|_{t=t_i}.$$

Por ejemplo, el método de Taylor de segundo orden estará dado por

$$x_{i+1} = x_i + hf(t_i, x_i) + \frac{h^2}{2}\left[f_t(t_i, x_i) + f_x(t_i, x_i)f(t_i, x_i)\right],$$

donde f_t y f_x denotan las derivadas parciales $\frac{\partial f}{\partial t}$ y $\frac{\partial f}{\partial x}$ respectivamente.

Los métodos de Taylor de orden mayor, tienen el inconveniente de requerir el conocimiento de las derivadas de la función que define la ecuación diferencial. Más adelante veremos otros métodos (de Runge-Kutta) que permiten obtener aproximaciones de orden mayor que 2 sin este requerimiento.

Para un método de orden p, el error local de truncamiento en el paso i-ésimo está dado por $d_i(h) = \frac{h^{p+1}}{(p+1)!}x^{p+1}(\xi_i)$ que es de orden $O(h^{p+1})$. En consecuencia, el método tendrá un error global de orden $O(h^p)$.

Método de Euler

El único método de Taylor de primer orden es el Método de Euler o Método de las Tangentes. A continuación consideraremos el caso de un sistema de dimensión uno para interpretar geométricamente el procedimiento de dicho método.

Como se conocen t_0 y $x(t_0)$, también se conoce la pendiente de la tangente en $t = t_0$, ésto es $\dot{x} = f(t_0, x_0)$. Por lo tanto, podemos trazar la tangente a la gráfica de la

solución en t_0 y obtener un valor aproximado x_1, de $x(t_1)$, moviéndonos a lo largo de la tangente hacia t_1 como se muestra en la figura 2.2.

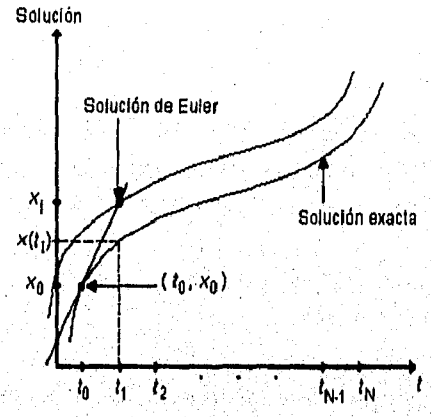


Figura 2.2. Solución por el método de Euler

Dado que $x_1 \approx x(t_1)$, entonces podemos decir que la tangente a la solución en el punto (t_1,x_1) tiene una pendiente aproximada de $\dot{x}_1 = f(t_1,x_1)$. Así, procediendo como antes, es posible trazar la tangente a la solución en t_1 para obtener un valor aproximado x_2 para $x(t_2)$. Si repetimos este proceso para los siguientes puntos del intervalo en que se definió t, obtenemos el método de Euler:

$$\begin{aligned} x_0 &= x(t_0), \\ x_{i+1} &= x_i + hf(t_i,x_i). \end{aligned} \tag{2.11}$$

Métodos de Runge-Kutta

Los Métodos de Runge-Kutta constituyen otro ejemplo de métodos de un paso. Se pensaron con el objetivo de contar con métodos que tuvieran orden p mayor o igual que 2 sin tener que conocer las derivadas de f. Pero, ¿Cómo garantizar que realmente tales métodos fueran de orden p tal como éste se definió? La respuesta es sencilla: proponiendo una forma para dichos métodos y después tratar de hacer coincidir tal expresión con la serie que define al método de Taylor de orden p. En lo que resta de ésta sección se profundiza en dicha idea.

Se propone que un método de *Runge Kutta de orden 2* tenga la siguiente forma:

$$x_{i+1} = x_i + a_1 k_1 + a_2 k_2, \tag{2.12}$$

en donde,

$$k_1 = hf(t_i, x_i),$$
$$k_2 = hf(t_i + \alpha_1 h, x_i + \beta_1 k_1).$$

Para que el orden de convergencia del método sea efectivamente dos, se escogen las constantes a_1, a_2, α_1, β_1 de manera que la ecuación en diferencias (2.12) coincida con la ecuación (2.10) hasta el término que contiene a h^2, es decir

$$x_i + a_i k_1 + a_2 k_2 = x_i + hf(t_i, x_i) + \frac{h^2}{2} \dot{f}(t_i, x_i).$$

Como el término x_i se encuentra en ambos lados puede omitirse. Tendremos entonces que ambas ecuaciones tienen el factor común h, por lo que el problema se reduce a resolver la igualdad:

$$a_1 f(t,x) + a_2 f(t + \alpha_1 h, x + \beta_1 h f(t,x)) = f(t,x) + \frac{h}{2} \dot{f}(t,x). \tag{2.13}$$

Por la regla de la cadena,

$$\dot{f}(t,x) = \frac{df}{dt}(t,x) = \frac{\partial f}{\partial t}(t,x) + \frac{\partial f}{\partial x}(t,x)\frac{dx}{dt} \qquad \text{y} \qquad \frac{dx}{dt} = f(t,x).$$

Sustituyendo en la ecuación (2.13),

$$a_1 f(t,x) + a_2 f(t + \alpha_1 h, x + \beta_1 h f(t,x)) = f(t,x) + \frac{h}{2}\frac{\partial f}{\partial t}(t,x) + \frac{h}{2}\frac{\partial f}{\partial x}(t,x) f(t,x). \tag{2.14}$$

Expandiendo el miembro izquierdo de la última ecuación en su polinomio de Taylor en dos variables de grado 1, alrededor de (t, x) tenemos:

$$a_1 f(t,x) + a_2 f(t + \alpha_1 h, x + \beta_1 h f(t,x)) = a_1 f(t,x) + a_2 f(t,x) + a_2 \alpha_1 h \frac{\partial f}{\partial t}(t,x) + a_2 \beta_1 h f(t,x) \frac{\partial f}{\partial x}(t,x) + O(h^2).$$

Sustituyendo este valor en el primer miembro de la ecuación (2.14) y despreciando el término $O(h^2)$, nos queda

$$a_1 f(t,x) + a_2 f(t,x) + a_2\alpha_1 h\frac{\partial f}{\partial t}(t,x) + a_2\beta_1 h f(t,x)\frac{\partial f}{\partial x}(t,x) = f(t,x) + \frac{h}{2}\frac{\partial f}{\partial t}(t,x) + \frac{h}{2}\frac{\partial f}{\partial x}(t,x) f(t,x).$$

Dado que el polinomio de Taylor es único, necesariamente, $a_1 + a_2 = 1$; $a_2\alpha_1 = \frac{1}{2}$; $a_2\beta_1 = \frac{1}{2}$.

Estas ecuaciones tienen infinidad de soluciones, en particular cuando $a_1 = a_2 = \frac{1}{2}$, $\alpha_1 = \beta_1 = 1$, tendremos el método de Euler Mejorado. De la misma forma, cuando $a_1 = 0$, $a_2 = 1$ y $\alpha_1 = \beta_1 = \frac{1}{2}$ tendremos el método del Punto Medio y, con $a_1 = \frac{1}{4}$, $a_2 = \frac{3}{4}$ y $\alpha_1 = \beta_1 = \frac{2}{3}$, resulta el Método de Heun. La interpretación geométrica de éstos métodos se discutirá en las siguientes 3 secciones.

Análogamente, para obtener un método de Runge-Kutta de orden 3 se buscan valores para $a_1, a_2, a_3, \alpha_1, \alpha_2, \beta_1, \beta_2$ tales que $x_{i+1} = x_i + a_1k_1 + a_2k_2 + a_3k_3$ coincida la ecuación en diferencias (2.10) hasta el término que contiene a h^3. Los métodos de cuarto orden son los más frecuentemente utilizados.

Deducir un método de Runge Kutta de cuarto orden consiste en determinar $a_1, a_2, a_3, a_4, \alpha_1, \alpha_2, \alpha_3, \beta_1, \beta_2, \beta_3$ tales que $x_{i+1} = x_i + a_1k_1 + a_2k_2 + a_3k_3 + a_4k_4$, donde,

$$\begin{aligned} k_1 &= hf(t_i, x_i), \\ k_2 &= hf(t_i + \alpha_1 h, x_i + \beta_1 k_1), \\ k_3 &= hf(t_i + \alpha_2 h, x_i + \beta_2 k_1 + \beta_3 k_2), \\ k_4 &= hf(t_i + \alpha_3 h, x_i + \beta_4 k_1 + \beta_5 k_2 + \beta_6 k_3). \end{aligned}$$

coincida con la ecuación en diferencias (2.10) hasta el término que contiene a h^4. La deducción del método es totalmente análoga a la correspondiente para orden 2, pero el álgebra necesaria es demasiada por lo que se omite y solamente se darán los 2 métodos de R-K-4 más usados.

El primero está dado por las ecuaciones:

$$
\begin{aligned}
w_0 &= \alpha \\
k_1 &= hf(t_i, x_i) \\
k_2 &= hf(t_i + \tfrac{1}{2}h, x_i + \tfrac{1}{2}k_1) \\
k_3 &= hf(t_i + \tfrac{1}{2}h, x_i + \tfrac{1}{2}k_2) \\
k_4 &= hf(t_{i+1}, x_i + k_3) \\
w_{i+1} &= w_i + \tfrac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4).
\end{aligned}
$$

El segundo:

$$
\begin{aligned}
w_0 &= \alpha \\
k_1 &= hf(t_i, x_i) \\
k_2 &= hf(t_i + \tfrac{1}{3}h, x_i + \tfrac{1}{3}k_1) \\
k_3 &= hf(t_i + \tfrac{2}{3}h, x_i + \tfrac{1}{3}k_1 + \tfrac{1}{3}k_2) \\
k_4 &= hf(t_{i+1}, x_i + k_1 - k_2 + k_3) \\
w_{i+1} &= w_i + \tfrac{1}{8}(k_1 + 3k_2 + 3k_3 + k_4).
\end{aligned}
$$

Método de Euler Mejorado

Con este método se busca mejorar la aproximación obtenida por el método de Euler. Para encontrar la aproximación x_{i+1}, utiliza la aproximación anterior x_i y obtiene una aproximación x_E de Euler para $x(t_{i+1})$, por lo que se pueden trazar las rectas tangentes a la solución en los puntos (t_i, x_i) y (t_{i+1}, x_E) y dibujar una recta más que pase por el punto donde se intersectan las otras dos pero cuya pendiente es el promedio de éstas, es decir, estará exactamente en medio.

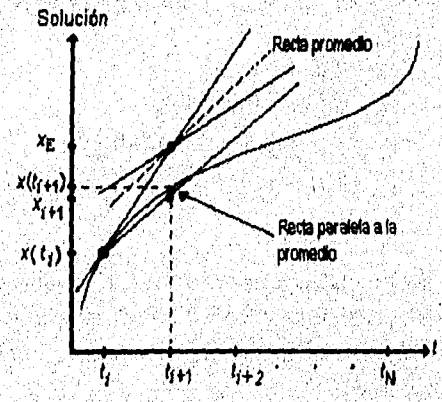


Figura 2.3 Solución por el Método de Euler Mejorado

Finalmente, se traza una recta paralela a la recta promedio que pase por el punto (t_i,x_i) y se desplaza hacia t_{i+1}, obteniendo así, una mejor aproximación x_{i+1} de $x(t_{i+1})$. Esto se ilustra en la figura 2.3.

De ahí que el método de Euler Mejorado queda expresado como sigue:

$$x_0 = x(t_0),$$
$$x_{i+1} = x_i + \frac{h}{2}\left(f(t_i,x_i) + f(t_{i+1},x_E)\right),$$
$$\text{donde } x_E = x_i + hf(t_i,x_i).$$

Método del Punto Medio

El hecho de que en el método de Euler Mejorado se emplee una pendiente promedio para aproximar la solución en el punto $x(t_{i+1})$, podría sugerirnos que el nombre más adecuado para dicho método es, precisamente, el de *Método del Promedio o del Punto Medio*. Sin embargo, la realidad es que el método que se presentará tiene muy bien ganado su nombre puesto que, para obtener una aproximación de la solución en el punto t_{i+1}, no utiliza la pendiente $f(t_i,x_i)$; sino $f(t_{i+h/2},x_{E/2})$ donde $x_{E/2}$ es la aproximación de Euler para $x(t_{i+h/2})$.

Geométricamente (figura 2.4), trazamos una recta que pase por el punto (t_i,x_i) y que sea paralela a la tangente en el punto $(t_{i+h/2},x_E)$ y nos movemos a lo largo de ésta hasta el punto t_{i+1} para obtener una aproximación x_{i+1} de $x(t_{i+1})$.

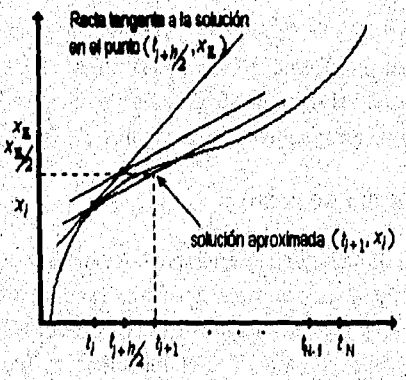


Figura 2.4 Solución por el Método del Punto Medio

Simbólicamente, el método queda expresado como sigue:

$$x_0 = x(t_0),$$

$$x_{i+1} = x_i + hf\left(t_i + \frac{h}{2}, x_{E/2}\right),$$

donde, $$x_{E/2} = x_i + \frac{h}{2} f(t_i, x_i).$$

Método de Heun

Otro método de Runge-Kutta de orden 2 utilizado con frecuencia es el método de Heun y se origina cuando $a_1 = \frac{1}{4}$, $a_2 = \frac{3}{4}$ y $\alpha_1 = \beta_1 = \frac{2}{3}$. Simbólicamente tenemos:

$$x_0 = x(t_0),$$

$$x_{i+1} = x_i + h\left[\frac{1}{4} f(t_i, x_i) + \frac{3}{4} f\left(t_i + \frac{2}{3}h, x_i + \frac{2}{3}hf(t_i, x_i)\right)\right].$$

Dicha expresión puede interpretarse como sigue. Se obtiene una aproximación de $x(t_i + \frac{2}{3}h)$ dada por

$$x_E = x_i + \frac{2}{3}hf(t_i, x_i).$$

Este punto es más cercano a x_{i+1} que x_i. Por lo tanto $f(t_i + \frac{2}{3}h, x_E)$ ilustra mejor el comportamiento de $f(t_{i+1}, x_{i+1})$ que $f(t_i, x_i)$. Así, para obtener el incremento con que nos moveremos desde (t_i, x_i), se hace un "promedio pesado" entre ambos valores asignando a $f(t_i + \frac{2}{3}h, x_E)$ el 75% de peso. Nótese que el analista numérico podría distribuir los pesos en otra forma, sobre todo si tuvlese alguna información sobre el comportamiento de $x(t)$. Hacer una asignación distinta de los pesos, haría que el método ya no tuviera un error equivalente al del método de Taylor de orden 2.

Variación del tamaño de paso

Cuando aplicamos un método numérico para obtener una solución de una ecuación diferencial ordinaria, quisiéramos poder ajustar en cada iteración el tamaño de paso a fin de no sobrepasar un límite establecido de error. Si el error local no estuviera dentro de la tolerancia deseada el paso debería reducirse; si fuese demasiado pequeño respecto a dicha tolerancia, debería agrandarse.

Vale la pena hacer notar que el comportamiento del error además de depender del tamaño de paso depende también de la solución de la ecuación diferencial. Supongamos que x_1 y x_2 son dos aproximaciones numéricas de la solución teórica en $x(t_i)$ y $x(t_{i+1})$ respectivamente. Cuando $x(t)$ crece o decrece suavemente, la distancia entre t_{i+1} y t_i no debería ser "demasiado" pequeña ya que no se espera que haya un cambio muy significativo entre x_1 y x_2. Por el contrario, si $x(t)$ cambia bruscamente, t_{i+1} y t_i deberían estar suficientemente cerca para no perder información de x en dicho intervalo.

En esta sección se construirá un algoritmo que permita variar el tamaño de paso en cada iteración, por un lado para mantener el error en un rango y, al mismo tiempo para responder al comportamiento de la solución no haciendo más o menos cálculos de los necesarios.

Para introducir la deducción de nuestro algoritmo, requerimos establecer lo que se conoce como la expresión asintótica del error local para un método de orden p. Para todo método de un paso, el error local de truncamiento, $d_i(h)$ se puede expresar como:

$$d_i(h) = h^{p+1}\phi(t,x) + O(h^{p+2}).$$

Obtengamos ahora una aproximación práctica del error local de truncamiento. Supóngase que $x(t+2h)$ es la solución teórica evaluada en el punto $t+2h$ y que x_1 y x_2 son dos aproximaciones obtenidas desde t con tamaños de paso $2h$, y h respectivamente, mediante un método de un paso de orden p. Entonces , los errores locales respectivos están dados por

$$x(t+2h) - x_1 = (2h)^{p+1}\phi(t+2h, x(t+2h)) + O(h^{p+2}), \tag{2.15}$$

$$x(t+2h) - x_2 = 2h^{p+1}\phi(t+2h, x(t+2h)) + O(h^{p+2}). \tag{2.16}$$

Sustrayendo (2.16) de (2.15), resulta

$$\phi(t+2h, x(t+2h)) = \frac{x_2 - x_1}{2h^{p+1}(2^p - 1)} + O(h^{p+2}).$$

Sustituyendo el valor obtenido para $\phi(t+2h, x(t+2h))$ en (2.16),

$$x(t+2h)-x_2=\frac{x_2-x_1}{2^p-1}+O(h^{p+2}), \tag{2.17}$$

$$\hat{x}(t+2h)=x_2+\frac{x_2-x_1}{2^p-1}. \tag{2.18}$$

De la fórmula (2.17) podemos decir que el error local está aproximado por $d(h)=\frac{x_2-x_1}{2^p-1}$. La ecuación (2.18) representa un método de orden $p+1$ en lugar de un método de orden p como los métodos usados para obtener x_1 y x_2.

Para variar el tamaño de paso, $d(h)$ es comparado con una tolerancia deseada T para determinar si es necesario o no reducir el tamaño de paso. Cuando el error resulta ser "demasiado" pequeño, se debe agrandar el tamaño de paso. En caso de que el error se salga de la tolerancia, se debe obtener un nuevo tamaño de paso $\hat{h}$. Tomemos el siguiente criterio: considerando que el error es aproximadamente igual a $\phi(t+2\hat{h},x(t+2\hat{h}))\hat{h}^{p+1}$ para algún tamaño de paso $\hat{h}$, entonces,

$$\hat{h}=\left(\frac{T}{\phi(t+\hat{h},x(t+\hat{h}))}\right)^{\frac{1}{p+1}}\approx\left(\frac{Th^{p+1}}{\phi(t+h,x(t+h))h^{p+1}}\right)^{\frac{1}{p+1}}=h\left(\frac{T}{d(h)}\right)^{\frac{1}{p+1}} \tag{2.19}$$

Debemos tener cierto cuidado para controlar el valor de $\hat{h}$. Generalmente los autores recomiendan multiplicar (2.19) por un factor de seguridad s (véase referencia [2]), usualmente s=0.8, 0.9, $(0.25)^{\frac{1}{p+1}}$, o $(0.38)^{\frac{1}{p+1}}$, de tal forma que muy probablemente será aceptado el error en el siguiente paso.

Que el error sea "demasiado" pequeño respecto a la tolerancia significa que está por debajo de una cifra mínima estipulada por el analista numérico. Por ejemplo, algunos autores proponen que esté dada por $(s/4)^{p+1}$, así como que h sea multiplicado por 4 cuando deba agrandarse.

Finalmente, deben fijarse valores máximos y mínimos en el tamaño de paso para evitar problemas de *"under flow"* en la computadora. La fórmula (2.19) para $\hat{h}$ quedaría como,

$$\hat{h} = h \cdot \min\left(h_{\max}, \max\left(h_{\min}, s \cdot \left(\frac{T}{d(h)} \right)^{\frac{1}{p+1}} \right) \right)$$

Aplicación de las técnicas de un paso a sistemas de ecuaciones diferenciales

El objetivo de esta sección es dar la forma en que las técnicas de un paso pueden extenderse a sistemas de cuaciones de primer orden. En realidad, dichas técnicas son directamente aplicables, ya que las ecuaciones que componen al sistema se trabajan simultanea pero separadamente. Así, podemos decir que la forma general de un método de un paso aplicado a sistemas está dada por

$$\begin{aligned} \mathbf{x}_0 &= \mathbf{x}(t_0), \\ \mathbf{x}_{i+1} &= \mathbf{x}_i + h\psi(t_i, \mathbf{x}_i, h), \end{aligned}$$

donde $\mathbf{x}_i$ y $\mathbf{x}_{i+1}$ son vectores de n componentes que aproximan la solución $\mathbf{x}(t)$ en los pasos i e $i+1$ respectivamente. ψ es una función vectorial determinada por $\mathbf{f}$ del problema (2.1). Por ejemplo, para el método de Euler se tiene $\psi=\mathbf{f}$, y por tanto

$$\begin{aligned} \mathbf{x}_0 &= \mathbf{x}(t_0) \\ \mathbf{x}_{i+1} &= \mathbf{x}_i + h\mathbf{f}(t_i, \mathbf{x}_i, h), \end{aligned}$$

que es equivalente a

$$\begin{aligned} x_{1,0} &= x_1(t_0) \\ x_{2,0} &= x_2(t_0) \\ &\vdots \\ x_{n,0} &= x_n(t_0) \\ x_{1,i+1} &= x_{1,i} + hf_1(t_i, x_{1,i}, x_{2,i}, \cdots, x_{n,i}) \\ x_{2,i+1} &= x_{2,i} + hf_2(t_i, x_{1,i}, x_{2,i}, \cdots, x_{n,i}) \\ &\vdots \\ x_{n,i+1} &= x_{n,i} + hf_n(t_i, x_{1,i}, x_{2,i}, \cdots, x_{n,i}). \end{aligned}$$

Por otra parte, al trabajar con un sistema de ecuaciones, en cada paso y para cada miembro del sistema se producirá un error, por lo que tendremos un vector $\mathbf{e}_i$ de errores. También se deben obtener cotas de dichos errores para asegurar que todos

éstos se mantengan dentro de un rango. Por tanto, conviene tener una sola cantidad que sirva como medida del tamaño de todos los errores. Para que la teoría presentada en las secciones anteriores sea aplicable a sistemas de ecuaciones, nuestra medida deberá tener las mismas propiedades que la transformación valor absoluto. La medida que tomaremos será una norma de vectores que denotaremos por $\|\mathbf{e}_i\|$. Por ejemplo, podemos definir $\|\mathbf{e}_i\| = \max_j |e_j|$, donde e_i son los componentes de $\mathbf{e}_i$. Igualmente podríamos haber elegido $\|\mathbf{e}_i\| = \sqrt{\sum_j (e_j)^2}$ que corresponde a la norma Euclidiana.

La teoría referente a la convergencia, error y estabilidad en los métodos de un paso puede ser generalizada de manera natural si consideramos vectores y normas de vectores en lugar de escalares y valores absolutos.

2.8 Métodos Multipaso Lineales

El método general multipaso

Se considera el problema de valor inicial (2.2),

$$\frac{dx}{dt} = \dot{x}(t) = f(t,x),$$

con condición inicial $x(t_0) = x_0$, donde $t_0 \le t \le b$ y $t_0 \in [a,b]$ y se supone que f satisface condiciones de convergencia, estabilidad, existencia y unicidad.

Como puntos de red consideramos la sucesión $\{t_i\}$ definida por $t_i = t_0 + ih$, con $n = 0,1,2,\ldots$. Mientras no se especifique otra cosa, h será constante.

Un método multipaso lineal puede expresarse en términos generales mediante la ecuación en diferencias

$$\sum_{j=0}^{k} \alpha_j x_{i+j} = \sum_{j=0}^{k} \beta_j f_{i+j}, \qquad (2.20)$$

donde x_{i+j} es una aproximación de $x(t_{i+j})$ y $f_i = f(t_i, x_i)$, y cada α_j, β_j son coeficientes constantes tales que $\alpha_k \neq 0$ y, $\alpha_0 \neq 0$ ó $\beta_0 \neq 0$. Nótese que el planteamiento no se altera si α_k=1 (pues ambos miembros de (2.20) pueden multiplicarse por $1/\alpha_k$), por lo que sucesivamente así se considerará. Diremos que (2.20) es un método explícito si $\beta_k = 0$, e implícito si $\beta_k \neq 0$.

Derivación de un Método Multipaso Lineal

Un método multipaso lineal puede derivarse de tres maneras: a través de expansiones de Taylor, a través de integración numérica o por interpolación. La idea general en cada caso será discutida mediante algún ejemplo en lo que resta de esta sección.

Derivación por Expansiones de Taylor

Considérense las expansiones de Taylor para $x(t_i + h)$ y $x(t_i - h)$ alrededor de x_i:

$$x(t_i + h) = x(t_i) + hx^{(1)}(t_i) + \frac{h^2}{2!}x^{(2)}(t_i) + \frac{h^3}{3!}x^{(3)}(t_i) + \ldots,$$

$$x(t_i - h) = x(t_i) - hx^{(1)}(t_i) + \frac{h^2}{2!}x^{(2)}(t_i) - \frac{h^3}{3!}x^{(3)}(t_i) + \ldots.$$

Sustrayendo obtenemos:

$$x(t_i + h) - x(t_i - h) = 2hx^{(1)}(t_i) + \frac{h^3}{3!}x^{(3)}(t_i) + \ldots.$$

Si despreciamos el término que contiene a h^3 y sustituimos $x^{(1)}(t)$ por $f(t,x)$, tendremos que $2hf(t_i, x(t_i))$ aproxima la parte izquierda de la expresión con un error local de truncamiento de

$$\pm\tfrac{1}{3}h^3x^{(3)}(x_i) + \ldots.$$

Sustituir los valores exactos de $x(t_i)$ y $x(t_i + h)$ por los aproximados x_i y x_{i+1}, produce el método multipaso lineal:

$$x_{i+2} - x_i = 2hf_{i+1}. \qquad (2.21)$$

De manera similar, es posible derivar algún otro método multipaso.

Derivación a través de integración numérica

Considérese la identidad,

$$x(t_{i+2}) - x(t_i) = \int_{t_i}^{t_{i+2}} \dot{x}(t)dt\,, \tag{2.22}$$

donde $\dot{x}(t) = f(t,x)$. Para derivar un método de dos pasos los únicos datos disponibles para evaluar la integral serán los valores f_i, f_{i+1}, f_{i+2}. Sea $P(t)$ el único polinomio de grado dos que pasa a través de los 3 puntos (t_i, f_i), (t_{i+1}, f_{i+1}), (t_{i+2}, f_{i+2}). Por la fórmula de interpolación de Newton-Gregory tenemos,

$$P(t) = P(t_i + rh) = f_i + r\Delta f_i + \frac{r(r-1)}{2!}\Delta^2 f_i\,,$$

donde $\Delta f_n = f_{n+1} - f_n$. Hagamos ahora la aproximación:

$$\int_{t_i}^{t_{i+2}} \dot{x}(t)dt = \int_{x_i}^{t_{i+2}} P(t)dt = \int_0^2 \left(f_i + r\Delta f_i + \frac{r(r-1)}{2!}\Delta^2 f_i \right) h dr$$

$$= h\left(2f_i + 2\Delta f_i + \frac{1}{3}\Delta^2 f \right).$$

Expandiendo Δf_i y $\Delta^2 f_i$ en términos de f_i, f_{i+1} y f_{i+2}, y sustituyendo en (2.22) tendremos

$$x_{i+2} - x_i = \frac{h}{3}(f_{i+2} + 4f_{i+1} + f_i), \tag{2.23}$$

conocida como la regla de Simpson que, además, es el método implícito de dos pasos más preciso que se puede derivar.

Si en lugar de (2.22), se tiene,

$$x(t_{i+2}) - x(t_{i+1}) = \int_{x_{i+1}}^{x_{i+2}} \dot{x}(t)dt,$$

mediante un proceso totalmente análogo al anterior obtenemos el Método de Adams-Moulton de dos pasos, dado por

$$x_{i+2} - x_{i+1} = \frac{h}{12}(5f_{i+2} + 8f_{i+1} - f_i).$$

Claramente, esta técnica puede utilizarse únicamente para derivar métodos con $\alpha_k = 1$, α_j=-1, $\alpha_n = 0$, $n = 0,1,\ldots,j-1,j+1,\ldots k$, $j \neq k$.

Derivación a través de interpolación

Nuevamente, haremos la ilustración derivando un método de 2 pasos. Sea $x(t)$ aproximada localmente en el intervalo $t_i \leq t \leq t_{i+2}$ por un polinomio $I(t)$. ¿Qué condiciones deberían imponerse a $I(t)$ para que fuese una buena representación de $x(t)$?. $I(t)$ debería interpolar los puntos (t_{i+j}, x_{i+j}), i=0,1,2 y además la derivada de $I(t)$ debería coincidir con las derivadas prescritas f_{i+j} para j=0,1,2. Entonces, las seis condiciones impuestas a $I(t)$ son:

$$I(t_{i+j}) = x_{i+j}; \qquad \dot{I}(t_{i+j}) = f_{i+j}, \quad \text{para } j=0,1,2.$$

Supongamos que $I(t)$ tiene cinco parámetros, es decir, es un polinomio de cuarto grado. Entonces,

$$I(t) = at^4 + bt^3 + ct^2 + dt + e.$$

Resolviendo el sistema de ecuaciones correspondiente para obtener los valores de los coeficientes, se obtiene la identidad:

$$x_{i+2} - x_i = \frac{h}{3}(f_{i+2} + 4f_{i+1} + f_i),$$

que es el método multipaso (2.23). La derivación de (2.23) por expansiones de Taylor muestra que el error local de truncamiento es $\pm\frac{1}{90}h^5x^{(5)}(t_i)+\ldots$

Nótese que el método será exacto cuando se aplique a un problema de valor inicial cuya solución teórica $x(t)$ sea un polinomio de grado no mayor que cuatro. En tal caso $I(t) = x(t)$.

De las técnicas de derivación, la más eficiente es la de expansiones de Taylor porque además de requerir menos cálculos, nos permite conocer información en cuanto al error del método multipaso que se derive. La última técnica tiene por objeto ilustrar que la aplicación de un método multipaso es equivalente a representar localmente la solución por un polinomio.

Convergencia, error y estabilidad

Definición 2.9. El método multipaso (2.20) es convergente si, para todos los problemas de valor inicial (2.2) cuya f satisface una condición de Lipschitz en x, tenemos que

$$\lim_{\substack{h\to 0 \\ ih=t-t_0}} x_i = x(t_i)$$

se cumple para $t \in [t_0, b]$ y para todas las soluciones $\{x_i\}$ de la ecuación de diferencias (2.20) se satisfacen las condiciones iniciales $x_\mu = \eta_\mu(h)$ para las cuales $\lim_{h\to 0} \eta_\mu(h) = x_0$, $\mu = 0,1,2,\ldots,k-1$.

Obsérvese que esta definición es "muy generosa" en las condiciones que impone a los valores x_μ, ya que no restringe a que éstas sean iguales a los valores de la solución exacta en los puntos de t correspondientes.

Para dar condiciones de convergencia, es necesario introducir alguna terminología en los párrafos siguientes.

Con el método lineal multipaso (2.2) se asocia el operador lineal de diferencias

$$\ell[x(t);h] = \sum_{j=0}^{k} [\alpha_j x(t+jh) - h\beta_j \dot{x}(t+jh)], \tag{2.24}$$

donde $x(t)$ es una función arbitraria, continuamente diferenciable en $[a,b]$. Expandiendo las funciones $x(t+jh)$ y $\dot{x}(t+jh)$ en series de Taylor alrededor de t y agrupando términos en (24), tenemos

$$\ell[x(t);h] = C_0 x(t) + C_1 h\dot{x}(t) + \cdots + C_q h^q x^{(q)}(t) + \cdots, \tag{2.25}$$

donde las C_q son constantes.

Mediante un simple desarrollo algebraico, se obtienen las siguientes fórmulas para las constantes C_q en términos de los coeficientes α_j, β_j,

$$\begin{aligned} C_0 &= \alpha_0 + \alpha_1 + \alpha_2 + \cdots + \alpha_k, \\ C_1 &= \alpha_1 + 2\alpha_2 + \cdots k\alpha_k - (\beta_0 + \beta_1 + \beta_2 + \cdots + \beta_k), \\ C_2 &= \frac{1}{q!}(\alpha_1 + 2^q \alpha_2 + \cdots k^q \alpha_k) \\ &\quad - \frac{1}{(q-1)!}(\beta_1 + 2^{q-1}\beta_2 + \cdots + k^{q-1}\beta_k), \qquad q = 2,3,\ldots \end{aligned} \tag{2.26}$$

Definición 2.10. El operador de diferencias (2.24) y el método multipaso asociado (2.20) son de orden p, si $C_0 = C_1 = \cdots = C_p = 0$, $C_{p+1} \neq 0$.

De acuerdo a la definición, las fórmulas (2.19) pueden ser utilizadas para derivar un método lineal multipaso de máximo orden para algún k fijo, haciendo $\alpha_j = 1$ y $\alpha_0 = t_0$.

Definición 2.11. El error de truncamiento local en t_{i+k} del método (2.20) está definido por $\ell[x(t);h]$ dado por (2.25), cuando $x(t)$ es la solución teórica del problema de valor inicial (2.2).

El error local de truncamiento no tiene la misma interpretación para métodos explícitos e implícitos. Para explicar lo anterior, consideremos la diferencia que se produce de despejar $x(t_{i+k})$ de (2.25) y x_{i+k} de (2.20) y restarlas,

$$x(t_{i+k}) - x_{i+k} = h\beta_k[f(t_{i+k}, x(t_{i+k})) - f(t_{i+k}, x_{i+k})] + \ell[x(t_i);h].$$

Por el teorema del valor medio

$$f(t_{i+k}, x(t_{i+k})) - f(t_{i+k}, x_{i+k}) = [x(t_{i+k}) - x_{i+k}]\frac{\partial f(t_{i+k}, \eta_{i+k})}{\partial x},$$

donde η_{i+k} es un punto interior del intervalo cuyos puntos terminales son x_{i+k} y $x(t_{i+k})$. Así

$$\ell[x(t_i);h]=\left[1-h\beta_k\frac{\partial f(t_{i+k},\eta_{i+k})}{\partial x}\right]\cdot[x(t_{i+k})-x_{i+k}]. \tag{2.27}$$

De (2.27), tenemos que para métodos explícitos, el error local es la diferencia que hay entre la solución teórica y la numérica, bajo el supuesto de que $x_{i+j}=x(t_{i+j})$, para $j=0,1,\ldots k-1$. Para un método implícito, el error de truncamiento local es (aproximadamente) proporcional a tal diferencia. Si hacemos la suposición de que la solución teórica $x(t)$ tiene derivadas continuas de orden suficientemente alto, entonces, para métodos explícitos e implícitos, deducimos de (2.27) que,

$$x(t_{i+k})-x_{i+k}=C_{p+1}h^{p+1}x^{(p+1)}(t)+O(h^{p+2}),$$

donde p es el orden del método y el término $C_{p+1}h^{p+1}x^{(p+1)}(t)$ es llamado constante de error. Si no se hace la suposición de que $x_{i+j}=x(t_{i+j})$, entonces $e_{i+k}=x(t_{i+k})-x_{i+k}$ corresponde al *error global de truncamiento* que es el error que involucra todos los errores de truncamiento hechos en cada aplicación del método.

Definición 2.12. Se dice que el método lineal multipaso (2.20) es consistente si tiene orden $p\geq 1$.

De (2.26) se sigue que el método (2.20) es consistente si y solo si

$$\sum_{j=0}^{k}\alpha_j=0;\qquad\sum_{j=0}^{k}j\alpha_j=\sum_{j=0}^{k}\beta_j. \tag{2.28}$$

La primera condición es necesaria para cualquier método en el que $i\to\infty$ y $h\to 0$ implique que $x_{i+j}\to x(t)$, donde $x(t)$ es una función arbitraria. La segunda condición quita la arbitrariedad a $x(t)$ ya que garantiza que ésta satisfaga a la ecuación diferencial, es decir asegura que $\dot{x}(t)=f(t,x(t))$.

Puede verse que todo método multipaso lineal convergente es consistente, sin embargo el que sea consistente no garantiza la convergencia.

Introduciremos ahora la noción de *primero* y *segundo polinomios característicos* del método multipaso (2.20), definidos por $\rho(\zeta)$ y $\sigma(\zeta)$ respectivamente, donde

$$\rho(\zeta) = \sum_{j=0}^{k} \alpha_j \zeta^j,$$
$$\sigma(\zeta) = \sum_{j=0}^{k} \beta_j \zeta^j. \tag{2.29}$$

De (2.28) se sigue que $\rho(1) = 0$, $\dot{\rho}(1) = \sigma(1)$. Así, para un método consistente, el primer polinomio característico $\rho(\zeta)$ siempre tiene una raíz en +1 que será llamada la raíz principal ζ_1.

Definición 2.13. Se dice que el método multipaso (2.20) es *cero-estable* si ninguna raíz del primer polinomio característico $\rho(\zeta)$ tiene módulo mayor que uno, y si cada raíz con módulo uno es simple.

Puede observarse que para un método lineal de un paso, el polinomio característico $\rho(\zeta)$ tiene grado uno, y si el método es consistente, la única raíz ζ_1, es +1. Así, un método consistente de un paso es necesariamente *cero-estable*. Por otro lado, es evidente que $\sigma(1)$ no puede ser cero, ya que si esto ocurriera, por (2.28) se tendría

$$\rho(1) = 0 = \dot{\rho}(1),$$

lo cual implica que $\rho(\zeta)$ tiene una raíz doble en +1, contrario a la definición de cero-estabilidad.

Podemos ahora enunciar el teorema fundamental de Dahlquist para métodos lineales multipaso.

Teorema 2.4. Las condiciones necesarias y suficientes para que un método lineal multipaso sea convergente son que sea consistente y cero-estable.

Cualitativamente hablando, la consistencia controla la magnitud del error local de truncamiento cometido en cada etapa de los cálculos, mientras que la cero-estabilidad controla la manera en la cual el error se propaga a medida que se efectúan los cálculos. Ambas son esenciales para que un método sea convergente. La convergencia es una propiedad mínima que un método aceptable debe poseer.

Algunos métodos multipaso

Para un número k dado de pasos, usualmente estamos interesados en escoger coeficientes α_j, β_j para los cuales el método (2.20) tenga un orden razonablemente alto (si no es que el más alto posible). Al buscar un orden alto, automáticamente se satisface la condición de consistencia pero encontramos un fuerte obstáculo en la condición de cero-estabilidad.

El método general de k pasos, tiene $2k+2$ coeficientes α_j, β_j, $j=0,1,...,k$, de los cuales uno, α_k tiene el valor de +1. Entonces, hay $2k+1$ parámetros libres y este número puede ser reducido a $2k$ si requerimos que el método sea explícito. De (2.26) es claro que, para que el método tenga orden p, dichos parámetros deben satisfacer $p+1$ condiciones. Así, el orden más alto que podemos esperar de un método de k pasos es $2k$, si el método es implícito; $2k-1$ si éste es explícito. Sin embargo, esos órdenes máximos no pueden obtenerse en general sin violar la condición de cero-estabilidad, como se expresa en el siguiente teorema.

Teorema 2.5. Ningún método multipaso lineal cero-estable puede tener un orden mayor de $k+1$ si k es impar, o mayor de $k+2$ si k es par.

A continuación se dan algunos métodos multipaso en los que se puede notar dicha situación.

Métodos Explícitos

$k=1$:

$$\alpha_1 = 1, \qquad \alpha_0 = -1, \qquad \beta_0 = 1, \qquad p = 1; \qquad C_{p+1} = \tfrac{1}{2}.$$

$k=2$:

$$\alpha_2 = 1, \qquad \alpha_1 = -1-a, \qquad \beta_1 = \tfrac{1}{2}(3-a), \qquad \alpha_0 = a, \qquad \beta_0 = -\tfrac{1}{2}(1+a), \qquad p = 2; \qquad C_{p+1} = \tfrac{1}{12}(5+a).$$

No existen valores para a con los que se logre un orden mayor a 2 y que el método sea cero-estable.

k=3:

$$\alpha_3 = 1,$$
$$\alpha_2 = -1-a, \qquad \beta_2 = \tfrac{1}{12}(23-5a-b)$$
$$\alpha_1 = a+b, \qquad \beta_1 = \tfrac{1}{3}(-1-2a+2b),$$
$$\alpha_0 = -b, \qquad \beta_0 = \tfrac{1}{12}(5+a+5b),$$
$$p=3; \qquad C_{p+1} = \tfrac{1}{24}(9+a+b).$$

No existen valores para a y b con los que se logre un orden mayor a 3 y que el método sea cero-estable.

k=4:

$$\alpha_4 = 1,$$
$$\alpha_3 = -1-a, \qquad \beta_3 = \tfrac{1}{24}(55-9a-b-c),$$
$$\alpha_2 = a+b, \qquad \beta_2 = \tfrac{1}{24}(-59-19a+13b+5c),$$
$$\alpha_1 = -b-c, \qquad \beta_1 = \tfrac{1}{24}(37+5a+13b-19c),$$
$$\alpha_0 = c, \qquad \beta_0 = \tfrac{1}{24}(-9-a-b-9c),$$
$$p=4; \qquad C_{p+1} = \tfrac{1}{720}(251+19a+11b+19c).$$

No existen valores para a, b y c con los que se logre un orden mayor a 4 y que el método sea cero-estable.

Métodos Implícitos

k=1:

$$\alpha_1 = 1, \qquad \beta_1 = \tfrac{1}{2}$$
$$\alpha_0 = -1, \qquad \beta_0 = \tfrac{1}{2},$$
$$p=2; \qquad C_{p+1} = -\tfrac{1}{12}.$$

k=2:

$$\alpha_2 = 1, \qquad \beta_2 = \tfrac{1}{12}(5+a),$$
$$\alpha_1 = -1-a, \qquad \beta_1 = \tfrac{2}{3}(1-a),$$
$$\alpha_0 = a, \qquad \beta_0 = \tfrac{1}{12}(-1-5a),$$

Si $a \neq -1$, entonces $p=3; \qquad C_{p+1} = -\tfrac{1}{24}(1+a).$

Si $a=-1$, entonces $p=4$; $\quad C_{p+1}=-\frac{1}{90}$.

k=3:

$$\alpha_3=1, \qquad \beta_3=\tfrac{1}{24}(9+a+b)$$
$$\alpha_2=-1-a, \qquad \beta_2=\tfrac{1}{24}(19-13a-5b)$$
$$\alpha_1=a+b, \qquad \beta_1=\tfrac{1}{24}(-5-13a+19b),$$
$$\alpha_0=-b, \qquad \beta_0=\tfrac{1}{24}(1+a+9b),$$
$$p=4; \qquad C_{p+1}=-\tfrac{1}{720}(19+11a+19b).$$

No existen valores para a y b con los que se logre un orden mayor a 4 y que el método sea cero-estable.

k=4:

$$\alpha_4=1, \qquad \beta_4=\tfrac{1}{720}(251+19a+11b+19c),$$
$$\alpha_3=-1-a, \qquad \beta_3=\tfrac{1}{360}(323-173a-37b-53c),$$
$$\alpha_2=a+b, \qquad \beta_2=\tfrac{1}{30}(-11-19a+19b+11c),$$
$$\alpha_1=-b-c, \qquad \beta_1=\tfrac{1}{360}(53+37a+173b-323c),$$
$$\alpha_0=c, \qquad \beta_0=\tfrac{1}{720}(-19-11a-19b-251c).$$

Si $27+11a+11b+27c\neq 0$, entonces

$p=5$; $\quad C_{p+1}=-\frac{1}{1440}(27+11a+11b+27c)$.

Si $27+11a+11b+27c=0$, entonces

$p=6$; $\quad C_{p+1}=-\frac{1}{15120}(74+10a-10b+74c)$.

No existen valores para a y b y c con los que se logre un orden mayor que 6 y que el método sea cero-estable.

De acuerdo a los valores de los coeficientes del polinomio característico $\rho(\zeta)$, existen "familias" de métodos multipaso lineales. Por ejemplo, aquellos métodos para los cuales $\rho(\zeta)=\zeta^k-\zeta^{k-1}$ son llamados *Métodos de Adams*. Éstos tienen la propiedad de que sus raíces no principales se encuentran en el origen; es decir, son cero-estables. Los métodos de Adams que son explícitos son llamados *Métodos de Adams-Bashfort*, mientras que los implícitos se conocen como *Métodos de Adams-Moulton*. Los métodos explícitos para los cuales $\rho(\zeta)=\zeta^k-\zeta^{k-2}$, son llamados

Métodos de Nyström, y los métodos implícitos con la misma forma para $\rho(\zeta)$ son los *Métodos Generalizados de Milne-Simpson*, ambas familias son cero-estables, ya que solamente tienen una raiz no principal en -1 y el resto en el origen.

Cálculo de valores iniciales

Cuando se elige algún método multipaso lineal convergente, surgen dos preguntas básicas.

i) En caso de que el número de pasos *sea mayor que 1*, ¿Cómo podemos encontrar los valores iniciales necesarios x_μ, $\mu=1,2,...,k-1$?

ii) En caso de que el método sea implícito, ¿Qué valor tomamos para el valor implícito x_{i+k}?

En esta sección se dará respuesta a la primera pregunta; la sección siguiente tiene que ver con la segunda pregunta.

Para encontrar los valores iniciales x_μ, $\mu=1,2,...,k-1$ es deseable que el método utilizado tenga al menos una precisión tan alta como la precisión local del método principal; ésto es, se requiere que

$$x_\mu - x(t_\mu) = O(h^{p+1}), \qquad \mu = 1,2,...k-1. \tag{2.30}$$

Además, tal método no debe requerir más valores iniciales que x_0; esto es, debe ser de un paso. Para asegurarnos que el error local de truncamiento sea el requerido, se recomienda utilizar algún método de Taylor. Sin embargo, dada la dificultad que representa el obtener las derivadas de f en dichos métodos, podrían utilizarse alternativamente los métodos de runge-Kutta.

Métodos Predictores-Correctores

Antes de responder a la segunda pregunta hecha en la sección anterior, es natural preguntarnos por qué es deseable utilizar métodos implícitos. Parecería suficiente con los métodos explícitos que por otra parte no tienen el problema de requerir el cálculo del valor implícito. Es claro que para un número k de pasos, el orden más alto alcanzable para un método cero-estable, es más alto para un método implícito que para uno explícito y en realidad no hay dificultad en calcular el valor implícito.

Supongamos que intentamos utilizar un método implícito lineal de k pasos para resolver el problema de valor inicial (2.2). Para cada x_{i+k} debemos resolver la ecuación

$$x_{i+k} + \sum_{j=0}^{k} \alpha_j x_{i+j} = h\beta_k f(t_{i+k}, x_{i+k}) + h\sum_{j=0}^{k-1} \beta_j f_{i+j}, \qquad (2.31)$$

donde x_{i+j}, f_{i+j}; $j=0,1,\ldots,k-1$, son conocidos. Puede demostrarse que existe una solución única para x_{i+k} y puede ser aproximada, tan cerca como se desee, por la iteración

$$x_{i+k}^{[s+1]} + \sum_{j=0}^{k} \alpha_j x_{i+j} = h\beta_k f(t_{i+k}, x_{i+k}^{[s]}) + h\sum_{j=0}^{k-1} \beta_j f_{i+j}, \qquad (2.32)$$

donde $x_{i+j}^{[0]}$ es arbitraria, siempre y cuando se satisfaga la condición

$$h < \frac{1}{L|\beta_k|}, \qquad \text{con } L \text{ la constante de Lipschitz de } f \text{ con respecto a } x.$$

Claramente, cada paso de la iteración (2.32) involucra una evaluación de $f(t_{i+k}, x_{i+k}^{[s]})$. Así estamos interesados en disminuir lo más posible el número de veces que dicha iteración es aplicada, por lo que es deseable que $x_{i+k}^{[0]}$ sea lo más preciso posible. Para ello utilizaremos un método explícito para estimar una "predicción" de x_{i+k} que tomaremos como el valor inicial para $x_{i+k}^{[0]}$. El método explícito es llamado *predictor* y el implícito *corrector*.

Una vez obtenido $x_{i+k}^{[0]}$, podemos continuar iterando mediante la fórmula (2.32) hasta que las iteraciones hayan convergido (en la práctica se utiliza el criterio $\left|x_{i+k}^{[s+1]} - x_{i+k}^{[s]}\right| < \varepsilon$, donde ε es una tolerancia prefijada del orden del error local de truncamiento). Entonces $x_{i+k}^{[s+1]}$ es una aproximación aceptable para la solución exacta x_{i+k} del método (2.31). Dado que a cada iteración le corresponde una aplicación del método corrector, a este modo de operación de los métodos Predictores-Correctores le llamaremos *corregir para converger*.

Los métodos Predictores-Correctores más usuales son:

Método de Adams-Bashfort-Moulton

Predictor: $x_{i+4} - x_{i+3} = \frac{h}{24}(55f_{i+3} - 59f_{i+2} + 37f_{i+1} - 9f_i)$,

Corrector: $x_{i+4} - x_{i+3} = \frac{h}{24}(9f_{i+4} + 19f_{i+3} - 5f_{i+2} + f_{i+1})$.

Estimación del error: $C_5 h^5 x^{(5)}(t_i) \approx -\frac{19}{270}(x_{i+4}^{[1]} - x_{i+4}^{[0]})$.

Método de Crane y Klopfenstein

Predictor:

$$x_{i+4} - 1.547652x_{i+3} + 1.867503x_{i+2} - 2.017204x_{i+1} + 0.697353x_i = h(2.002247f_{i+3} - 2.031690f_{i+2} + 1.818609f_{i+1} - 0.714320f_i),$$

Corrector: $x_{i+4} - x_{i+3} = \frac{h}{24}(9f_{i+4} + 19f_{i+3} - 5f_{i+2} + f_{i+1})$.

Estimación del error: $C_5 h^5 x^{(5)}(t_i) = \frac{1}{16.21966}(x_{i+4}^{[1]} - x_{i+4}^{[0]})$

Método de Klopfenstein y Millman

Predictor:

$$x_{i+4} + 0.29x_{i+3} + 15.39x_{i+2} - 12.13x_{i+1} - 4.55x_i = h(2.27f_{i+3} + 6.65f_{i+2} + 13.91f_{i+1} - 0.69f_i),$$

Corrector: $x_{i+4} - x_{i+3} = \frac{h}{24}(9f_{i+4} + 19f_{i+3} - 5f_{i+2} + f_{i+1})$.

Estimación del error: $C_5 h^5 x^{(5)}(t_i) = -\frac{1}{18.0274}(x_{i+4}^{[1]} - x_{i+4}^{[0]})$.

BIBLIOGRAFÍA

[1] C. WILLIAM GEAR, *Numerical Initial Value Problems in Ordinary Differential Equations,* Prentice-Hall Inc., Engleewood Cliffs, N.J., 1971, 255pp.

[2] E. HAIRER, S.P. NORSETT, G. WANNER, *Solving Ordinary Differential Equations I, Nonstiff Problems,* Springer-Verlag, Berlín, 1987, p. 165-174.

[3] LAMBERT, *Computational Methods in Ordinary Differential Equations,* John Wiley&Sons, Great Britain, 1983.

[4] RICHARD L BURDEN, J. DOUGLAS FAIRES, *Análisis Numérico,* Grupo Editorial Iberoamética, México D.F., 1985, p. 13-38, 222-316.

[5] WILLIAM E. BOYCE, RICHARD C. DIPRIMA, *Ecuaciones Diferenciales y Problemas con Valores en la Frontera,* Noriega Limusa, México D.F, 1991, 743 pp.

[6] WILLIAM H. PRESS, BRIAN P. FLANNERY, SAUL A. TEUKOLSKY, WILLIAM T. VETTERLING, *Numerical Recipes in C,* Cambridge University Press, New York, 1989, 566-592.

[7] SHOICHIRO NAKAMURA, *Numerical Analysis,* Cambridge University Press, New York, 1990.

Capítulo 3

LOCALIZACIÓN DE PUNTOS DE EQUILIBRIO, DETERMINACIÓN DE ESTABILIDAD Y TRAZO DE CEROCLINAS

INTRODUCCIÓN

Para la caracterización del comportamiento de un sistema dinámico existen dos importantes tipos de soluciones: las soluciones periódicas y las soluciones de equilibrio. Ello se debe a que el comportamiento de todas las demás soluciones gira en torno a éstas, es decir, o todas las soluciones cercanas a un punto crítico u órbita periódica convergen a éstas con el transcurrir del tiempo, o todas son repelidas por éste. Para saber la dirección en la que se alejan o convergen hacia una solución especial, es muy útil el conocimiento de las ceroclinas de un sistema. Para saber si las demás soluciones se alejan o convergen a las soluciones especiales se hace un análisis de estabilidad. El que sean estables o inestables implica que atraen o repelen a las demás soluciones respectivamente.

La localización de soluciones de equilibrio implica encontrar el valor donde se anula una función vectorial. Una vez resuelta dicha función, se desea conocer la estabilidad del punto encontrado, para lo cual se requiere obtener los valores característicos de la matriz jacobiana evaluada en dicho punto. De la misma forma, el cálculo de ceroclinas implica encontrar el valor en donde se anula una función real.

Resolver analíticamente los problemas mencionados es sumamente complicado para el caso no lineal, de hecho a veces es imposible. De allí la importancia de utilizar métodos numéricos para la localización de los ceros de funciones no lineales multidimensionales y unidimensionales. En este capítulo se discuten algunos de los métodos numéricos más relevantes para dichos fines.

3.1 El Problema General de Solución de Ecuaciones

Una de las tareas básicas en el área de las matemáticas es la solución de ecuaciones. De hecho es un problema con el que se tiene contacto casi desde los primeros niveles de escolaridad. Claro está, la complejidad es cada vez mayor hasta que nuestros recursos algebraicos se agotan al estar frente a frente con sistemas no lineales y nos vemos inducidos a aplicar métodos numéricos para su solución. Para plantear los métodos numéricos para solución de ecuaciones se parte de la premisa de que se trata de resolver la ecuación

$$f(x)=0 \tag{3.1}$$

para la cual se dirá que se busca su raíz o raíces. Nótese que cualquier ecuación puede expresarse en la forma de (3.1), de manera que ésta se considera general.

Cuando se tiene solamente una variable independiente el problema es *unidimensional*. Para más de una variable independiente se tendrán que satisfacer varias ecuaciones simultáneamente. En este caso, podría ocurrir que el sistema no tuviera solución, que no tuviera solución real ó que tuviera más de una solución (un espacio de soluciones). El que no exista una solución real, de una u otra forma puede ser superado intrumentando algún método adecuado para trabajar con números complejos. Por otra parte, si se tiene un espacio de soluciones, éste no podrá ser determinado con precisión mediante los métodos numéricos convencionales.

En notación vectorial, nuestro problema quedará expresado como

$$\mathbf{f}(\mathbf{x})=0 \tag{3.2}$$

donde $\mathbf{f}:\Re^n \rightarrow \Re^n$ y $\mathbf{x} \in \Re^n$. Evidentemente, encontrar raíces es mucho más sencillo en el caso unidimensional que en el *n*-dimensional. De hecho, los métodos de solución que se utilizan en uno y otro caso son un tanto diferentes en cuanto a su forma de operación y justificación.

Excepto en el caso lineal, la búsqueda de raíces invariablemente se hace mediante iteraciones. Iniciando de una solución que se propone, un algoritmo irá mejorando la aproximación hasta que se satisfaga algún criterio de convergencia. Cuando las funciones tienen primera derivada continúa, los buenos algoritmos siempre convergen cuando la aproximación inicial es "suficientemente buena". Es muy importante

recalcar que una buena aproximación inicial depende de análisis matemático y no de un algoritmo computacional.

3.2 Métodos de Solución para Funciones Unidimensionales

Método de Newton-Raphson

El método de Newton-Raphson, también conocido como método de Newton, es uno de los más usuales y poderosos para la solución de problemas de búsqueda de raíces. Geométricamente, la fórmula de Newton-Raphson consiste en extender la línea tangente a la función para un punto x_i hasta que cruce con el eje x y, entonces fijar la siguiente aproximación x_{i+1} como la abscisa en la cual se produce el cruce (Figura 3.1). Algebraicamente, el método se deriva de la expansión de f en series de Taylor como se explica a continuación.

Supóngase que la función f es dos veces continuamente diferenciable en el intervalo $[a,b]$, o sea, $f \in C^2[a,b]$. Sea $\bar{x} \in [a,b]$ una aproximación a p tal que $\dot{f}(\bar{x}) \neq 0$ y $|\bar{x} - p|$ es "pequeño". Considere el polinomio de Taylor de primer grado para $f(x)$ alrededor de $\bar{x}$,

$$f(x) = f(\bar{x}) + \dot{f}(\bar{x})(x-\bar{x}) + \frac{\ddot{f}(\xi(x))}{2}(x-\bar{x})^2 + \ldots \tag{3.3}$$

donde $\xi(x)$ está entre x y $\bar{x}$. Como $f(p)=0$, la ecuación (3.3), con $x=p$, nos da

$$0 = f(\bar{x}) + \dot{f}(\bar{x})(p-\bar{x}) + \frac{\ddot{f}(\xi(x))}{2}(p-\bar{x})^2 + \ldots \tag{3.4}$$

El método de Newton se deriva suponiendo que el término que contiene a $(p-\bar{x})$ es despreciable y que

$$0 = f(\bar{x}) + \dot{f}(\bar{x})(p-\bar{x}). \tag{3.5}$$

Despejar p de ésta ecuación da

$$p \approx \bar{x} - \frac{f(\bar{x})}{\dot{f}(\bar{x})}, \tag{3.6}$$

que debe ser una mejor aproximación a p que $\bar{x}$. El método de Newton consiste en generar la sucesión $\{p_n\}$ definida por

$$p_n = p_{n-1} - \frac{f(p_{n-1})}{\dot{f}(p_{n-1})}, \; n \geq 1 \tag{3.7}$$

Como criterios de paro se tienen tres opciones:

$$i) \quad |p_N - p_{N-1}| < \varepsilon,$$

$$ii) \quad \frac{|p_N - p_{N-1}|}{|p_N|} < \varepsilon, \qquad p_N \neq 0,$$

$$iii) \quad |f(p_N)| < \varepsilon.$$

Seleccione una tolerancia $\varepsilon > 0$ y construya

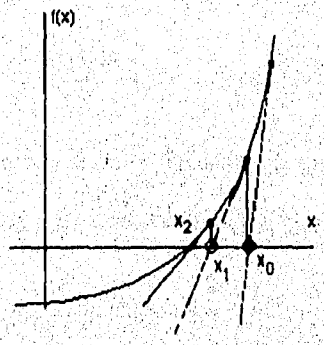


Figura 3.1. El Método de Newton extrapola la derivada local para encontrar la siguiente estimación de la raíz. En este ejemplo el método trabaja muy bien y converge cuadráticamente.

Analicemos ahora algunos casos patológicos que podrían acabar con todo el poder del método de Newton. Lejos de la raíz, donde los términos de orden más alto en las series de Taylor son importantes, el método de Newton podría arrojar resultados

imprecisos. Por ejemplo, la aproximación inicial para la raíz podría estar tan alejada del valor real que existiera un máximo o un mínimo local de la función en el intervalo de búsqueda (figura 3.2). Si una iteración ubica la aproximación calculada cerca del extremo local, de tal suerte que la primera tienda a anularse, entonces el método de Newton disparará su solución al infinito. La figura 3.3 muestra otro caso patológico. Podemos entonces decir que, como la mayoría de las más poderosas herramientas, el Newton-Raphson puede ser un total fracaso y de hecho, destructivo, si se usa en circunstancias inapropiadas. Lo único que podemos recomendar es ser muy cuidadosos en la elección de la aproximación inicial. De hecho puede utilizarse un método menos poderoso como el popular de BISECCIÓN y después aplicar Newton-Raphson.

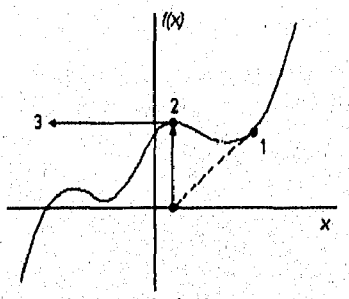


Figura 3.2. Caso patológico donde el método de Newton encuentra un extremo local y se dispara.

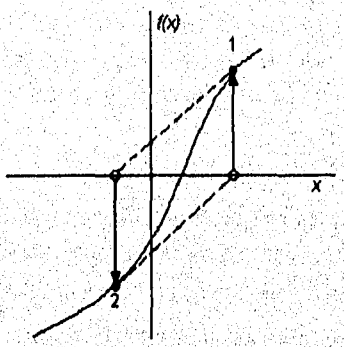


Figura 3.3. Caso patológico en el que el método de Newton entra en un ciclo no convergente. Este comportamiento se encuentra a menudo cuando la función *f* es obtenida completamente o en parte por tablas de interpolación. Con una mejor aproximación inicial, el método podría tener éxito.

Se ha dicho que el método de Newton-Raphson es uno de los más podersos, la pregunta es ¿en qué radica ese poder?. La respuesta está sustentada en su rapidez de convergencia: con una distancia "corta" ε hacia x, la función y su derivada son aproximadamente:

$$\begin{aligned} f(x+\varepsilon) &= f(x)+\varepsilon\dot{f}(x)+\varepsilon^2\frac{\ddot{f}(x)}{2}+\cdots, \\ \dot{f}(x+\varepsilon) &= \dot{f}(x)+\varepsilon\ddot{f}(x)+\cdots \end{aligned} \tag{3.8}$$

Por la fórmula de Newton-Raphson,

$$\varepsilon_{i+1} = \varepsilon_i - \frac{f(x_i)}{\dot{f}(x_i)}. \tag{3.9}$$

Donde la solución aproximada difiere de la real por ε_i. Ahora podemos utilizar (3.8) para expresar $f(x_i)$, $\dot{f}(x_i)$ de (3.9) en términos de ε_i y derivadas en la raíz. El resultado es la relación de concurrencia

$$\varepsilon_{i+1} = -\varepsilon^2\frac{\ddot{f}(x)}{2\dot{f}(x)}. \tag{3.10}$$

Esta ecuación nos dice que el método de Newton converge *cuadráticamente*. Cerca de la raíz, el número de dígitos significativos aproximadamente se duplica por cada iteración. Esta fuerte propiedad de convergencia hace del Newton-Raphson, el método de elección para casi cualquier función cuyas derivadas puedan ser evaluadas eficientemente, y cuya derivada sea continua en una vecindad de la raíz.

Método de la Secante

El método de Newton es una técnica extremadamente poderosa, pero tiene una dificultad grande: la necesidad de saber el valor de la derivada de f en cada aproximación. Frecuentemente ocurre que $\dot{f}(x)$ es mucho más complicada y necesita de más operaciones aritméticas para su cálculo que $f(x)$.

Para evitar el problema de la evaluación de la derivada en el método de Newton, podemos derivar una pequeña variación de éste. Por definición,

$$\dot{f}(p_{n-1}) = \lim_{x \to p_{n-1}} \frac{f(x) - f(p_{n-1})}{x - p_{n-1}}.$$

Tomando $x=p_{n-2}$.

$$\dot{f}(p_{n-1}) \approx \frac{f(p_{n-2}) - f(p_{n-1})}{p_{n-2} - p_{n-1}} = \frac{f(p_{n-1}) - f(p_{n-2})}{p_{n-1} - p_{n-2}}.$$

Usando esta aproximación para $f(p_{n-1})$ en la fórmula de Newton da

$$p_n = p_{n-1} - \frac{f(p_{n-1})(p_{n-1} - p_{n-2})}{f(p_{n-1}) - f(p_{n-2})} \tag{3.11}$$

que es lo que se conoce como *Método de la Secante*. Su nombre, se debe a que si se hace una derivación gráfica similar a la del Método de Newton Raphson pero tomando las rectas secantes en vez de las tangentes, se obtiene la expresión (3.4) que define al método. Este hecho se muestra en la figura (3.4)

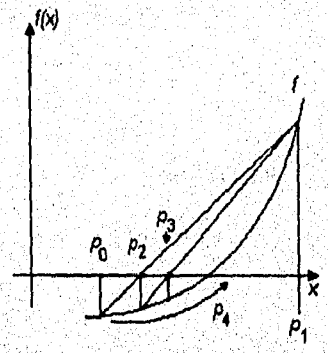


Figura 3.4. Ilustración gráfica del método de la secante

3.3 Métodos para Obtener Raíces de Polinomios

Aquí se presentarán algunos métodos para encontrar raíces de polinomios. Por un polinomio de grado *n* entenderemos una función de la forma

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0, \tag{3.12}$$

donde las a_i, llamadas los coeficientes de P, son constantes y $a_n \neq 0$. La función cero, $P(x)=0$ para todos los valores de x, se considera un polinomio pero no se le asigna ningún grado.

El *Teorema Fundamental del Álgebra* garantiza la existencia de cuando menos una raíz para polinomios de grado mayor o igual que uno, haciendo hincapié en que dicha raíz bien pudiera ser compleja. De la misma forma, otros resultados afirman que los ceros de un polinomio son únicos y que si cada cero x_j, es contado tantas veces como su multiplicidad m_i, entonces un polinomio de grado n tiene exactamente n ceros. Trabajaremos entonces bajo tales premisas y, para el lector que tenga curiosidad acerca de tales resultados se le recomienda consultar las referencias [4] y [5] en las secciones correspondientes a búsqueda de raíces de polinomios y teoría de polinomios respectivamente.

En principio, para encontrar las raíces de un polinomio podrían aplicarse los métodos antes discutidos ya que el problema es básicamente el mismo y la función (3.12) puede ser expresada en la forma general (3.1). Sin embargo, la forma de un polinomio permite optimizar en cálculos si en lugar de evaluar convencionalmente un polinomio, aplicamos el método de Horner cuyo funcionamiento abordaremos a continuación.

Método de Horner

Teorema 3.1 (Método de Horner). Sea

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 \qquad \text{y} \qquad b_n = a_n,$$

si

$$b_k = a_k + b_{k+1} x_0 \quad \text{para} \quad k = n-1, n-2, \ldots, 1, 0,$$

entonces $b_0 = P(x_0)$. Además, si

$$Q(x) = b_n x^{n-1} + b_{n-1} x^{n-2} + \cdots + b_2 x + b_1,$$

entonces

$$P(x) = (x - x_0) Q(x) + b_0, \tag{3.13}$$

Demostración

Por la definición de $Q(x)$,

$$\begin{aligned}(x-x_0)Q(x)+b_0 &= (x-x_0)\left(b_n x^{n-1}+\cdots+b_2 x+b_1\right)+b_0\\ &= \left(b_n x^n+\cdots+b_2 x^2+b_1 x\right)\\ &\quad -\left(b_n x_0 x^{n-1}+\cdots+b_2 x_0 x+b_1 x_0\right)+b_0\\ &= b_n x^n+\left(b_{n-1}-b_n x_0\right)x^{n-1}+\cdots+\left(b_1-b_2 x_0\right)x+\left(b_0-b_1 x_0\right).\end{aligned}$$

Por las hipótesis del teorema, $b_n=a_n$ y $b_k-b_{k+1}x_0=a_k$, por lo tanto

$$(x-x_0)Q(x)+b_0=P(x) \quad \text{y} \quad b_0=P(x_0).$$

Ejemplo 3.1. Evaluemos el polinomio $P(x)=2x^4-3x^2+3x-4$ en $x_0=-2$.

$b_4=2$, $b_3=2(-2)+0=-4$,

$b_2=(-4)(-2)-3=5$, $b_1=5(-2)+3=-7$

y finalmente

$P(-2)-b_0=(-7)(-2)-4=10$.

Además, el teorema nos dice que

$$P(x)=(x+2)\left(2x^3-4x^2+5x-7\right)+10$$

Una ventaja adicional al usar el procedimiento de Horner es que, como

$$P(x)=(x-x_0)Q(x)+b_0,$$

donde

$$Q(x)=b_n x^{n-1}+b_{n-1}x^{n-2}+\cdots+b_2 x+b_1,$$

diferenciando con respecto a x da

$$P'(x)=Q(x)+(x-x_0)Q'(x) \quad \text{y}$$

$$\dot{P}(x_0) = Q(x_0)$$

Entonces, cuando se use el método de Newton-Raphson para encontrar un cero aproximado de un polinomio P, ambos P y $\dot{P}$ pueden ser evaluados de esta manera.

Método de Deflación

El lector habrá notado que el polinomio denotado por Q depende de la aproximación usada y cambia de iteración a iteración.

Si la N-ésima iteración, x_N, en el procedimiento de Newton-Raphson es un cero aproximado de P, entonces

$$P(x) = (x - x_N)Q(x) + b_0 = (x - x_N)Q(x) + P(x_N) \approx (x - x_N)Q(x).$$

Por tanto, $x - x_N$ es un factor aproximado de $P(x)$. Tomando $\hat{x}_1 = x_N$ como un cero aproximado de P y $Q_1(x)$ como el factor aproximado,

$$P(x) \approx (x - \hat{x}_1)Q_1(x),$$

podemos encontrar un segundo cero aproximado de P aplicando el procedimiento de Newton-Raphson a $Q_1(x)$. Si P es un polinomio de grado n con n ceros reales, este procedimiento aplicado repetidamente resultará eventualmente en $(n-2)$ ceros aproximados de P y en un factor cuadrático aproximado $Q_{n-2}(x)$. A este nivel,

$$Q_{n-2}(x) = 0$$

puede resolverse por la fórmula cuadrática para encontrar los dos últimos ceros aproximados de P. A esta forma de operar se le conoce como *Método de Deflación* que aún cuando puede ser usado para encontrar ceros aproximados de muchos polinomios, depende del uso repetido de aproximaciones, por lo que en ocasiones puede llevar a resultados muy imprecisos. Más explícitamente, cuando obtenemos los ceros aproximados de P, el procedimiento de Newton-Raphson se usa en el polinomio reducido Q_k, que es el polinomio con la propiedad de que

$$P(x) \approx (x - \hat{x}_1)(x - \hat{x}_2)\cdots(x - \hat{x}_k)Q_k(x).$$

Un cero aproximado $\hat{x}_{k+1}$ de Q_k generalmente no aproximará a una raíz de $P(x)=0$ tan bien como una raíz de $Q_k(x)=0$. La imprecisión usualmente se incrementará

conforme crezca k. Una manera de eliminar esta dificultad consiste en usar las ecuaciones reducidas, esto es, los factores aproximados del polinomio original P, para encontrar aproximaciones, $\hat{x}_2, \hat{x}_3, \ldots, \hat{x}_k$ a sus ceros y luego mejorarlas aplicándoles el método de Newton-Raphson. Dicha forma de proceder tendrá dificultades en dos casos:

i) Existen raíces que están muy cerca una de otra. En tal caso el método podría converger a una(s) de ellas y jamás ir hacia otras.

ii) Existen raíces complejas. Si la aproximación inicial al usar el método de Newton es un número real, todas las aproximaciones subsecuentes serán también números reales. Una manera de atacar esta dificultad es empezar con aproximaciones iniciales complejas y hacer todos los cálculos usando aritmética compleja. Un enfoque diferente se basa en el siguiente teorema.

Teorema 3.2. Si $z=a+ib$ es un número complejo de multiplicidad m del polinomio P, entonces $\bar{z} = a - ib$ es también un cero de multiplicidad m del polinomio P y $(x^2-2ax+a^2+b^2)^m$ es un factor de P.

Se puede diseñar una división sintética que involucre polinomios cuadráticos para factorizar aproximadamente el polinomio, de tal manera que uno de los términos sea un polinomio cuadrático cuyas raíces complejas sean aproximaciones a las raíces del polinomio original.

En vez de proceder en esta dirección consideraremos un método presentado por primera vez por D.E, Müller en 1956. Esta técnica puede ser usada para cualquier problema de búsqueda de raíces, pero es particularmente útil para aproximar raíces de polinomios.

Método de Müller

El método de Müller es una generalización del método de la secante. El método de la secante empieza con dos aproximaciones iniciales x_0 y x_1 y determina la siguiente aproximación x_2 como la intersección del eje x con la recta que pasa por $\big(x_0, f(x_0)\big)$, $\big(x_1, f(x_1)\big)$ (Véase la figura 3.4). El método de Müller usa tres aproximaciones iniciales x_0, x_1 y x_2 y determina la siguiente aproximación x_3 considerando la intersección del eje x con la parábola que pasa por $\big(x_0, f(x_0)\big)$, $\big(x_1, f(x_1)\big)$ y $\big(x_2, f(x_2)\big)$ (Vea figura 3.5).

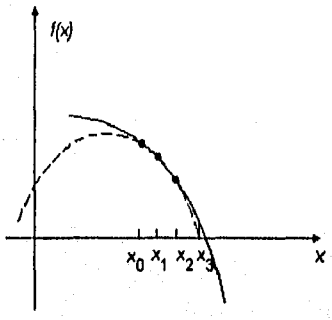


Figura 3.5. El método de Müller construye una parábola que intersecta a las tres aproximaciones iniciales y, la siguiente aproximación está sobre dicha parábola

La derivación del método de Müller comienza considerando el polinomio cuadrático

$$P(x) = a(x - x_2)^2 + b(x_0 - x_2) + c,$$

que pasa por $(x_0, f(x_0))$, $(x_1, f(x_1))$ y $(x_2, f(x_2))$. Las constantes a, b y c pueden determinarse de las condiciones

$$\begin{aligned} f(x_0) &= a(x_0 - x_2)^2 + b(x_0 - x_2) + c, \\ f(x_1) &= a(x_1 - x_2)^2 + b(x_1 - x_2) + c, \\ f(x_2) &= a \cdot 0^2 + b \cdot 0 + c \end{aligned}$$

las cuales dan

$$\begin{aligned} c &= f(x_2), \\ b &= \frac{(x_0 - x_2)^2[f(x_1) - f(x_2)] - (x_1 - x_2)^2[f(x_0) - f(x_2)]}{(x_0 - x_2)(x_1 - x_2)(x_0 - x_1)}, \\ c &= \frac{(x_1 - x_2)[f(x_0) - f(x_2)] - (x_0 - x_2)[f(x_1) - f(x_2)]}{(x_0 - x_2)(x_1 - x_2)(x_0 - x_1)}. \end{aligned} \tag{3.14}$$

Para determinar x_3, la raíz de P, aplicamos la fórmula cuadrática a P ligeramente transformada para evitar problemas del error de redondeo causados por la sustracción de números casi iguales

$$x_3 - x_2 = \frac{-2c}{b \pm \sqrt{b^2 - 4ac}}. \tag{3.15}$$

Esto da dos posibilidades para x_3, dependiendo del signo que precede al término radical en 3.15. En el método de Müller, el signo se elije para que coincida con el de b. Escogido de esta manera, el denominador será el más grande en magnitud y resultará en seleccionar a x_3 como la raíz de P más cercana a x_2. Así,

$$x_3 = x_2 - \frac{2c}{b + \text{signo}(b)\sqrt{b^2 - 4ac}}$$

donde a, b y c están dadas en (3.14).

Una vez que se determina x_3, el procedimiento se reinicia usando x_1, x_2 y x_3 en lugar de x_0, x_1 y x_2 para determinar la siguiente aproximación. El método continúa hasta que se obtiene una conclusión satisfactoria. Ya que el método involucra en cada paso al radical $\sqrt{b^2 - 4ac}$, el método aproximará raíces complejas cuando sea apropiado.

La importancia del método de Müller reside en que generalmente convergerá a la raíz del polinomio para cualquier elección de las aproximaciones iniciales. Se pueden construir problemas en los que no habrá convergencia para ciertas aproximaciones iniciales. Por ejemplo, si para alguna i, x_i, x_{i+1}, y x_{i+2} tienen la propiedad de que $f(x_i)=f(x_{i+1})=f(x_{i+2})$, entonces la ecuación cuadrática se reducirá a una función constante no cero y nunca cruzará al eje x aunque usualmente, éste no es el caso.

El método de Müller no es tan eficiente como el método de Newton, su orden de convergencia cerca de una raíz es aproximadamente α=1.84 comparado con el cuadrático, α=2, del de Newton, pero es mejor que el de la secante, cuyo orden es aproximadamente α=1.62.

Como recomendaciones, después de determinar una aproximación a la raíz mediante el método de deflación, úsese ya sea el método de Newton o el de Müller en el polinomio original. Esto asegurará que la raíz que se está aproximando es una solución de la ecuación verdadera y no de la obtenida por el método de deflación.

3.4 Métodos de Solución para Funciones Vectoriales

Los métodos que se discutirán en esta sección son aquellos que se usan para resolver ecuaciones de la forma (3.2). Cabe hacer hincapié en que no existen métodos generales que sean buenos para resolver el problema mencionado. Para explicar por qué, considérese el caso de dos dimensiones, en que se desea resolver

$$f(x,y)=0$$
$$g(x,y)=0$$

Las funciones f y g son dos funciones arbitrarias, cada una de las cuales tiene trayectorias en donde se anula la función y que además dividen al plano xy en regiones donde sus funciones respectivas son positivas o negativas. Dichas trayectorias frontera son de nuestro interés. Las soluciones que buscamos representan aquellos puntos que son comunes a las respectivas soluciones (vea figura 3.6). Desafortunadamente, las funciones f y g no tienen relación entre sí. No hay nada en común entre el comportamiento de g y el de f. Para encontrar todos los puntos en común, que son soluciones de nuestro sistema de ecuaciones no lineal, tendremos ni más ni menos que mapear absolutamente todo el espacio de solución de cada una. Note que las trayectorias solución pueden consistir de un número desconocido de curvas cerradas disjuntas. ¿Cómo se puede determinar cuando han sido encontradas todas esas curvas disjuntas?. En realidad no se tiene una respuesta satisfactoria.

Para problemas de más de dos dimensiones, necesitamos encontrar puntos mutuamente comunes a N hiperplanos de dimensión $N-1$ no-relacionados entre sí. En la práctica se verá que encontrar raíces se convierte casi imposible si no se tiene idea de "por donde andan los puntos". Por lo menos debería tenerse idea de si se espera una solución única y aproximadamente donde.

Una vez que se tiene identificada la vecindad donde se encuentra la solución, el problema se facilita considerablemente y es tiempo de utilizar un método numérico como el de Newton generalizado a n dimensiones. Este método, tiene la más alta tasa de convergencia, aunque al igual que la versión para dimensión 1 puede ser desastrozo si se aplica en las condiciones equivocadas.

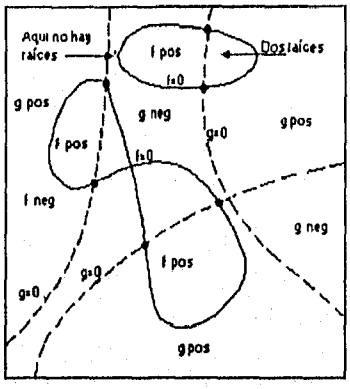


Figura 3.6 Solución de dos ecuaciones no lineales y dos incógnitas. Las curvas sólidas se refieren a f(x,y), las punteadas a g(x,y). Cada ecuación divide el plano xy en regiones positivas y negativas, acotadas por las curvas cero. Las soluciones deseadas son las intersecciones de estas curvas cero no-relacionadas. El número de soluciones es desconocido a *priori*

El Método de Newton para Sistemas de Ecuaciones no Lineales

Expresando la ecuación (3.2) en términos de sus componentes vectoriales, tenemos

$$f_i(\mathbf{x}) = 0, \qquad i=1,2,\ldots,n. \tag{3.16}$$

La expansión en series de Taylor para las funciones f_i alrededor de $\hat{\mathbf{x}}$ produce

$$f_i(\mathbf{x}) = f_i(\hat{\mathbf{x}}) + \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j}(\hat{\mathbf{x}})(x_j - \hat{x}_j) + O\left((x_j - \hat{x}_j)\hat{\mathbf{x}}^2\right).$$

Ignorando el término cuadrático de error y expresando en términos de vectores tenemos que

$$f_i(\mathbf{x}) \approx f_i(\hat{\mathbf{x}}) + \left(\frac{\partial f_i}{\partial x_1}(\hat{\mathbf{x}}), \frac{\partial f_i}{\partial x_2}(\hat{\mathbf{x}}), \ldots, \frac{\partial f_i}{\partial x_N}(\hat{\mathbf{x}})\right) \cdot (\mathbf{x} - \hat{\mathbf{x}})$$

Si $\mathbf{F}=(f_1, f_2, \ldots, f_n)$, entonces

$$\mathbf{F}(\mathbf{x}) \approx \mathbf{F}(\hat{\mathbf{x}}) + \mathbf{J}(\hat{\mathbf{x}})(\mathbf{x} - \hat{\mathbf{x}})$$

donde J es la matriz jacobiana de F. Supongamos que p es una raíz de F. Entonces,

$$\mathbf{F}(\mathbf{p}) = 0 \approx \mathbf{F}(\hat{\mathbf{x}}) + \mathbf{J}(\hat{\mathbf{x}})(\mathbf{p} - \hat{\mathbf{x}})$$

Despejando obtendremos

$$\mathbf{p} \approx \hat{\mathbf{x}} - \mathbf{J}(\hat{\mathbf{x}})^{-1}\mathbf{F}(\hat{\mathbf{x}})$$

Si se aplica iterativamente esta fórmula, cada valor calculado será una mejor aproximación al valor de la raíz p. El proceso del método inicia al seleccionar $\mathbf{x}^{(0)}$ generar, para $k \geq 1$,

$$\mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} - \mathbf{J}\left(\mathbf{x}^{(k-1)}\right)^{-1}\mathbf{F}\left(\mathbf{x}^{(k-1)}\right). \tag{3.17}$$

Nótese la necesidad de que la matriz $\mathbf{J}(\mathbf{x})$ sea no singular en cada iteración. Una debilidad clara de este método surge de la necesidad de invertir la matriz $\mathbf{J}(\mathbf{x})$ en cada paso. En la práctica, el método se realiza generalmente en una forma de dos pasos. Primero, se encuentra un vector y que satisfaga $\mathbf{J}(\mathbf{x}^{(k)})\mathbf{y} = -\mathbf{F}(\mathbf{x}^{(k)})$. Después de que se ha logrado ésto, la nueva aproximación $\mathbf{x}^{(k+1)}$, puede obtenerse sumando los vectores $\mathbf{x}^{(k)}$ y y.

Métodos Cuasi-Newton

Una deblidad importante del método de Newton para resolver sistemas de ecuaciones no lineales es la necesidad de calcular la matriz Jacobiana en cada iteración y de resolver un sistema lineal de $n \times n$ asociado a esta matriz, Para ilustrar la magnitud de esta debilidad, consideremos la cantidad de cálculos necesarios para llevar a cabo una iteración del método de Newton. La matriz Jacobiana asociada a un sistema de n ecuaciones no lineales escrita en la forma $\mathbf{F}(\mathbf{x})=0$ requiere de la determinación y evaluación de las n^2 derivadas parciales de las n componentes de **F**. En la mayoría de los casos, la evaluación exacta de las derivadas parciales es complicada y en muchas ocasiones imposible. Para sobrepasar esta dificultad se pueden usar aproximaciones de diferencia finita a las derivadas parciales. Por ejemplo,

$$\frac{\partial f_j}{\partial x_k}\left(\mathbf{x}^{(i)}\right) \approx \frac{f_j\left(\mathbf{x}^{(i)} + \mathbf{e}_k h\right) - f_j\left(\mathbf{x}^{(i)}\right)}{h} \tag{3.18}$$

donde h es "pequeña" en valor absoluto y e_k es el vector cuyo único elemento diferente de cero es uno en la k-ésima componente. Esta aproximación, sin embargo, requiere aún de la realización de por lo menos n^2 evaluaciones funcionales escalares para aproximar el Jacobiano y no reduce la cantidad de cálculos que es en general $O(n^3)$, para resolver el sistema lineal que contiene al Jacobiano aproximado. El esfuerzo computacional total para sólo una iteración del método de Newton es entonces de por lo menos n^2+n evaluaciones funcionales escalares (n^2 para la evaluación de la matriz Jacobiana y n para la evaluación de **F**) junto con $O(n^3)$, operaciones aritméticas para resolver el sistema lineal. Esta cantidad de esfuerzo computacional es prohibitiva excepto para valores relativamente pequeños de n y para funciones escalares fáciles de evaluar.

En esta sección se considera una generalización del método de la secante a sistemas de ecuaciones no lineales y, en particular, una técnica conocida como el *método de Broyden*. El método requiere solamente de n evaluaciones funcionales escalares por iteración y reduce también el número de cálculos aritméticos a $O(n^2)$. Es uno de los métodos conocidos como *renovaciones de secante de mínimo cambio* que producen los algoritmos llamados cuasi Newton. Estos métodos reemplazan la matriz Jacobiana en el método de Newton por una matriz de aproximación que se renueva en cada iteración. La desventaja de estos métodos consiste en que se pierde la convergencia cuadrática del método de Newton y se reemplaza por una convergencia llamada **superlineal**. La convergencia superlineal implica que

$$\lim_{i\to\infty} \frac{\left\|\mathbf{x}^{(i+1)}-\mathbf{p}\right\|}{\left\|\mathbf{x}^{(i)}-\mathbf{p}\right\|}=0,$$

donde **p** denota la solución a $\mathbf{F}(\mathbf{x})=0$ y $\mathbf{x}^{(i)}$, $\mathbf{x}^{(i+1)}$ son aproximaciones consecutivas. En la mayoría de las aplicaciones, la reducción a la convergencia superlineal es un cambio más que aceptable por la reducción en la cantidad de recursos de cómputo.

Una desventaja consiste en que, a diferencia del método de Newton, los métodos cuasi-Newton no son autocorregibles. El método de Newton, por ejemplo, generalmente corrige errores de redondeo mediante iteraciones sucesivas, en cambio, el método de Broyden no lo hace.

Supongamos que se da una aproximación inicial $\mathbf{x}^{(0)}$ a la solución $\mathbf{p}$ de $\mathbf{F}(\mathbf{x})=0$. Calculamos la siguiente aproximación $\mathbf{x}^{(1)}$ de la misma manera que en el método de Newton o, si es inconveniente determinar $\mathbf{J}(\mathbf{x}^{(0)})$ exactamente, usamos la ecuación de diferencia dada en (3.18) para aproximar las derivadas parciales. Sin embargo, para calcular $\mathbf{x}^{(2)}$, nos apartamos del método de Newton y examinamos el método de la secante para una sola ecuación no lineal. El método de la secante usa la aproximación

$$\frac{f(x_1)-f(x_0)}{x_1-x_0}$$

para reemplazarla por $\dot{f}(x)$ en el método de Newton. Para sistemas no lineales, $\mathbf{x}^{(1)}-\mathbf{x}^{(0)}$ es un vector y el cociente correspondiente está indefinido. Sin embargo, el método procede de manera similar reemplazando la matriz $\mathbf{J}(\mathbf{x}^{(1)})$ en el método de Newton por una matriz A_1 con la propiedad de que

$$A_1\left(\mathbf{x}^{(1)}-\mathbf{x}^{(0)}\right)=\mathbf{F}\left(\mathbf{x}^{(1)}\right)-\mathbf{F}\left(\mathbf{x}^{(0)}\right) \tag{3.19}$$

La ecuación (3.19) no define a una sola matriz debido a que no describe como A_1 opera en vectores ortogonales a $\mathbf{x}^{(1)}-\mathbf{x}^{(0)}$. Como no se tiene información disponible acerca de los cambios de $\mathbf{F}$ en dirección ortogonal a $\mathbf{x}^{(1)}-\mathbf{x}^{(0)}$ se pide además de A_1 que

$$A_1\mathbf{z}=\mathbf{J}\left(\mathbf{x}^{(0)}\right)\mathbf{z} \quad \text{siempre que } (\mathbf{x}^{(1)}-\mathbf{x}^{(0)})^t\mathbf{z}=0. \tag{3.20}$$

Esta condición especifica que cualquier vector ortogonal a $\mathbf{x}^{(1)}-\mathbf{x}^{(0)}$ no será afectado por la renovación de $\mathbf{J}(\mathbf{x}^{(0)})$, que se utilizó para calcular $\mathbf{x}^{(1)}$, a A_1, y que se usará en la determinación de $\mathbf{x}^{(2)}$.

Las condiciones (3.19) y (3.20) definen únicamente a A_1 como

$$A_1=\mathbf{J}\left(\mathbf{x}^{(0)}\right)+\frac{\left[\mathbf{F}\left(\mathbf{x}^{(1)}\right)-\mathbf{F}\left(\mathbf{x}^{(0)}\right)-\mathbf{J}\left(\mathbf{x}^{(0)}\right)\left(\mathbf{x}^{(1)}-\mathbf{x}^{(0)}\right)\right]\left(\mathbf{x}^{(1)}-\mathbf{x}^{(0)}\right)^t}{\left\|\mathbf{x}^{(1)}-\mathbf{x}^{(0)}\right\|_2^2}$$

Esta es la matriz que se usa en lugar de $\mathbf{J}(\mathbf{x}^{(1)})$ para determinar $\mathbf{x}^{(2)}$:

$$\mathbf{x}^{(2)} = \mathbf{x}^{(1)} - A_1^{-1}\mathbf{F}\left(\mathbf{x}^{(1)}\right).$$

El método se puede repetir entonces para encontrar a $\mathbf{x}^{(3)}$ usando A_1, en lugar de $A_0=\mathbf{J}(\mathbf{x}^{(0)})$ y con $\mathbf{x}^{(2)}$ y $\mathbf{x}^{(1)}$ en lugar de $\mathbf{x}^{(1)}$ y $\mathbf{x}^{(0)}$. En general, una vez que se ha encontrado $\mathbf{x}^{(i)}$, se calcula $\mathbf{x}^{(i+1)}$ mediante

$$A_i = A_{i-1} + \frac{\left(\mathbf{y}_i - A_{i-1}\mathbf{s}_i\right)}{\|\mathbf{s}_i\|_2^2}\mathbf{s}_i^t \tag{3.21}$$

$$\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - A_i^{-1}\mathbf{F}\left(\mathbf{x}^{(i)}\right) \tag{3.22}$$

donde se introduce en (3.21) la notación $\mathbf{y}_i=\mathbf{F}(\mathbf{x}^{(i)})-\mathbf{F}(\mathbf{x}^{(i-1)})$ y $\mathbf{s}_i=\mathbf{x}^{(i)}-\mathbf{x}^{(i-1)}$ para simplificar la ecuación.

Si el método se lleva a cabo como se indica en las ecuaciones (3.21) y (3.22), el número de evaluaciones funcionales escalares se reduce de n^2+n a n (las requeridas para evaluar a $\mathbf{F}(\mathbf{x}^{(i)})$, pero el método aún requiere de $O(n^3)$ cálculos para resolver el sistema lineal $n\times n$ asociado

$$A_i\mathbf{y}_i=-\mathbf{F}(\mathbf{x}^{(i)}), \tag{3.23}$$

Ordinariamente no tendría justificación emplear el método en esta forma debido a que la convergencia cuadrática del método de Newton se reduce a la convergencia superlineal. Sin embargo, se puede incorporar una mejoría considerable empleando una fórmula de inversión matricial de Sherman y Morrison. Este resultado dice que si A es una matriz no singular y $\mathbf{x}$ y $\mathbf{y}$ son vectores, entonces $A+\mathbf{x}\mathbf{y}^t$ es no singular siempre que $\mathbf{y}^tA^{-1}\neq -1$. Además, en este caso,

$$\left(A+\mathbf{x}\mathbf{y}^t\right)^{-1} = A^{-1} - \frac{A^{-1}\mathbf{x}\mathbf{y}^tA^{-1}}{1+\mathbf{y}^tA^{-1}\mathbf{x}}. \tag{3.24}$$

Esta fórmula permite calcular a A_i^{-1} directamente de A_{i-1}^{-1}, eliminando la necesidad de invertir una matriz en cada iteración. Tomando $A=A_{i-1}$, $\mathbf{x}=\left(\mathbf{y}_i - A_{i-1}\mathbf{s}_i\right)/\left\|\mathbf{s}_i\right\|_2^2$, y $\mathbf{y}=\mathbf{s}_i$, la fórmula (3.21) junto con la (3.24) implican que

$$A_i^{-1} = \left(A_{i-1} + \frac{\left(\mathbf{y}_i - A_{i-1}\mathbf{s}_i\right)}{\left\|\mathbf{s}_i\right\|_2^2}\mathbf{s}_i^t \right)^{-1}$$

$$= A_{i-1}^{-1} - \frac{A_{i-1}^{-1}\left(\frac{\left(\mathbf{y}_i - A_{i-1}\mathbf{s}_i\right)}{\left\|\mathbf{s}_i\right\|_2^2} \right) A_{i-1}^{-1}}{1+\mathbf{s}_i^t A_{i-1}^{-1}\left(\frac{\left(\mathbf{y}_i - A_{i-1}\mathbf{s}_i\right)}{\left\|\mathbf{s}_i\right\|_2^2} \right)},$$

$$= A_{i-1}^{-1} - \frac{\left(A_{i-1}^{-1}\mathbf{y}_i - \mathbf{s}_i\right)\mathbf{s}_i^t A_{i-1}^{-1}}{\left\|\mathbf{s}_i\right\|_2^2 + \mathbf{s}_i^t A_{i-1}^{-1}\mathbf{y}_i - \left\|\mathbf{s}_i\right\|_2^2};$$

Así que

$$A_i^{-1} = A_{i-1}^{-1} + \frac{\left(\mathbf{s}_i - A_{i-1}^{-1}\mathbf{y}_i\right)\mathbf{s}_i^t A_{i-1}^{-1}}{\mathbf{s}_i^t A_{i-1}^{-1}\mathbf{y}_i} \tag{3.25}$$

Esta fórmula involucra sólo multiplicación de matrices en cada paso y requiere de $O(n^2)$ cálculos aritméticos. Por lo tanto, ésto elimina la necesidad de calcular A_i, así como la de resolver el sistema lineal dado por (3.23).

3.5 Un Algoritmo para Aproximar Ceroclinas

Recordemos el planteamiento del problema de ceroclinas. Dado el problema autónomo

$$\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}),$$

el conjunto de soluciones de $\dot{x}_i = f_i(\mathbf{x}) = 0$ se denomina ceroclina en x_i. Dicho en otras palabras, la ceroclina en x_i es el conjunto $D \in \Re^n$ para el cual se anula el "movimiento" respecto a dicha variable. Por ejemplo, para el caso de dimensión 2, se tiene que la ceroclina en x_1 es aquella curva que es cruzada por las soluciones en forma de recta paralela al eje de x_2; análogamente, la ceroclina en x_2 es una curva atravesada por las soluciones en forma de recta paralela al eje x_1. Por ejemplo, para el sistema neuronal de Fitzhugh-Nagumo dado por las ecuaciones

$$\dot{x} = -f(x) - y + I, \qquad \text{donde } f(x) = x(x-1)(x-a)$$
$$\dot{y} = b(x - gy)$$

se tiene la situación representada en la siguiente figura:

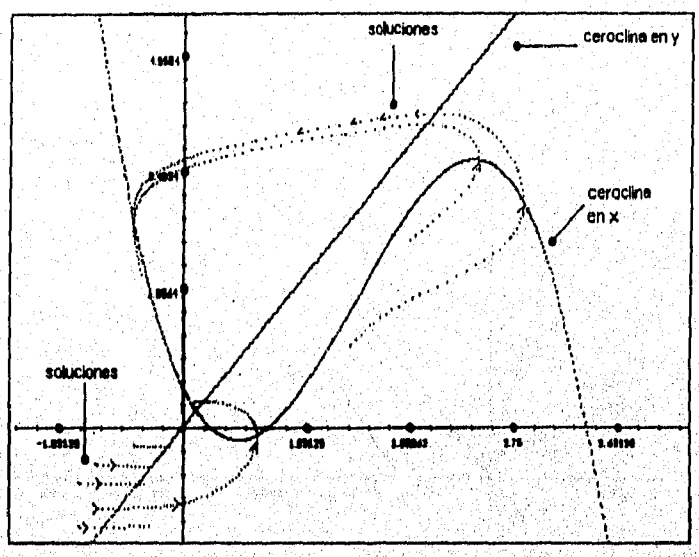


Figura 3.7. Las soluciones cruzan verticalmente (*x* no tiene movimiento) a la ceroclina en x. La ceroclina en y es cruzada horizontalmente pues y no tiene movimiento en esa curva. Los valores de los parámetros son a=3.3, I=0.53, b=1, g=0.5

Los puntos donde todas las ceroclinas se intersectan, son los puntos de equilibrio, que habían sido definidos anteriormente como soluciones especiales donde se anula el movimiento del sistema dinámico. En la figura 3.8 se observa la ubicación de la solución de equilibrio para el mismo sistema propuesto en el ejemplo.

Volvamos a considerar el caso general de dimensión n y pensemos en que se desea encontrar la ceroclina en x_i (para algún $i=1,2,\ldots,n$ arbitrario); es decir, se desea resolver la ecuación

$$\dot{x}_i = f_i(x_1, x_2, \ldots x_i, \ldots x_n) = 0.$$

La solución no es única, sino que consta de un conjunto de puntos. Tenemos ante nosotros un problema de búsqueda de soluciones de una ecuación donde $f_i: \Re^n \rightarrow \Re$. Los métodos numéricos que han sido discutidos hasta ahora son útiles para encontrar soluciones únicas ya sea de ecuaciones unidimensionales o vectoriales (sistemas de ecuaciones). Trataremos de deducir un algoritmo que parta de los métodos para ecuaciones unidimensionales pero que sea capaz de proporcionarnos el conjunto solución que estamos buscando.

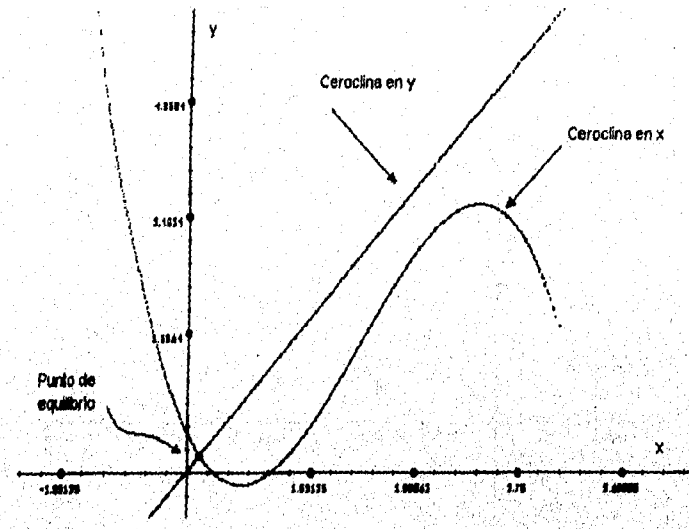


Figura 3.8 El punto de equilibrio para el sistema de FitzHugh Nagumo es en donde se intersectan sus ceroclinas.

La idea es intuitiva y muy sencilla. Pensemos en lo que solemos hacer cuando se nos pide graficar la solución de una ecuaciones de dos variables reales. Por ejemplo $x+2y=5$.

Lo primero que hacemos es expresar "una de las dos variables" en función de la otra, por ejemplo y en función de x: $y=(5-x)/2$. El siguiente paso es definir cuál

será el intervalo sobre el cual se variará a x, por ejemplo -10 a 10. Finalmente definimos el tamaño del incremento para x, por ejemplo .5 y construimos una tabla similar a la siguiente:

x	y
−10	7.5
−9.5	7.25
⋮	⋮
9.5	−2.25
10	−2.5

Una vez obtenidos dichos valores se grafican y en muchos casos se unen para formar una curva continua.

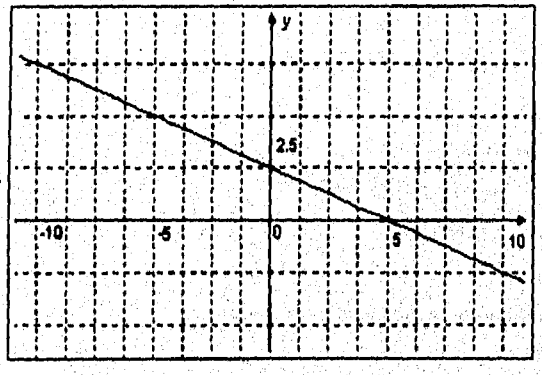


Figura 3.9. Gráfica correspondiente a la función x+2y=5 utilizando el intervalo [-10,10] para x.

Evidentemente, si se tienen los elementos de geometría analítica necesarios, el análisis es menos intuitivo en el caso de ecuaciones con una forma ya estudiada como la recta propuesta o como parábolas, hipérbolas, circunferencias, etc.

Esta forma de proceder para n dimensiones puede expresarse en los siguientes pasos:

1. Seleccionar la variable dependiente x_k donde $k \in \{1,2,\ldots,n\}$.

2. Seleccionar los intervalos en que serán variadas las variables independientes, así como el incremento con que se moverá cada una.

3. Para cada valor de x_i en su dominio, mover la variable x_{i+1} en todo su dominio. Para cada vector resultante obtener la solución para la variable dependiente utilizando alguno de los métodos numéricos para funciones unidimensionales .

Para comprender mejor la variación de las variables independientes, piénsese en el conjunto de vectores de $n-1$ componentes que es producido por el producto cartesiano de los conjuntos $D_1, D_2, .. D_{k-1}, D_{k+1}, ..., D_n$. Para cada elemento del conjunto se busca la incógnita x_k. Consideremos la ecuación $f(\mathbf{x})=0$, el algoritmo entonces puede expresarse en forma recursiva de la siguiente manera:

Procedimiento *Calcula*

1.Fijar *vfija* que es el índice de la variable dependiente y su valor inicial.

2.Fijar *min*[*i*], *max*[*i*] e *incr*[*i*] para $i=1,2\ldots,vfija-1,vfija+1,\ldots,n$, donde *min*[*i*] y *max*[*i*] son los límites inferior y superior para los intervalos de cada variable e *incr*[*i*] es el incremento con que se moverá cada una.

3. Mientras que $i \leq n$ seguir los pasos 4 y 5

4. Si *incr*[*i*] ≥ 0) entonces *CI*[*i*]=*min*[*i*].

en otro caso, *CI*[*i*]=*max*[*i*].

5. $i=i+1$.

CI se usa para almacenar los inicios de cada dominio

1. Definir *VarIt*=1.

2. Llamar al proceso *Itera*(*VarIt*).

3. SALIR

Procedimiento *Itera*(*VarIt*)

1.Si *VarIt*=$n+1$ entonces

$\mathbf{x}[vfija]$=***Metodo_Sol***$(f, vfija, x_1, x_2, \ldots, x_n)$

(Si ***Metodo_Sol*** converge entonces procesar el nuevo punto: registrarlo o graficarlo).

REGRESAR

2.Si *VarIt*=*vfija* entonces

llamar a *Itera*(*VarIt*+1)

REGRESAR

3.Hacer x[*Varlt*]=*Cl*[*Varlt*]

4.Hacer *i*=1.

5.Mientras que x[*Varlt*]≤*max*[*Varlt*] ejecutar los pasos 6 y 7

6.Llamar a *Itera*(*Varlt*+1);

7.x[*Varlt*]= x[*Varlt*]+*incr*[Varlt];

Metodo_Sol es el método de Newton Raphson, el de la Secante o cualquier otro que el usuario elija para solución de ecuaciones unidimensionales.

El usuario que así lo desee puede ver la implantación de este algoritmo en lenguaje C++ que se encuentra en el archivo METMATE.CPP del programa BIFURCA. Las funciones son *Calcula* e *Itera* y son miembros de la clase llamada *NumericCero*.

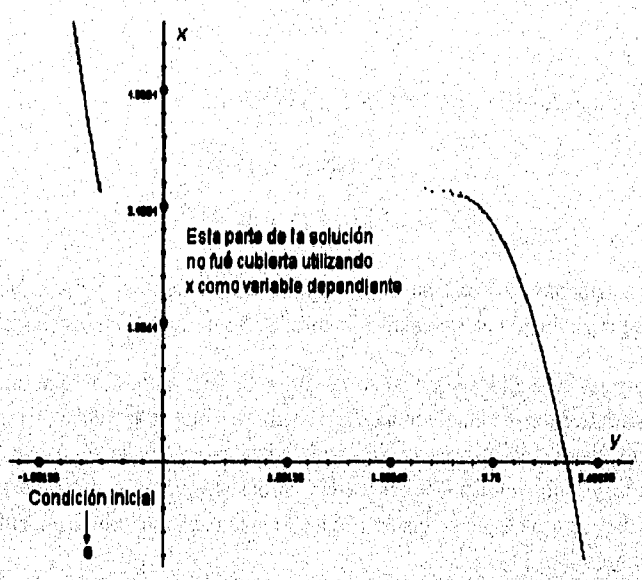


Figura 3.10 Caso en que la ceroclina no puede ser completamente determinada por el algoritmo.

En algunos casos, la elección de la variable dependiente no es determinante, es decir, se obtiene con la misma eficiencia. Sin embargo, en algunas otras ocasiones requiere de mucho cuidado si se quieren obtener resultados satisfactorios. Pensemos por ejemplo en el caso del modelo de FitzHugh-Nagumo. Si obtuviéramos la ceroclina

en x utilizando a x como variable dependiente, la solución graficada quedaría como en la figura (3.10). El método es capaz de encontrar sólo una variable dependiente para cada y. En el caso planteado, un mismo valor de y corresponde a dos diferentes en x por lo que es mucho mejor elección que y sea la variable dependiente.

3.6 Obtención de Puntos de Equilibrio

Toca su turno a la obtención de equilibrios. Recordemos que el problema consiste en encontrar las soluciones para las cuales el sistema dinámico

$$\frac{dx_1}{dt} = f_1(x_1, x_2, \ldots x_n)$$
$$\vdots$$
$$\frac{dx_n}{dt} = f_n(x_1, x_2, \ldots x_n)$$

no tiene movimiento. Es decir, las soluciones del sistema

$$f_1(x_1, x_2, \ldots x_n) = 0$$
$$\vdots$$
$$f_n(x_1, x_2, \ldots x_n) = 0$$

que es exactamente de la forma (3.2), así que en este caso se aplican los métodos discutidos para solución de sistemas n-dimensionales de ecuaciones no lineales.

Debido a las dificultades que ya se discutieron, sería imposible que un algoritmo pudiera encontrar todas las soluciones del sistema, así que se se dejó a responsabilidad del usuario elegir sus propias condiciones iniciales en donde intuya que hay una solución. Se ejecuta entonces la búsqueda y si después de un cierto número máximo de iteraciones retorna si no se encontró una solución satisfactoria. En ejemplo puede verse en la figura (3.11)

La determinación del número máximo de iteraciones y el criterio de error que permite decidir si una solución es satisfactoria o no, también son tareas del usuario. La implantación de los algoritmos de búsqueda de equilibrios en un sistema dinámico, se puede ver más detalladamente en el archivo METMATE.cpp. La clase es *NumericSolEq* y su definición se encuentra en el archivo METMATE.h.

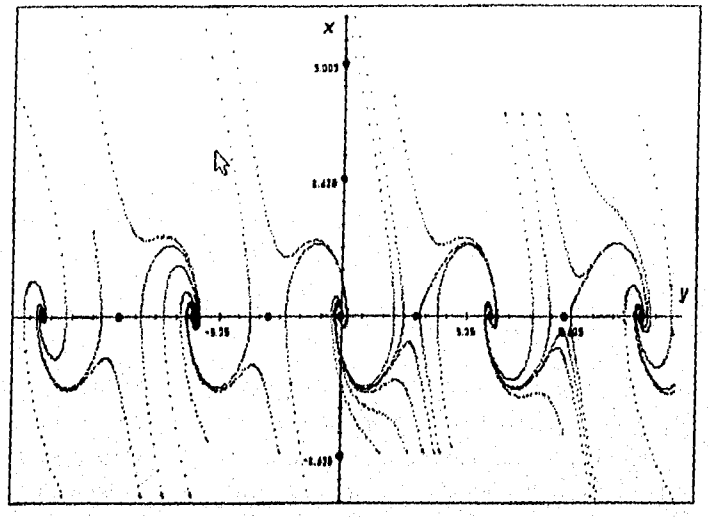


Figura 3.11 El sistema del péndulo con rozamiento, tiene un punto de equilibrio en cada múltiplo de π. Para localizarlos, el usuario debió ubicarse "suficientemente cerca" de cada uno.

3.7 Estabilidad de las Soluciones de Equilibrio

Una vez encontrada una solución de equilibrio, la estabilidad de éste se puede determinar si encontramos los eigenvalores de la matriz Jacobiana del sistema dinámico evaluada en dicho punto. Si la parte real de todos estos eigenvalores es menor que cero, entonces el punto es estable; en otro caso, el punto es inestable. La pregunta es ¿Cómo pueden determinarse los eigenvalores?. La respuesta a esta pregunta es bien sencilla si pensamos que los eigenvalores son las raíces del polinomio característico, P, de la matriz Jacobiana y que dicho polinomio es de la forma (3.3.1). La dificultad que queda es calcular los coeficientes de P, sobre todo cuando la dimensión del sistema es grande.

Existen métodos especialmente orientados a la búsqueda de eigenvalores. Sin embargo, en la bibliografía referente a la materia no se encontró ninguno que contemplara el caso complejo. Así, basados en los algoritmos para encontrar raíces de polinomios, se estableció una mecánica de operación. Los casos contemplados fueron sólo para dimensiones 2,3 y 4. Para dimensiones mayores a 5 ó 6, dicha mecánica

puede volverse inoperante debido a que se requieren los coeficientes del polinomio característico y no se construyó ninguna forma generalizada de obtenerlos a partir de la matriz Jacobiana. Para encontrar dichos coeficientes se utiliza la fórmula existente para encontrar determinantes y reduciendo términos semejantes después.

Para el caso de dimensión dos, se aplica la fórmula para solución de ecuaciones cuadráticas. En el caso de dimensión 3 al menos existe una solución real, por lo que ésta se busca mediante el método de deflación y las otras dos raíces se buscan nuevamente con la conocida fórmula utilizando el polinomio reducido Q que resulta al aplicar el método de Horner.

Para solucionar el caso de dimensión 4, se busca la primera solución λ utilizando el método de Müller. Si λ resulta ser real, entonces por lo menos existe otra raíz real y entonces se procede como en el caso de dimensión 3. Si λ es compleja, se efectúa una división del polinomio P entre a^2+b^2, donde $a=Im(\lambda)$ y $b=Re(\lambda)$, para obtener los coeficientes del nuevo polinomio reducido Q tal que $P(x)=\lambda\bar{\lambda}Q(x)$. El polinomio resultante es cuadrático por lo que nuevamente se aplica la fórmula analítica.

Para el lector interesado en profundizar en el desarrollo del algoritmo, la implantación se encuentra en el archivo METMATE.CPP del programa BIFURCA. en las funciones *CalcSolReal*, *P* y *SolCuadratica* que son miembros de la clase *NumericEigenv*.

BIBLIOGRAFÍA

[1] C. WILLIAM GEAR, *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall Inc., Engleewood Cliffs, N.J., 1971, 255pp.

[2] E. HAIRER, S.P. NORSETT, G. WANNER, *Solving Ordinary Differential Equations I, Nonstiff Problems*, Springer-Verlag, Berlín, 1987, p. 165-174.

[3] LAMBERT, *Computational Methods in Ordinary Differential Equations*, John Wiley&Sons, Great Britain, 1983.

[4] RICHARD L BURDEN, J. DOUGLAS FAIRES, *Análisis Numérico*, Grupo Editorial Iberoamérica, México D.F., 1985, p. 13-38, 222-316.

[5] KENNETH HOFFMAN, RAY KUNZE, *Álgebra Lineal*, Prentice Hall, Méx, D.F. 1987.

[6] WILLIAM E. BOYCE, RICHARD C. DIPRIMA, *Ecuaciones Diferenciales y Problemas con Valores en la Frontera*, Noriega Limusa, México D.F, 1991, 743 pp.

[7] WILLIAM H. PRESS, BRIAN P. FLANNERY, SAUL A. TEUKOLSKY, WILLIAM T. VETTERLING, *Numerical Recipes in C*, Cambridge University Press, New York, 1989, 566-592.

[8] SHOICHIRO NAKAMURA, *Numerical Analysis*, Cambridge University Press, New York, 1990.

[9] T.S. PARKER AND L.O. Chua, *Practical Numerical Algorithms for Chaotic Systems*, Springer-Verlag, New York, 1989.

Capítulo 4

UN ALGORITMO PARA OBTENER DIAGRAMAS DE BIFURCACIÓN

INTRODUCCIÓN

La ubicación y estabilidad de una solución de equilibrio en un sistema dinámico pueden cambiar al modificar los valores de sus parámetros. Por ejemplo, tal solución podría dejar de ser un atractor y dar lugar a un ciclo límite. Éste es un cambio cualitativo drástico en el comportamiento de un sistema, y uno quisiera poder caracterizar las condiciones que deben satisfacer los parámetros para que tales cambios ocurran. Conociendo la variedad de cambios posibles y las circunstancias en que estos suceden, se podría anticipar la evolución del sistema. De ésto se encarga la Teoría de Bifurcaciones.

El análisis de las órbitas en el espacio de fases permite entender los distintos comportamientos que puede exhibir el sistema al modificar las condiciones iniciales. El análisis de bifurcaciones también es importante, ya que el aspecto cualitativo del retrato de fases puede transformarse notoriamente al variar sus parámetros. Los valores para los cuales ésto ocurre son llamados valores o puntos de bifurcación. Puede pensarse que, para cada valor de los parámetros, se tiene un retrato de fases distinto. Esta información se puede desplegar pictóricamente en una gráfica que la codifica de manera sintética: el diagrama de bifurcación. En este capítulo se construye un algoritmo cuyo objetivo es la construcción de dichos diagramas, basado en los métodos numéricos para encontrar soluciones de sistemas de ecuaciones no lineales.

4.1 Un Ejemplo de Continuación de Puntos de Equilibrio

Consideremos el sistema de FitzHugh-Nagumo

$$\dot{x} = -f(x) - y + I, \qquad \text{donde } f(x) = x(x-1)(x-a)$$
$$\dot{y} = b(x - gy)$$

Obtener sus ceroclinas involucra solucionar las ecuaciones

$$-x(x-1)(x-a) - y + I = 0,$$
$$b(x - gy) = 0.$$

Entonces, la ceroclina en x está dada por una curva cúbica que puede ser trasladada hacia arriba o hacia abajo si incrementamos o decrementamos I. La ceroclina en y a su vez está representada por una recta que pasa por el origen con pendiente b/g. Esta situación puede apreciarse en la figura 4.1

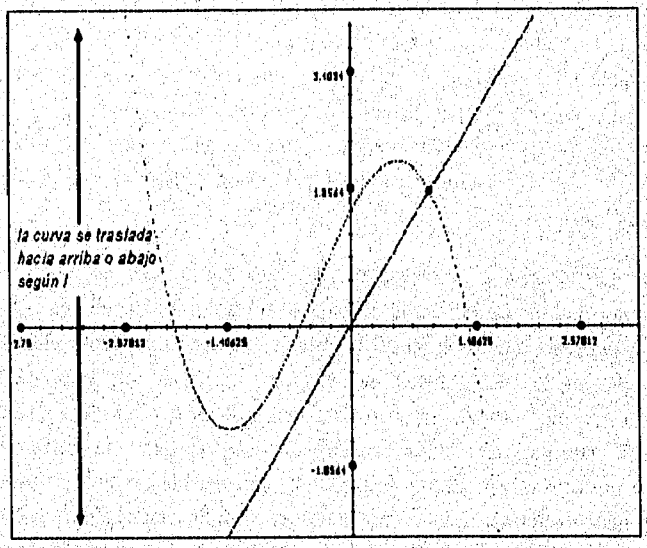


Figura 4.1 Forma general de las ceroclinas para el sistema de FitzHugh-Nagumo. Obsérvese que si la curva estuviera más abajo habría un momento en que aparecerían 3 equilibrios, pues las curvas estarían intersectadas en 3 puntos. Los valores de los parámetros fueron a=-2.6, I=1.51, b=2, g=1.50

Para continuar con nuestro ejemplo, analicemos qué sucedería con los equilibrios si incrementamos o decrementamos *I*. En la figura 4.2 se puede observar que cuando *I* decrece la tendencia es que exista solo un equilibrio estable. Cuando *I* crece, llega un momento en que las dos curvas se intersectan en dos puntos lo cual significa que ha aparecido un nuevo punto de equilibrio y que el actual cambió de estabilidad. Al seguir incrementando *I*, aparecerá entonces un equilibrio más hasta que la curva de la ceroclina en *x* suba tanto que prevalezca definitivamente únicamente un equilibrio estable.

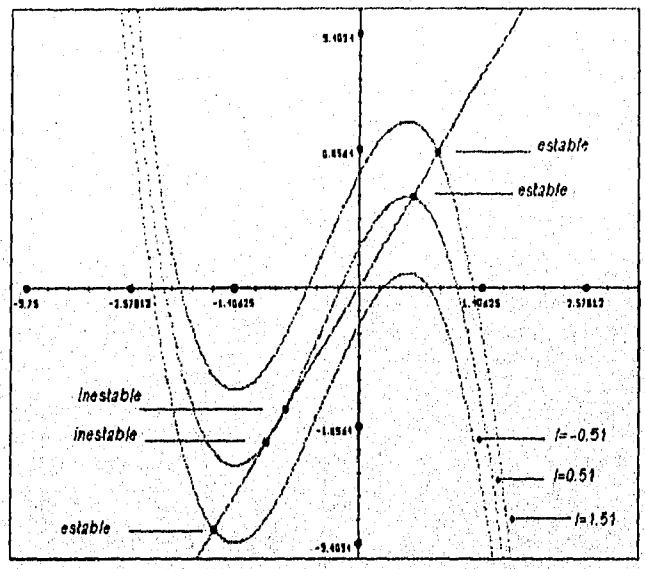


figura 4.2. La ceroclina en x sube o baja dependiendo del incremento en el parámetro I. Los valores para los demás parámetros fueron tomados como en el diagrama de la figura anterior.

En las figuras 4.3–4.5 se muestra el comportamiento del sistema dinámico para cada uno de los tres casos alrededor de los puntos de equilibrio existentes.

De la misma forma, podría realizarse el análisis por ejemplo sobre el parámetro *b* que determina la pendiente de la curva. Si la pendiente es positiva y crece, tiende a existir sólo un punto de equilibrio; si disminuye, aparecen dos y después tres equilibrios. Si la pendiente es negativa y decrece, se tiende a la existencia de un sólo equilibrio; si crece aparecerán dos y después tres. El lector que así lo desee puede efectuar dicho experimento utilizando el programa BIFURCA.

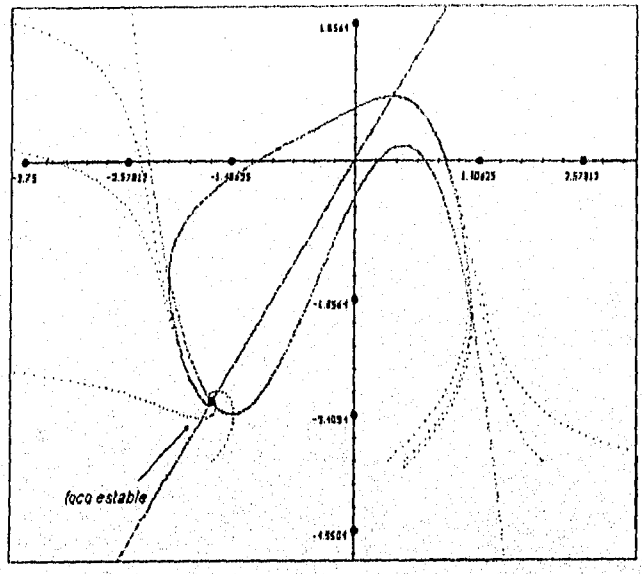


Figura 4.3 Comportamiento del sistema cuando I=–0.51. Los valores de los demás parámetros se siguen conservando como en los otros ejemplos. Se tiene la presencia de un sólo punto de equilibrio estable.

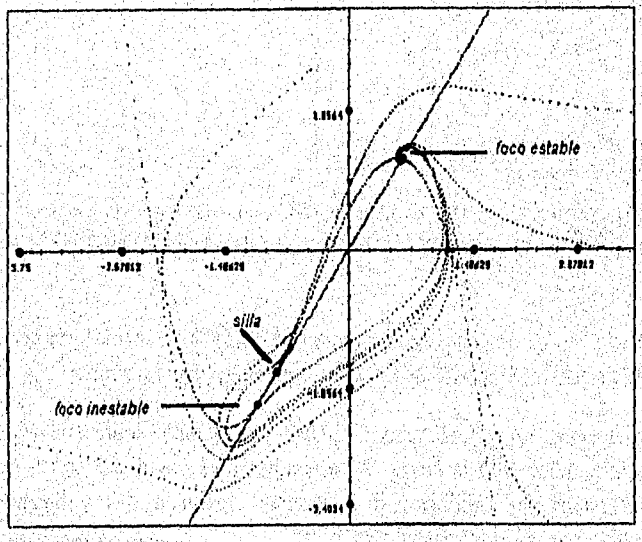


Figura 4.4 Comportamiento del sistema cuando I=0.51. Los valores de los demás parámetros se siguen conservando como en los otros ejemplos. Se tiene la presencia de tres puntos de equilibrio, dos inestables y uno estable.

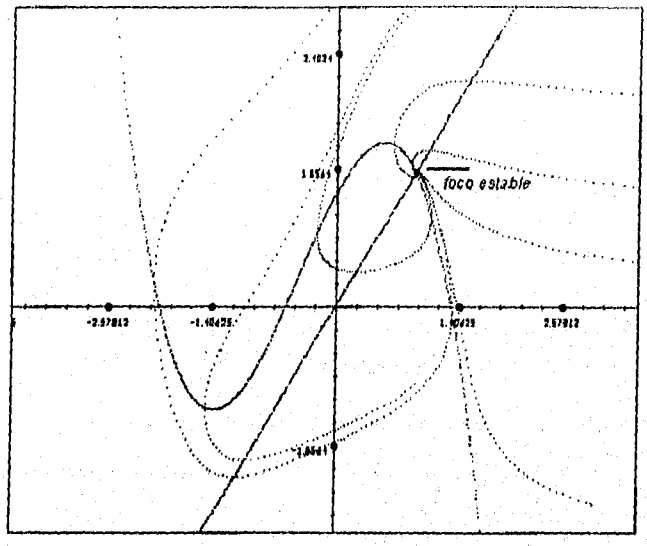


Figura 4.5 Comportamiento del sistema cuando I=-0.51. Los valores de los demás parámetros se siguen conservando como en los otros ejemplos. Se tiene la presencia de un sólo punto de equilibrio estable que prevalecerá definitivamente.

4.2 Un Algoritmo para el Trazo de Ramas de Puntos de Equilibrio

Supóngase que se tiene un sistema dinámico de la forma

$$\frac{d\mathbf{x}}{dt} = \mathbf{F}(\mathbf{x}; \lambda_1, \lambda_2, \ldots, \lambda_m),$$

donde $\mathbf{F}: \Re^n \rightarrow \Re^n$ y los λ_i son los parámetros que caracterizan al sistema. Construir un diagrama de bifurcación para un parámetro λ_i consiste en encontrar los puntos de equilibrio del sistema cuando el parámetro toma valores dentro de cierto intervalo. Una vez encontrado un punto de equilibrio se determina su estabilidad y se grafica. Para que pueda ser apreciado el cambio de estabilidad se utilizan diferentes estilos para el caso estable y el inestable. Evidentemente, para encontrar cada punto de equilibrio se utilizará uno de los métodos discutidos en el capítulo anterior para solución de sistemas de ecuaciones no lineales.

El lector recordará que para ejecutar cada uno de los métodos para búsqueda de soluciones es necesario predeterminar una condición inicial. En este caso, el usuario es responsable de proporcionarla para el primer valor que tome el parámetro. Una vez localizados los puntos de equilibrio para dicho sistema, se varía el parámetro y, la solución del punto de equilibrio para el nuevo sistema dinámico se busca utilizando el resultado obtenido en el sistema anterior como condición inicial. Dado que la curva de puntos de equilibrio respecto al movimiento de parámetros es continua, tomar el punto encontrado en la iteración anterior, acelerará la búsqueda del actual. Resumiendo, el usuario proporciona el número de iteraciones que serán realizadas sobre el parámetro, el incremento con que se varía éste y la condición inicial. Adicionalmente, se proporciona la información requerida por el método de solución que se emplee.

Algorítmicamente esta forma de proceder se puede expresar como:

1. Definir:
   Numlt=número de iteraciones a realizar;
   *Incr*λ=cantidad con que se incrementará el parámetro en cada iteración;
   CondIni=vector inicial determinado por el usuario;
   Estabilidad, para indicar si el punto encontrado es inestable o no;
   Eignv, matriz de dimensión *n*×2 para almacenar los eigenvalores de la matriz Jacobiana evaluada en el punto de equilibrio. En la primera columna se almacenarán las partes imaginarias y en la segunda las reales.
2. Mientras *Numlt* >0 seguir los pasos 3 al 7
   3. *Sol*=***Metodo_Sol***(*CondIni*, Eignv, Estabilidad, **F**, λ).
   4. Si ***Metodo_Sol*** encontró una solución entonces
      Pintar punto de equilibrio con el estilo adecuado para *estabilidad*.
   5. *CondIni*=*Sol*.
   6. λ=λ+*Incr*λ.
   7. *Numlt*=*Numlt*−1.

Como puede apreciarse, el algoritmo es bastante sencillo aunque está limitado a quizás una sola rama de puntos de equilibrio y, de hecho, ésta misma podría ser construida sólo parcialmente. Consideremos el ejemplo ilustrado en la figura 4.6 nuevamente con el sistema de FitzHugh-Nagumo y construyamos el diagrama de bifurcación para el parámetro *I* graficado contra *x*. El lector puede notar que para una

condición inicial dada por el usuario, el diagrama es incompleto pues para cierto valor del parámetro deberían aparecer tres puntos de equilibrio y conforme éste va incrementándose tales debieran desaparecer hasta quedar uno sólo nuevamente. Lo que el usuario obtiene con la forma de proceder descrita anteriormente es un sólo equilibrio para cada valor del parámetro.

Para obtener una descripción más completa de su diagrama, el usuario debería entonces construir un primer bosquejo del diagrama como el de la figura 4.6, a continuación moverse a condiciones iniciales cerca de la discontinuidad y ejecutar nuevamente el algoritmo de continuación de puntos de equilibrio. Evidentemente esta forma manual de operación debe complementarse con un buen conocimiento intuitivo.

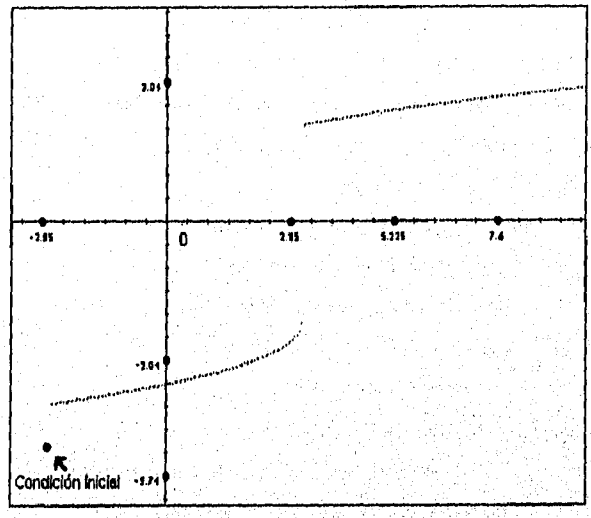


Figura 4.6 Ejemplo de un diagrama de bifurcación incompleto. El sistema es FitzHugh-Nagumo con a=-2.6, b=2, g=1.50. I se corre desde -2.267

Para construir un algoritmo más efectivo, se piensa en la posibilidad de que el usuario establezca una malla de condiciones iniciales para x. En este caso, para cada valor del parámetro, el algoritmo tendría que buscar una solución para cada condición inicial en la malla en lugar de utilizar el último punto de equilibrio encontrado.

Evidentemente, esta última forma de operación es muy efectiva aunque lenta y, de hecho, sigue requiriendo del conocimiento del usuario. En el ejemplo ilustrado, estaríamos en condiciones de dar la malla de condiciones iniciales porque por lo menos ya se tiene un bosquejo del diagrama.

El diagrama de ejemplo fué construido por BIFURCA e igual que fué graficado en el plano *lx*, también podría haberse escogido la proyección *ly* o bien verlo en el espacio tridimensional *lxy*. Para todos los casos el problema de encontrar más de una solución para cierto valor del parámetro es exactamente el mismo y el diagrama igual se vería discontinuo.

La segunda forma de operar se resume en el siguiente algoritmo de búsqueda:

1. Definir:
   NumIt=número de iteraciones a realizar;
   Incrλ=cantidad con que se incrementará el parámetro en cada iteración;
   CondIni=vector inicial determinado por el usuario;
   Estabilidad, para indicar si el punto encontrado es inestable o no;
   Eignv, matriz de dimensión *n*×2 para almacenar los eigenvalores de la matriz Jacobiana evaluada en el punto de equilibrio. En la primera columna se almacenarán las partes imaginarias y en la segunda las reales.
   VarMalla, índice de la variable sobre la que se define la malla
   LimInfMalla, límite inferior del intervalo de la malla
   LimSupMalla, límite superior del intervalo de la malla
   IncrMalla, incremento para la variable de malla
2. Mientras *NumIt* >0 seguir los pasos 3 al 9
   3. Hacer *CondIni*[*VarMalla*]=*LimInfMalla*
   4. Mientras *CondIni*[*VartMalla*] ≠ *LimSupMalla* seguir los pasos 5 a 7
      5. *Sol*=***Metodo_Sol***(*CondIni*, *Eignv*, *Estabilidad*, F, λ).
      6. Si ***Metodo_Sol*** encontró una solución entonces
         Pintar punto de equilibrio con el estilo adecuado para *estabilidad*.
      7. *CondIni*[*VarMalla*]=*CondIni*[*VarMalla*]+*IncrMalla*.
   8. λ=λ+*Incr*λ.
   9. *NumIt*=*NumIt*-1.

Dentro del programa BIFURCA, a este tipo de búsqueda se le denominó como *búsqueda exahustiva*. En la figura 4.7 se muestra un ejemplo de una corrida del algoritmo utilizando una malla sobre *x* estableciendo *LimInfMalla*=−3, *LimSupMalla*=2, e *IncrMalla*=0.2

Nuevamente, el lector interesado podrá encontrar la implantación de los algoritmos descritos dentro del archivo METMATE.CPP en las funciones *DiagBifSimple* y *DiagBifExahus* miembros de la clase **NumericBifurca**.

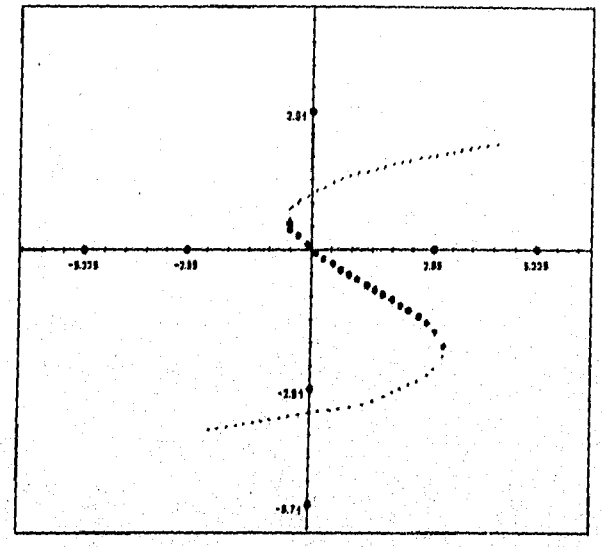


Figura 4.7 Es el diagrama correspondiente al de FitzHugh-Nagumo pero utilizando el algoritmo exahustivo. Note que ha sido dibujado mucho más completo. Los puntos más gruesos representan inestabilidad y los normales corresponden a puntos de equilibrio estables.

A continuación se discutirá un ejemplo en el que la posición de los puntos de equilibrio no varía, pero sí cambia su estabilidad. El péndulo matemático con rozamiento dado por las ecuaciones

$$\dot{x} = y$$
$$\dot{y} = -y - A\,sen(x)$$

tiene un parámetro que no determina la ubicación de los puntos críticos, únicamente determina la rapidez con que se mueve el péndulo y, por tanto, con que pasa a su posición de reposo (cuando cuelga hacia abajo con velocidad cero) para los mismos puntos críticos que siempre estarán en $(2\pi n, 0)$, es decir cada vuelta que el péndulo logre dar sobre su cuerda. En las figuras 4.8 y 4.9 se ilustra el comportamiento de las soluciones alrededor de los puntos críticos variando el parámetro A. Se ilustra también la forma de sus ceroclinas.

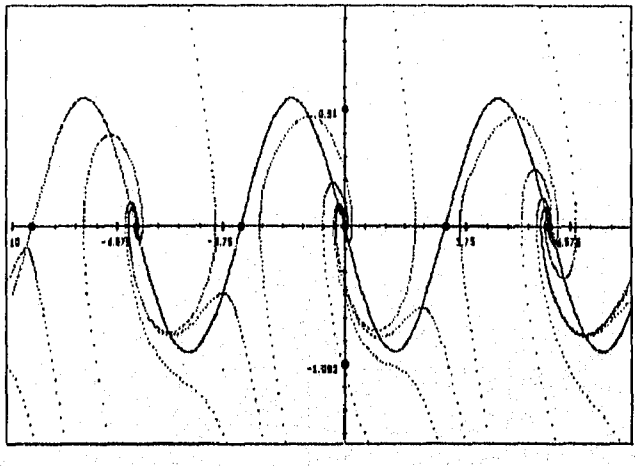


Figura 4.8. Plano fase del péndulo matemático con rozamiento cuando A=1.0

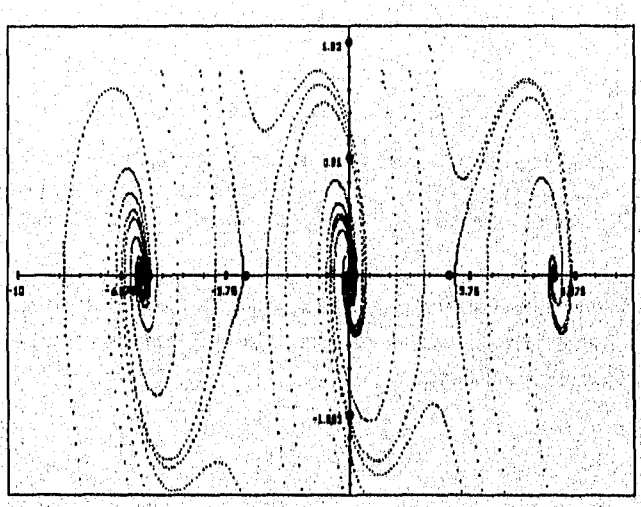


Figura 4.9. Plano fase del péndulo matemárico con rozamiento cuando A=2.0

En la figura 4.10 se puede apreciar la forma que toman las ceroclinas para diferentes valores de A.

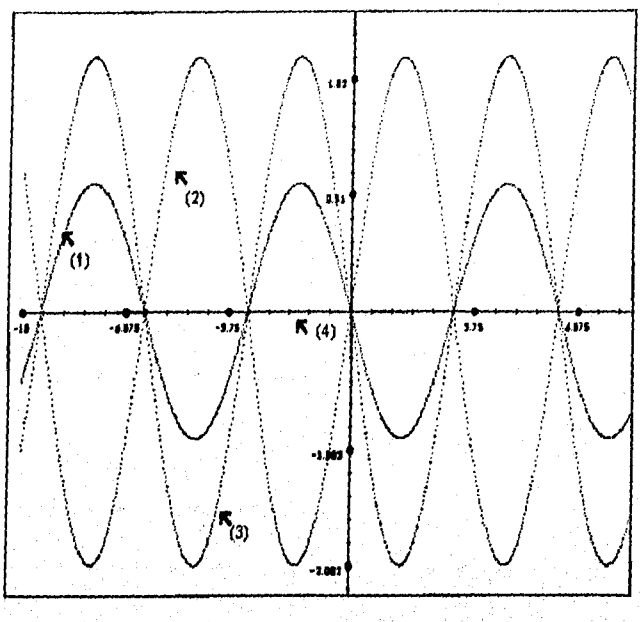


Figura 4.10. Forma para las ceroclinas para diferentes valores de A. (1) Ceroclina en y cuando A=1; (2) Ceroclina en y cuando A=2; (3) Ceroclina en y cuando A=−2; (4) Ceroclina en x es el eje y=0.

Finalmente, en la figura 4.11 se presenta el diagrama de bifurcación que indica que la posición y estabilidad de los puntos críticos no varía conforme varía el parámetro.

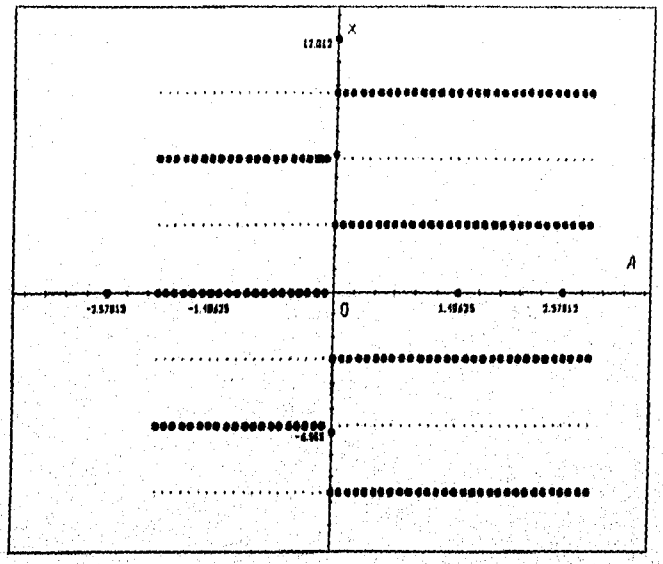


Figura 4.11. Diagrama de bifurcación para el sistema del péndulo. La posición de los puntos de equilibrio es invariante a cambios en el parámetro. Si la cuerda del péndulo pudiera ser negativa, entonces la estabilidad de los puntos cambiaría para valores negativos de A. Sin embargo, aunque A<0 es matemáticamente válido, en la realidad no es aplicable.

BIBLIOGRAFÍA

[1] RICHARD L. BURDEN, J. DOUGLAS FAIRES, *Análisis Numérico*, Grupo Editorial Iberoamérica, México D.F., 1985, p. 13-38, 222-316.

[2] WILLIAM H. PRESS, BRIAN P. FLANNERY, SAUL A. TEUKOLSKY, WILLIAM T. VETTERLING, *Numerical Recipes in C*, Cambridge University Press, New York, 1989, 566-592.

[3] SHOICHIRO NAKAMURA, *Numerical Analysis*, Cambridge University Press, New York, 1990.

[4] T.S. PARKER AND L.O. Chua, *Practical Numerical Algorithms for Chaotic Systems*, Springer-Verlag, New York, 1989.

Capítulo 5

PROGRAMACIÓN ORIENTADA A OBJETOS Y EL PROGRAMA BIFURCA PARA EL ANÁLISIS DE SISTEMAS DINÁMICOS

INTRODUCCIÓN

Nada para mi es mas satisfactorio que la presentación de este último capítulo. Pues no tan solo es la parte con que queda concluido este trabajo, sino también la consolidación de lo que alguna vez fué un sueño: desarrollar una aplicación de software que tuviera las características funcionales que las desarrolladas por expertos profesionales y que además sirviera como herramienta visual y de experimentación auxiliar en alguna área teórica de las matemáticas.

Si bien existen aplicaciones tan fuertes como Maple V, MathCad, Matemática, etc., éstas son tan generales que no alcanzan a cubrir las partes más especializadas de cada área. El caso del análisis de Sistemas Dinámicos no constituye una excepción, ya que este tipo de programas sólo cubren el trazo de planos fase, de gráficas de solución y del campo vectorial para ecuaciones diferenciales a lo más de dimensión dos. También existen programas cuyo único fin es el análisis de sistemas dinámicos que permiten una experimentación más completa, pero manejan únicamente sistemas de dimensión 1 ó 2 sin incluir el análisis de bifurcación.

Es muy común que en los laboratorios universitarios de investigación se desarrollen programas que instrumentan alguno o algunos algoritmos en particular para realizar experimentación y análisis de tópicos tan especializados como el análisis de bifurcación. Sin embargo, estos programas no están integrados a una aplicación

interactiva con el usuario y difícilmente pueden ser utilizados por alguien diferente de su creador.

El programa BIFURCA fué desarrollado con muchas cualidades adicionales a otros programas. Por ejemplo, es capaz de calcular el espacio en que se anula alguna de las ecuaciones (ceroclina) y graficarlo en proyecciones de dimensión 2 ó 3. También maneja sistemas de la dimensión que alcance a soportar la computadora que lo corre, dependiendo de la memoria RAM. Quizás la principal cualidad y que lo hace útil para muchos investigadores del área de Sistemas Dinámicos es el hecho de que incluye un módulo para el análisis de bifurcación.

Los conceptos fundamentales de la metodología orientada a objetos para el desarrollo de sistemas de software y su identificación en el Programa BIFURCA son materia de este capítulo. Por el momento justificaremos la elección por un lado de la metodología mencionada y, por otro, del lenguaje de programación C++.

Una característica fundamental de la Programación Orientada a Objetos es la de reusabilidad. Pretendiendo que las herramientas matemáticas implantadas en BIFURCA pudieran ser reutilizadas en cualquier otro diseño y, con ello, alargar su tiempo de utilidad, se eligió la Programación Orientada a Objetos. Ahora, imaginemos que ha llegado el momento en que es necesario añadir, modificar o utilizar uno de los módulos de BIFURCA para construir otro programa. Definitivamente, la cuna de cualesquiera de estas actividades será una universidad o similar y, por ahora, el lenguaje más popular en éstas es C o C++. Los otros lenguajes orientados a objetos no son muy conocidos y si lo son, no se pueden obtener tan fácilmente obtenibles. Así que se piensa que para los fines que BIFURCA fué creado el lenguaje óptimo es C++.

5.1 El origen de la programacion orientada a objetos

"Programación Orientada a Objetos" es un término de moda, sin embargo, acaba de cumplir sus bodas de plata. En 1967 fué desarrollado SIMULA, un lenguaje para aplicaciones de simulación, considerado por muchos el precursor en la introducción del concepto de objeto. Lamentablemente, este nacimiento en el norte europeo pasó inadvertido para gran parte de la familia de los programadores. Resulta que ahora tenemos un "niño de 25 años" que muchos se apuran en rebautizar.

Por aquel entonces se hablaba de la "crisis del software" que provocaría el surgimiento de la Programación Estructurada. Las "grandes" brigadas de programadores que programaban los "grandes" sistemas para las "grandes" máquinas enfrentaban serios problemas de organización y productividad. La respuesta fueron metodologías que promovían una solución jerárquica, disciplinada, organizada y ¡planificada! para desarrollar el *software*. Algunas mejoras fueron introducidas en los lenguajes de programación para facilitar ésto: mejores estructuras de control para obviar el *goto*, algunos recursos de modularidad y de organización de bibliotecas de programas.

Esto pareció ser suficiente durante los 70's. No obstante, pocos sistemas lograban terminarse, pocos se terminaban cumpliendo con los requerimientos iniciales y no todos los que se terminaban y cumplían los requisitos iniciales se usaban según lo planificado. El problema (mal llamado de mantenimiento) consistía en cómo adaptar el *software* a nuevos requerimientos imposibles de haber sido planificados inicialmente. La "estructuración" facilitaba en todo caso la zambullida de los "programadores de mantenimiento" en los mares de líneas de programa fuente, pero no la impedía.

Este alto grado de planificación y previsión es contrario a la propia realidad. El hombre aprende y crea a través de la experimentación, no de la planificación. Se necesitan medios que faciliten la experimentación y no que exijan que tenga que ser planificado un proyecto entero antes de poder escribir una línea de programa para que luego este proyecto se convierta en una camisa de fuerza en el desarrollo y evolución del sistema. El desarrollo de SMALLTALK a finales de los 70's fue una respuesta a este problema.

El desarrollo técnico del hardware, su disminución en precios y la explosión de las computadoras personales fue el detonante final. Con las computadoras al alcance de más personas, más programadores y además problemas más variados a resolver, aumenta la necesidad de desarrollar rápidos prototipos sin esperar a que los requerimientos iniciales estén totalmente precisos. Es conveniente poder aprovechar el software existente. También es necesario facilitar la adaptación de este *software* a otros usos, diferentes a los originales, sin necesidad de modificar el código ya existente.

La *Programación Orientada a Objetos* (POO), que por todo lo anterior algunos han dado en llamar la Programación Estructurada de los 90's, encuentra aquí un terreno fértil. ¿Es la POO un mejor paradigma que otros?. En cierto sentido sí lo es ¿Es una cura de todos nuestros problemas?. No, no lo es. ¿Significa que la arriba mencionada "crisis del software" desaparecerá?. Probablemente no. Pero entonces, *¿qué es lo grande de la Programacion Orientada a Objetos?*

En lugar de tratar de modelar un problema en algo familiar a la computadora se trata ahora de acercar a computadora al problema. Es decir, modelar la realidad del problema a través de entidades independientes pero que interactúan entre sí y cuyas fronteras no estén determinadas por su instrumentación computacional sino por la naturaleza del problema. Estas entidades serán denominadas objetos por analogía con el concepto de objeto en el mundo real.

Esto es tal vez lo bello de la POO, la sencillez de sus conceptos. Resolver problemas consiste en definir objetos y sus acciones y entonces invocar las acciones enviando mensajes a los objetos que ocultan las caracerísticas internas de cómo llevan a cabo estas acciones. Esta forma de resolver problemas luce familiar y lógica y ha sido utilizada por otras disciplinas científicas ya que en nuestra experiencia diaria nosotros "manipulamos objetos".

Programar en un lenguaje de programación orientado a objetos es definir clases (nuevos tipos de datos) que "expresan" una determinada funcionalidad la cual es común a todos los individuos de una clase. A través de esta funcionalidad los objetos o instancias concretas de una clase dan respuesta a las solicitudes (mensajes) que les envían otros objetos. Las clases deben ser lo suficientemente *CERRADAS* como para que cada objeto pueda ocultar la información (datos) que lo caracteriza como Individuo. El cómo llevar a cabo la funcionalidad descrita de la clase, es un problema interno del objeto.

Desde el punto de vista de las arquitecturas convencionales de computadoras, en la POO los datos no circulan abiertamente por todo el sistema como en la programación tradicional. Los programas no se dividen en declaraciones pasivas de estructuras de datos y funciones (que tal vez actúen sobre algunas de las estructuras de datos). Los datos están ahora encerrados dentro de cada objeto y éste es quien decide internamente cómo trabajar con ellos para dar respuesta a una solicitud de otro objeto.

Por otra parte estas clases deben ser lo suficiente *ABIERTAS* para permitir la reutilización, adaptación y extensión de las mismas a nuevas funcionalidades, sin correr el riesgo de afectar el funcionamiento de lo que ya está correcto.

Esta aparente contradicción de lo que se conoce como principio ABIERTO-CERRADO, es la piedra angular de la POO.

En ausencia de una definición formal de lo que es la POO, algunos prefieren identificarla por sus objetivos. Dos objetivos fundamentales persigue la POO para facilitar la experimentación:

1. La *robustez de las partes* que garanticen su integridad, y funcionamiento propio (lo cual a su vez facilita el arrinconamiento de las fallas). Esto se facilita en la POO a través de recursos como el encapsulamiento y el manejo de excepciones.

2. La *reusabilidad* y la *extensibilidad*. Su significado es que se pueden "derivar" (más que definir) nuevas clases de objetos a partir de las ya existentes, facilitándole con ello al programador el desarrollo de prototipos y una rápida exploración en las nuevas ideas. Ésto se facilita en los Lenguajes Orientados a Objetos (LOO) mediante la redefinición de operadores y funciones, la genericidad, la herencia y el polimorfismo.

Esta búsqueda de la comunalidad, y de aprovechar lo existente, es una de las características básicas de la POO. Algunos de estos recursos como el polimorfismo y la jerarquía de clases a través de la herencia, nos dotan de recursos para clasificar lógicamente los objetos, evitando las redundancias y obviando restricciones que otros paradigmas de programación nos imponen.

Otros conceptos como la *genericidad*, la manipulación de excepciones que no son inherentes al modelo de objetos, evitan las redundancias y obvian restricciones que otros paradigmas de programación del lenguaje C++.

5.2 C++ y otros lenguajes de programación

Algunos de los lenguajes orientados a objetos son SMALLTALK, Eiffel, y Actor. Sin embargo, la dificultad para los programadores de hacer el tránsito de la viejas a las nuevas técnicas y la "pesada" carga instalada (mental y electrónicamente) de software viejo, ha provocado el surgimiento de extensiones de los lenguajes tradicionales como LISP, Pascal, y C que pretenden establecer un compromiso con sus viejos programadores y las nuevas ideas. Esto último ha hecho que algunos se pregunten ¿es éste el mismo vino en nuevas botellas?. Al finalizar esta sección, el lector se dará cuenta de que no es así, es decir que C++ es una nueva concepción que soporta directamente conceptos de la Orientación a Objetos y además retiene los recursos de bajo-nivel y la eficiencia de C.

C++ comenzó su desarrollo en 1980 cuando Bjarne Stroustrup de los Bell Laboratories de AT&T necesitaba un lenguaje para desarrollar grandes programas de simulación. Su propuesta fué adicionar a C recursos de clases tipo SIMULA. El resultado inicial fué un lenguaje denominado C con Clases. Este lenguaje continuó evolucionando y se le fueron añadiendo nuevos recursos, aunque manteniendo los recursos cercanos a la máquina que tiene C, y el nombre fue cambiado por el de C++ (lo cual denota un paso más allá de C, tomado del operador de incremento de C, ++). Es por esto que C++ es considerado un lenguaje híbrido, porque en él coexisten la forma clásica de programación en C con los nuevos recursos y conceptos de la POO.

Algunos LOO (principalmente SMALLTALK considerado por algunos el patrón de la POO) no hacen chequeo estático de tipos. Es decir, todos los chequeos de tipo los hacen en tiempo de ejecución. Esto es aparentemente menos restrictvo durante la fase de experimentación pero impone una sobrecarga de tiempo de ejecución para los controles, pues es en ejecución donde se determina si un objeto puede dar respuesta o no a un mensaje. Por otra parte, deja los programas abiertos para posibles errores a veces difíciles de detectar.

Como LOO, C++ persigue la facilidad para el diseño de rápidos prototipos pero manteniendo de C la eficiencia y seguridad que deben tener los sistemas definitivos. Es por ello que C++ es un lenguaje con chequeo estático de tipos (static type checking): el compilador *controla el uso adecuado de las funciones que pueden ser realizadas a través de los objetos y de cuáles deben ser sus argumentos*, evitando con ello errores en tiempo de ejecución.

Algunos detractores esgrimen este hecho como contradictorio con la facilidad de experimentación. No obstante, mediante el polimorfismo y las funciones virtuales , C++ determina en tiempo de ejecución cuál es la función que dará respuesta a un mensaje y garantiza en tiempo de compilación la existencia de la misma. Esto combina la seguridad con un adecuado nivel de flexibilidad para múltiples situaciones.

Desde el punto de vista de la POO, C++ ofrece una buena cantidad de recursos (aunque a veces con sintaxis de semántica algo bizarras) como son las formas de encapsulamiento, la sobrecarga de operadores y funciones, la herencia (incluyendo la herencia múltiple) y el polimorfismo. Algunas instrumentaciones como BORLAND C++ incluyen la genericidad y otras, como Microsoft C++ incluyen el manejo de excepciones (ambas ya incluidas en ANSI C++).

Los mayores beneficios de C++ se apreciarán a largo plazo: alta calidad del software, mayor reutilización de código con más facilidades de adaptación apreciada sobre todo en los proyectos grandes. Una experiencia desarrollada por AT&T (los creadores de C y C++) durante 3 años demostró que los sistemas desarrollados en C tenían sólo una cuarta parte de reutilización de código anterior por tres cuartas partes en los sistemas desarrollados en C++.

Desafortunadamente C++, aunque permite evitar algunos de los problemas que promueve C (como el abuso en el manejo de punteros), adolece de algunas de sus deficiencias (como es la notación algo críptica).

Algunos autores plantean que los nuevos recursos de POO de C++ son más fáciles de aprovechar por un programador nuevo que por uno experimentado y deformado en los trucos de C. Incluso el propio Stroustrup plantea que quien programe en C++ pensando en el estilo tradicional de C no estará haciendo todavía POO. Sin embargo C++ ha sido diseñado como un supraconjunto de C para facilitar la migración de los programadores de C a C++.

Es este lastre de C, que incluye la "obsesión" por la eficiencia, lo que le ha hecho a C++ perder parte de la sencillez y la elegancia del paradigma de objetos. Realmente muchas de las "eficiencias" están condicionadas por las arquitecturas sobre las que trabajamos. Un compilador inteligente podría silenciosamente lograr muchas de estas optimizaciones. Lenguajes como C++ incitan a que sea el programador el que las busque desviándolo de la esencia del nuevo paradigma.

Sea el mejor o no, el mercado de C y los productores comerciales de software han convertido a C++ en un hecho (no obstante la falta de ortodoxia para algunos y la falta de elegancia para otros) que algunos han dado en llamar el *LOO "standard" de los 90's.*

5.3 Clases y Objetos

Tanto el ingeniero como el artista deben estar muy familiarizados con los materiales de su "negocio". Cuando utilizamos métodos orientados a objetos, nuestros bloques básicos de construcción son las clases y los objetos, con los que tendremos que estar muy bien familiarizados. En esta sección se definirán dichos conceptos más formalmente y se introducirán los conceptos más importantes relacionados con ellos.

¿Qué es un objeto?

La habilidad de reconocer los objetos físicos es una habilidad que el ser humano aprende desde muy temprana edad. Informalmente, decimos que un objeto es una entidad tangible que exhibe un comportamiento bien definido. Desde la perspectiva del aprendizaje humano, un objeto es cualquiera de lo siguiente:

- Una cosa tangible y/o visible.
- Algo que puede comprenderse intelectualmente.
- Algo hacia lo cual se dirige una acción.

Añadamos a nuestra definición informal la idea de que un objeto modela alguna parte de la realidad y es algo que existe en tiempo y espacio. En software, el término *objeto* fué formalmente aplicado en el lenguaje **Simula**. Otro aspecto importante de los objetos es que tienen un papel en el dominio de un problema que determina la manera en que colaboran entre sí para proporcionar un comportamiento global.

Podemos ahora construir una definición formal de *objeto*:

Un objeto tiene estado, comportamiento, e identidad; la estructura y comportamiento de objetos similares son definidos en su clase común; los términos instancia y objeto son equivalentes.

- El *estado* de un objeto incluye todas las propiedades de éste, mas los valores actuales de cada una de esas propiedades.

- Ningún objeto existe aislado. Más aún, los objetos actúan sobre otros objetos y a la vez otros actúan sobre ellos. Así, podemos decir que el *comportamiento* de un objeto es la manera en que éste actúa y reacciona en términos de sus cambios de estado y los mensajes que envía y recibe. El estado de un objeto representa el acumulado de los resultados de su comportamiento.

- La *identidad* de un objeto es aquella propiedad que lo distingue de los demás objetos.

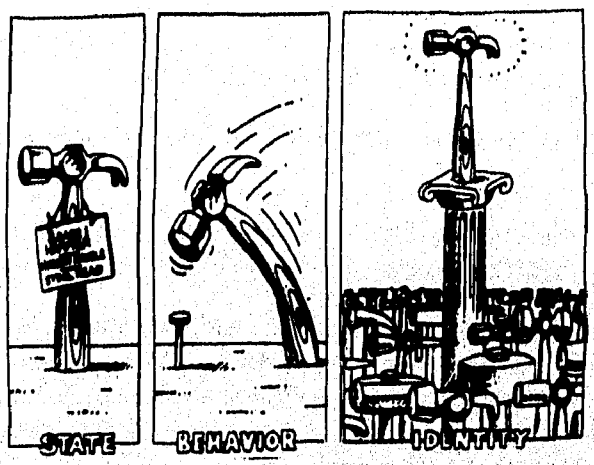


Figura 5.1. Un objeto tiene estado, exhibe un comportamiento bien definido, y tiene una identidad única

Relaciones entre objetos

Un objeto por sí mismo es poco interesante. Los objetos contribuyen con el comportamiento global del sistema colaborando entre sí. La relación entre dos objetos arbitrarios involucra la consideración de las operaciones que pueden ser ejecutadas y el comportamieto resultante.

Los dos tipos de relación que pueden darse son:

- **Ligas**

- Agregación

Ligas

El término de liga se deriva de Rumbaugh, quien la define como una conexión física o conceptual entre objetos. Un objeto colabora con otros a través de sus ligas con ellos. Dicho de otra manera, una liga denota la asociación específica a través de la cual un objeto (*el cliente*) solicita los servicios de otro (*el servidor*), o a través de la cual un objeto puede navegar hacia otro. Como participante de una liga, un objeto puede jugar uno de tres papeles:

Actor. Un objeto que puede operar sobre otros, pero nunca es operado por otros; en algunos contextos, este término es equivalente a *objeto activo.*

Servidor. Un objeto que nunca opera sobre otros objetos; solamente es operado por otros.

Agente. Un objeto que hace las veces de actor y de servidor. Generalmente un agente es creado para realizar alguna operación en nombre de actor u otro agente.

Para que un objeto pueda enviar un mensaje a otro, dicho mensaje debe ser visible en alguna forma. Durante el análisis del problema, la determinación de visibilidad puede ser ignorada, pero una vez que empieza la instrumentación concreta, debe considerarse la visibilidad, debido a que nuestras decisiones darán pie al alcance y acceso de los objetos en cada extremo de la liga.

Agregación

Así como una liga denota relaciones cliente/servidor, la agregación denota una jerarquía parte-de con la habilidad de navegar del todo (usualmente llamado *agregado*) hacia sus partes (igualmente conocidas como sus *atributos*). La agregación puede o no determinar contención física, por ejemplo, un aeroplano está compuesto de alas, motores, equipo de aterrizaje, etc. éste es el caso de contención física. Por otra parte, la relación entre un accionista y sus acciones es una relación de agregación que no implica contención física.

¿Qué es una clase?

Los conceptos de clase y objeto están estrechamente vinculados, pues no podemos hablar de un objeto sin considerar su clase. Sin embargo, existen diferencias sustanciales entre estos dos términos. Así como un objeto es una entidad concreta que existe en tiempo y espacio, una clase sólo es una abstracción, la esencia de lo que sería un objeto. Así, podemos hablar de la clase de los mamíferos que representa las características comunes a todos ellos. Para identificar un mamífero en particular de dicha clase, decimos "éste mamífero" o "aquel mamífero".

En términos cotidianos, podemos definir una clase como un grupo o conjunto caracterizado por atributos comunes. En el contexto de la orientación a objetos, definimos una clase como sigue:

Una clase es un conjunto de objetos que comparten una estructura y un comportamiento común. Un objeto es simplemente una instancia de una clase.

Evidentemente un objeto no es una clase, aunque curiosamente, una clase sí podría ser un objeto. Los objetos que no comparten una estructura y un comportamiento comunes no pueden ser agrupados dentro de una clase porque, por definición, no están relacionados más que por su naturaleza de objetos. Existen abstracciones tan complejas que no pueden ser expresadas en términos de una sola declaración de clases, así que se construyen varias más cuyas instancias colaboran entre sí para proporcionar la estructura y el comportamiento deseado.

Desde el punto de vista de la POO, la "colaboración entre objetos" significa que existe un contrato entre dichas clases en el cual se estipula la funcionalidad de cada instancia de esa clase. La interfaz es la vista externa que a la vez caracteriza a la abstracción; pero esa abstracción posee "secretos" acerca del por qué de su comportamiento. Dichos "secretos" son la vista interna de la clase.

La interfase de una clase se divide en 3 partes:

- ***Public.*** Accesible a todos los clientes.

- ***Protected.*** Accesible únicamente a los elementos internos de la clase, a las subclases y a sus "amigos".

- ***Private.*** Accesible solo a sus elementos internos y a sus amigos.

"Los amigos" son las funciones o clases a las que la clase en cuestión da privilegios sobre sí misma. Está por demás decir cuán cuidadoso debe ser el desarrollador para elegir a los amigos de una clase.

Una clase incluye la descripción de los datos (componentes) de cada objeto o instancia de la clase y de sus operaciones, llamadas funciones miembro en la terminología de C++ o métodos para otros como Smalltalk. Estas operaciones sólo podrán invocarse a través de los objetos de la clase o de los objetos de las clases "amigas". Dichas operaciones de una clase son las que dan vida al comportamiento de sus objetos.

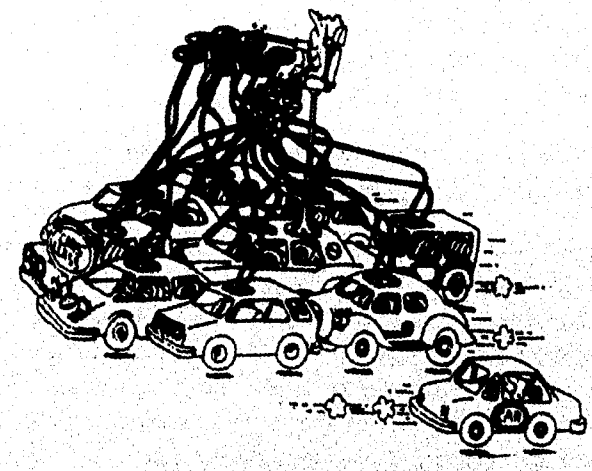


Figura 5.2. Una clase representa un conjunto de objetos que comparten una estructura y un comportamiento común

Relaciones entre clases

Por el momento, consideremos las similitudes y diferencias entre las siguientes clases y objetos: flores, margaritas, rosas rojas, rosas amarillas, pétalos y mariquitas. Podemos hacer las siguientes observaciones:

-Una margarita es una clase de flor.

-Una rosa es otra clase distinta de flor.

-Las rosas amarillas y las rojas son clases de rosas.

-Un pétalo es una parte tanto de las margaritas como de las rosas.

-Las mariquitas comen algunos organismos que pueden infestar ciertos tipos de flores.

De este ejemplo concluimos que las clases, como objetos, no pueden existir aisladas y de hecho, para un problema en particular, éstas están relacionadas en una interesante variedad de formas posibles. Son dos las razones por las cuales se establece una relación entre clases: primero, una relación entre clases podría indicar algún tipo de compartición de características comunes. Por ejemplo, tanto las margaritas como las rosas, ambas son clases de flores. Eso significa que las dos tienen pétalos de color brillante y emiten una fragancia. En segunda instancia, una relación entre clases podría indicar una conexión semántica. Por ejemplo, decimos que hay más cosas en común entre las rosas rojas y amarillas que entre las margaritas y las rosas y, las margaritas y las rosas a su vez tienen más relación que los pétalos con las flores. Análogamente, existe una conexión simbiótica entre las mariquitas y las flores: las mariquitas protejen a las flores de ciertas plagas, las cuales a su vez sirven como alimento para las mariquitas.

Existen tres tipos básicos de relación entre clases, mismas que podemos distinguir en nuestro ejemplo:

1. **Generalización-especialización.** Por ejemplo, una rosa es una clase de flor.

2. **Total-Parte.** Un pétalo no es una clase de flor; es una parte de ella.

3. **Asociación.** Denota alguna dependencia semántica entre clases con características diferentes. Por ejemplo la relación entre las mariquitas y las flores.

Los lenguajes orientados a objetos soportan tales tipos básicos de relación entre clases mediante los siguientes tipos de relación: asociación, herencia, agregación, uso e instanciación.

Asociación

Es el tipo de relación más general, ya que únicamente expresa que dos clases son dependientes entre sí. La dependencia entre clases se expresa en los términos de sus instancias. Por ejemplo, podrían expresarse algunas asociaciones como:

"Por cada instancia de la clase A, deben existir una o más de la clase B."
"Para varias instancias de la clase A pueden existir varias instancias de la clase B".

La identificación de asociaciones entre clases es a menudo una actividad que se lleva a cabo durante la etapa del análisis y durante los inicios del diseño y es, por tanto, semánticamente débil y no "demasiado expresiva". Conforme se avanza en el diseño y la instrumentación, las asociaciones entre clases van siendo refinadas hasta convertirse en una de las relaciones concretas como la de agregación o la de uso.

Herencia

Se dió ya una definición de lo que es la Herencia, así que enfocaremos nuestras definiciones hacia los dos tipos de herencia que existen y hacia el importante concepto de *polimorfismo.*

Herencia Simple

La herencia simple es una relación que existe entre clases en donde una clase comparte la estructura y/o comportamiento definido para otra. La clase de la cual hereda otra es conocida como su *superclase.* La clase heredera será, entonces una *subclase* para la *superclase.*

Una subclase generalmente aumenta o restringe el comportamiento de su superclase. Cuando una subclase aumenta el comportamiento de su superclase, se dice que utiliza la herencia por extensión. En una jerarquía de clases siempre existen una o más clases que son las más generales y de las cuales se empiezan a construir diferentes extensiones. Dichas clases raíz se conocen como *clases base.*

Una clase dada generalmente tiene dos clases de clientes:

- Instancias
- Subclases

A menudo es recomendable definir distintas interfases para estas dos clases de clientes. En particular, las instancias sólo deben ver la parte que define el comportamiento que para el público tiene un objeto de la clase. Sin embargo, es necesario exponer algunas otras funciones y representaciones para las clases herederas. La situación descrita es precisamente la motivación para las definiciones *private, public* y *protected* de C++.

Dado que el uso de la herencia expone algunos de los "secretos" de una clase heredada, se puede decir que existe un cierto grado de conflicto entre los conceptos de herencia y de encapsulamiento. Ésto se debe a que para poder entender el

funcionamiento de una clase heredera, muchas veces será necesario estudiar no sólo el comportamiento exterior de su superclase, sino también su vista interna.

Herencia significa que la subclase hereda el comportamiento y la estructura de su superclase. Los lenguajes de programación orientados a objetos permiten que los métodos de una superclase sean redefinidos y que se añadan nuevos métodos. Por ejemplo C++ permite declarar funciones como *virtual*, lo cual significa que pueden ser redefinidas por las clases herederas.

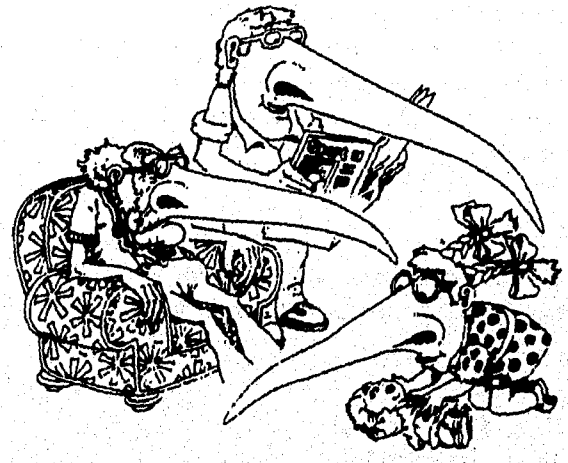


Figura 5.3. Una subclase puede heredar la estructura y el comportamiento de sus superclases

La facilidad que tienen las subclases para redefinir los métodos de su superclase se conoce como *polimorfismo*. Polimorfismo igualmente significa que un mismo objeto puede responder a un mismo conjunto de operaciones en diferentes formas, dependiendo, por ejemplo, del tipo de argumentos recibidos por sus métodos.

Herencia Múltiple

Con herencia simple, cada subclase tiene exactamente una superclase, sin embargo, en ocasiones se tiene la dificultad de decidir entre dos o más clases de las cuales otra debería heredar y de hecho, a veces se desea unificar varios comportamientos en uno solo, es decir, se desea utilizar la *Herenca Múltiple*. Algunos lenguajes orientados a objetos (C++ entre ellos) permiten este tipo de herecia si bien la cuestión de si ésta es realmente necesaria es un tema serio de discusión. En efecto, las más de las veces no

se necesita la herencia múltiple, pero cuando se requiera, será de mucha utilidad disponer de ella.

Agregación

Las relaciones de agregación entre clases tienen una relación directa con la relación de agregación entre objetos, ya que las relaciones de agregación entre clases darán como resultado relaciones de agregación entre las instancias de dichas clases. Recordemos que una relación de agregación es básicamente una relación de contención, es decir que las instancias de la clase contenedora poseen objetos de la clase contenida. Ahora, vale la pena aclarar que la agregación no necesariamente implica contención física, sino puede solamente contener una referencia a un objeto contenido en alguna otra clase. Así, hablaremos de *agregación por valor* y de *agregación por referencia.*

Uso

Cuando alguno de los métodos de una clase utiliza ya sea como argumento o como variable interna algún objeto de otra clase, diremos que existe una relación de uso entre dichas clases.

Instanciación

La idea de instanciación está íntimamente ligada con la de *genericidad* en tanto que instanciar una clase se define en términos de una *clase parametrizada*. Definamos entonces lo que ésto significa:

Una clase parametrizada es una a partir de la cual no pueden hacerse instancias en tanto que es genérica. Esto significa que la clase es una plantilla que define a un conjunto de clases. Dependiendo de los parámetros especificados en dicha plantilla, se producirán diferentes clases a partir de las cuales se pueden declarar objetos. En el momento que se construye una clase a partir de la plantilla, se dice que la clase plantilla (*template*) ha sido instanciada.

Supóngase por ejemplo una clase que define el conjunto de matrices numéricas. Las operaciones definidas sobre una matriz pueden ser de suma, resta, multiplicación, etc. Sin embargo, no se procede de la misma manera cuando se trata de una matriz en el espacio de los números complejos que en el espacio real. Se puede entonces construir una clase plantilla que puede instanciarse en una clase que maneje matrices reales y en otra para matrices complejas. Una vez instanciada dicha clase plantilla, se

tendrán dos clases para manejo de matrices a partir de las cuales ya se pueden declarar objetos (instancias).

5.4 El Modelo de Objetos

La tecnología orientada a objetos se fundamenta en una serie de elementos que colectivamente llamamos el *Modelo de Objetos*. Dicho modelo incluye los principios de *abstracción, encapsulamiento, modularidad, jerarquía, tipificación, concurrencia* y *persistencia*. Por sí mismos ninguno de ellos es nuevo. Lo importante es la manera en que el modelo de objetos los integra para formar un todo.

El análisis y diseño orientados a objetos constituyen un enfoque totalmente diferente del estructurado en tanto que tienen su fundamento en la programación orientada a objetos y no en la estructurada. Grady Booch define la POO, al DOO y al AOO de la siguiente manera:

Programación Orientada a Objetos (POO)

La programación orientada a objetos es un método de implantación en el cual los programas se organizan como colecciones "cooperativas" de objetos cada uno de los cuales representa una instancia de alguna clase, y cuyas clases son todas miembros de una jerarquía de clases unidas mediante relaciones de herencia.

Grady Booch nos dice que, si un programa parece orientado a objetos pero carece de uno de estos tres elementos, entonces no lo es. Por ejemplo, un programa sin herencia no es orientado a objetos y, de hecho se conoce como *Programación con tipos abstractos*.

Diseño Orientado a Objetos (DOO)

El DOO es un método de diseño que incluye el proceso de descomposición orientado a objetos y la notación necesaria paa expresar los modelos diseñados tanto lógica y física como estática y dinámicamente.

Precisamente, el proceso de descomposición orientado a objetos es el que hace al DOO tan diferente del estructurado. El primero utiliza abstracciones de objetos y clases; el segundo utiliza abstracciones algorítmicas.

Análisisis Orientado a Objetos (AOO)

Recuérdese que en el análisis estructurado, la parte medular es la construcción de los diagramas de flujo que siguen los procesos de la entidad a modelar. El AOO enfatiza la construcción de modelos del mundo real utilizando una visión orientada a los objetos del mundo.

El AOO es un método de análisis que examina los requerimientos desde la perspectiva de las clases y objetos encontrados en el vocabulario del dominio del problema.

Abstracción

Una abstracción denota las características esenciales que distinguen a un objeto de otras clases de objetos y de esta manera proporciona límites conceptuales, relativos a la perspectiva de cada quien.

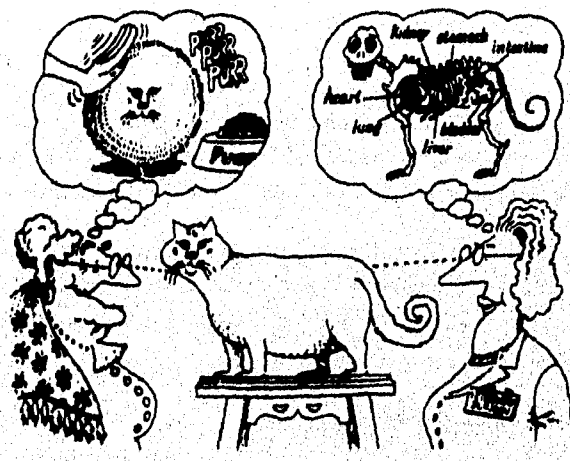


Figura 5.4. La abstracción se enfoca hacia las características esenciales de algún objeto, relativas a la perspectiva del espectador

Una abstracción se enfoca hacia el aspecto exterior de un objeto. Sirve entonces para separar el comportamiento esencial de un objeto de su instrumentación. Existe un espectro de abstracción que va desde objetos que modelan con mucha precisión problemas reales hasta los que verdaderamente carecen de razón de ser. De las más a las menos usadas, estos tipos de abstracción son:

•Abstracción de entidades. Un objeto que representa un modelo para el dominio de un problema o una solución.

•Abstracción de acciones. Un objeto que proporciona un conjunto generalizado de operaciones que pertenecen a un mismo tipo de función.

•Abstracción de máquina virtual. Un objeto que agrupa operaciones que son utilizadas por algún nivel superior de control ó todas las operaciones que utilizan un nivel inferior específico de otras operaciones.

•Abstracción coincidental. Un objeto que contiene un grupo de operaciones que no tienen relación entre sí.

El esfuerzo principal en la metodología orientada a objetos es sobre la construcción de abstracciones de entidad porque con éstas es que realmente podemos reflejar el dominio de un problema dado.

Un cliente es un objeto que utiliza los recursos de otro objeto (conocido como el servidor). Podemos caracterizar el comportamiento de un objeto considerando los servicios que éste proporciona a otros, así como por las operaciones que puede efectuar sobre otros. Esta forma de ver las cosas, nos fuerza a concentrarnos en el aspecto externo de un objeto y nos lleva a lo que se conoce como el *Modelo de Contrato* en programación: la vista exterior de un objeto define un contrato en el que se involucran otros objetos y el cual, eventualmente depende también de su organización interna a menudo en colaboración con otros objetos. Por tanto, este contrato establece todas las suposiciones que un objeto cliente puede hacer acerca del objeto servidor. Dicho en otras palabras, este contrato establece las responsabilidades de un objeto. Individualmente, cada operación que contribuye a este contrato tiene una identificación que incluye sus argumentos y tipo de información que retorna. Llamaremos PROTOCOLO al conjunto de operaciones que un objeto cliente puede ejecutar sobre otro, con las especificaciones de legalidad en que pueden ser llamadas.

Un concepto muy importante, es el de invarianza. Consideremos un "invariante" como una condición booleana cuyo estado verdadero debe ser preservado. Para cada operación asociada con un objeto podemos definir precondiciones (invariantes asumidas por la operación) y postcondiciones (invariantes satisfechas por la operación). La violación de una invariante rompe el contrato asociado con una abstracción. Si una precondición no es satisfecha, entonces el cliente no satisfizo su parte del compromiso y por tanto el servidor no puede proceder confiablemente. De la

misma forma, el que una postcondición haya sido violada, significa que el servidor falló con su parte del contrato, por lo que los clientes no pueden confiar en su comportamiento. Una excepción es una indicación de que una invariate no ha sido o no puede ser satisfecha. Ciertos lenguajes permiten a los objetos el disparo de excepciones que abandonan el proceso y alertan a algún otro objeto sobre el problema.

Encapsulamiento

La abstracción de un objeto debe ser precedida por las decisiones referentes a su instrumentación. Una vez que ésta es elegida, debería ser tratada como un secreto y escondida de los clientes, es decir mantenerse "encapsulada" o aislada de la apariencia exterior del objeto.

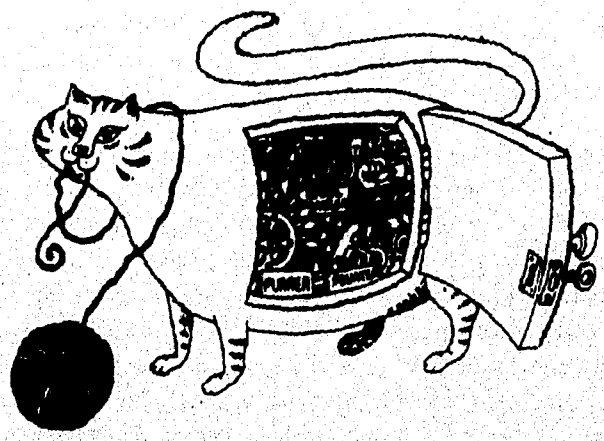


Figura 5.5. El encapsulamiento oculta los detalles de instrumentación de un objeto

Abstracción y encapsulamiento son conceptos complementarios: la primera se enfoca hacia el comportamiento observable de un objeto, mientras que el segundo lo hace sobre la instrumentación que dará lugar a dicho comportamiento. El encapsulamiento se logra escondiendo aquella información que no contribuye a las características esenciales del objeto; típicamente, la estructura de un objeto es ocultada.

El encapsulamiento proporciona barreras explícitas entre distintas abstracciones y produce una clara división de responsabilidades. Considérese por ejemplo la estructura de una planta. Para entender como trabaja la fotosíntesis a un alto nivel de abstracción, pueden ignorarse detalles como las responsabilidades de la raíz de la planta o las reaccioes químicas de las células. *Encapsular es guardar los secretos de una abstracción.*

Modularidad

Modularidad es la propiedad de un sistema que ha sido descompuesto en un conjunto de módulos cohesivos pero libremente acoplados.

Analizaremos ahora las ventajas de particionar un programa en componentes individuales. En primer lugar, ésto ayuda a reducir la complejidad del sistema y, en segundo, crea fronteras bien delimitadas que se documetan por sí solas en el programa. Dichas fronteras o interfases son invaluables en la comprensión del programa. Algunos lenguajes como SMALLTALK no incluyen el concepto de módulo por lo que la clase constituye la única forma de descomposición física. En otros lenguajes como C++, Object Pascal y Ada, el módulo es una construcción aparte y garantizan entonces, un conjunto separado de decisiones durante el diseño de nuestra aplicación.

En este tipo de lenguajes, las clases y los objetos constituyen la estructura lógica del sistema; distribuimos entonces estas abstracciones en módulos para producir su arquitectura física. Especialmente para aplicaciones muy grandes, en las cuales podemos tener varios clientes de las clases, el uso de módulos es esencial para manejar la complejidad.

La mayoría de lenguajes que soportan el módulo como un concepto separado, igualmente distinguen entre la interfaz y la instrumentación del módulo, es decir, encapsulación y modularidad van de la mano. Lo que en todo caso varía, es la forma

en que cada lenguaje maneja la modularidad, por ejemplo los módulos en C++ no son más que archivos compilados separadamente.

Al igual que la decisión de elegir el conjunto adecuado de abstraccciones, es muy difícil la decisión de cuál será el conjunto de módulos para un problema dado. Al respecto, se pueden dar las siguientes recomendaciones:

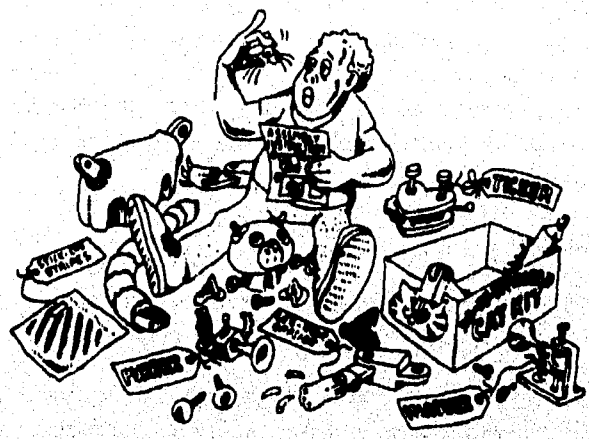


Figura 5.6. La modularidad agrupa las abstracciones dentro de unidades cohesivas libremente acopladas

•Para cualquiera, excepto para el software más trivial, la mejor solución es agrupar las clases y objetos relacionados en el mismo módulo.

•El objetivo de la modularización es la reducción en el costo del software ya que cada módulo puede ser diseñado y revisado independiente. La estructura de cada módulo deberá ser entonces suficientemente simple para ser entendida fácilmente.

•Debería ser posible cambiar un módulo sin requerir conocer la instrumentación de los otros módulos y sin afectar su comportamiento.

•El desarrollador debe balancear entre el deseo de encapsular las abstracciones y la necesidad de hacer ciertas abstracciones visibles a otros módulos. Este segundo punto es determinante en la modularización, ya que determina las dependencias entre módulos y por tanto los tiempo requeridos para recompilar.

•Los módulos deberían estar organizados de tal forma que pudiesen ser reutilizados convenientemente por otros desarrolladores; es decir, éstos deberían constituir un paquete de objetos y clases.

Jerarquía

Supongamos que un estudiante de anatomía quiere construir un modelo del cuerpo humano para entender su fisiología. Evidentemente éste es un sistema demasiado complejo para ser modelado con pocas abstracciones, más aún, todas esas abstracciones están relacionadas entre sí, en el sentido de que muchas de ellas se construyen a partir de otras; ésto es, unas heredan parte de la definición de las otras. Cuando la abstracción de un objeto incluye como parte suya a otro objeto, entonces diremos que existe una relación de agregación entre clases. Tenemos así, dos tipos de jerarquía: estructura de clases ("es una") y la estructura de objetos ("parte de").

Herencia

Cuando tenemos el caso en que una clase *x* es también otra (clase *y*), pero con propiedades adicionales que la identifican, decimos que la clase *x* heredó de la clase *y*. En caso de que una clase reuna las propiedades de varias otras clases y agregue o no propiedades, entonces tendremos un caso de herencia múltiple.

Agregación

Las relaciones "parte de" son formalmente llamadas *agregación*. Mientras las relaciones de herencia denotan especialización/generalización, las de agregación describen relaciones de contención/integración.

La agregación no es un concepto único de los lenguajes orientados a objetos, ya que cuaquier lenguaje de programación que soporte estructuras de registros, la soporta. Sin embargo, la combinación de agregación con herencia es poderosa: la primera permite la agrupación física de estructuras lógicamente relacionadas, y la segunda permite que las agrupaciones puedan ser reutilizadas. El lector puede ahora notar que el elemento de la POO que hace posible la reusabilidad es la herencia.

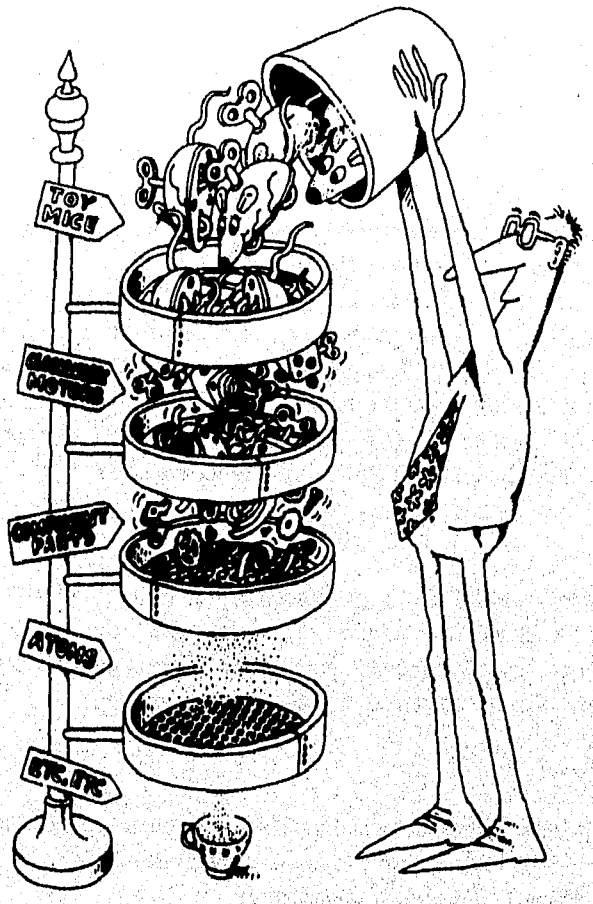


Figura 5.7. Las abstracciones forman una jerarquía

Tipos

El concepto de tipo y el de clase son muy similares. Se incluye, sin embargo, el de tipos porque hace un énfasis muy distinto al de abstracción. Booch define el concepto de tipos como aquello que fuerza a objetos de diferentes tipos a no ser intercambiados o, a ser intercabiados en una forma muy restringida.

Para comprender la importancia de la noción de tipos, pensemos en las unidades de medida utilizadas en las ciencias físicas.

Cuando dividimos la distancia por tiempos, esperamos un valor que denote velocidad y no peso. Similarmente el multiplicar una unidad de fuerza por temperatura no tiene sentido, pero multiplicar masa por aceleracón sí lo tiene. Existen entonces reglas que prescriben y fuerzan hacia ciertas combinaciones legales de abstracciones.

Un lenguaje de programación puede ser fuertemente tipeado, débilmente tipeado, o aún sin restricciones de tipo y todavía ser llamado orientado a ojetos. Los dos ejemplos de ciencias físicas dados, son de fuerte validación de tipos. En los lenguajes fuertemente tipeados, la violación de concordancia de tipos puede ser detectada en tiempo de compilación. En los lenguajes no tipeados como Smalltalk, dicha violación puede no ser conocida hasta el tiempo de ejecución. C++ es un ejemplo de lenguaje fuertemente tipeado ya que verifica en tiempo de compilación que haya consistencia en los tipos, es decir que las operaciones entre objetos involucren las clases adecuadas o bien que sea posible realizar una conversión de tipos, aunque ésta debería evitarse hasta donde sea posible porque a menudo representa violaciones a las abstracciones.

Los principales beneficios que tienen los lenguajes fuertemente tipeados son:

- Un lenguaje sin chequeo de tipos muy probablemente producirá programas que pueden fallar "misteriosamente" en cualquier momento durante su ejecución. No así uno fuertemente tipeado, ya que verifica durante el tiempo de compilación para que el desarrollador corrija cualquier posible inconsistencia y, por tanto, para que no reciba sorpresas desagradables posteriormente.

- Obligan a que el programador corrija la mayor parte posible de errores en un sólo tiempo de compilación sin tener que compilar-ligar-ejecutar cada vez que ocurra un error de inconsistencia de tipos durante la ejecución.

- Las declaraciones de tipo (necesarias en los lenguajes fuertemente tipeados) sirven como documentación para el programa.

- La mayoría de compiladores generan un código objeto si se hacen declaraciones de tipo y esto hace que la ejecución sea más rápida.

Figura 5.8. La validación de tipos previene la combinación indebida de abstracciones

Tal vez la única desventaja de los lenguajes fuertemente tipeados es que no son tan flexibles, como los que no lo son. Pero, ¿qué significa flexibilidad en el chequeo de tipos? La respuesta es que tal vez el programador no siempre necesita saber qué tipo entra o sale de un mensaje enviado a los objetos y, si es así, no es necesaria tal flexibilidad, ya que la cultura de los desarrolladores está orientada a siempre a conocer dicha información. Por tanto, en términos de eficiencia, definitivamente se considera que un lenguaje con chequeo de tipos es mejor a uno que no lo es.

Manejo dinámico y estático de tipos (ligadura dinámica y ligadura estática)

El concepto de verificación de tipos es muy distinto al concepto de manejo estático de tipos, también conocido como *ligadura estática* o *ligadura temprana.*

Este último concepto se refiere al momento en que los nombres se convierten en tipos. Ligadura estática significa que todos los tipos son fijados en tiempo de compilación. *Ligadura dinámica* (o *ligadura tardía*) significa que los tipos de todas las variables y expresiones no son conocidos sino hasta el tiempo de ejecución. El lenguaje C++ soporta el manejo de ligadura tardía, al mismo tiempo que el de chequeo de consistencia de tipos.

Concurrencia

Para ciertas clases de problemas, un sistema automatizado puede requerir el manejo de muchos eventos simultáneamente. Otros problemas involucran tal cantidad de cálculos que no es suficiente un sólo procesador. En ambos casos, es natural pensar en el uso de una computadora que sea capaz de soportar multitarea o bien, de varias computadoras distribuidas conectadas entre sí y ejecutando tareas simultáneas. Los sistemas que se ejecutan mediante múltiples UCP's (Unidades de Procesamiento Central) permiten la ejecución concurrente de diferentes procesos, mientras que los que se ejecutan sobre un solo UCP sólo aparentan hacerlo, usualmente mediante algún algoritmo de repartición de tiempos.

Actualmente, muchos sistemas operativos proporcionan soporte directo a la concurrencia en los sistemas orientados a objetos. Por ejemplo UNIX, Windows/NT y OS/2 proporcionan entrefases para crear y manipular procesos.

Figura 5.9. La concurrencia permite a distintos objetos actuar al mismo tiempo

Mientras la programación orientada a objetos se enfoca hacia la abstracción, encapsulamiento y herencia de datos, la concurrencia se enfoca hacia la abstracción y sincronización de procesos. El objeto es un concepto que unifica esos dos puntos de vista: cada objeto representa un proceso, es decir, es una *abstracción* de un proceso.

Tales tipos de objetos se llaman *activos*. En un sistema basado en el diseño orientado a objetos podemos conceptualizar el mundo como consistente de un conjunto de objetos en cooperación, algunos de los cuales son activos y por tanto sirven como centros de actividad independiente. Desde ese punto de vista:

concurrencia es la propiedad que distingue un objeto activo de uno que no lo es.

Persistencia

Cuando se habla de *software*, cada objeto tiene cierto espacio utilizado y existe durante un tiempo determinado.

Usualmente los lenguajes tradicionales de programación no manejan como parte de sí la persistencia de los datos que existen entre diferentes ejecuciones ó versiones de un programa, como tampoco de aquellos que persisten aún fuera del programa. Usualmente esos tipos de persistecia son el dominio de las tecnologías de base de datos. Así, introducir el concepto de persistencia al modelo de objetos da como resultado la idea de base de datos orientada a objetos. En la práctica, tales bases de datos se construyen sobre tecnologías ya probadas, tales como modelos jerárquicos, relacionales, secuenciales, etc., pero ofrecen al programador la abstracción de una interfase orientada a objetos. A través de dicha entrefase las consultas a la base de datos y otras operaciones son completadas en términos de objetos cuyo tiempo de vida trasciende el tiempo de vida de un programa individual. Esta unificación simplifica bastante el desarrollo de ciertas aplicaciones. En particular, nos permite aplicar los mismos métodos de diseño de los segmentos de base de datos y de los que no lo son.

Muy pocos lenguajes orientados a objetos proporcionan un soporte directo para manejar la persistencia: Smalltalk es una notable excepción aunque su manejo de persistencia no es tan eficiente como el de un manejador de base de datos.

Por otra parte, la idea de utilizar la programación orientada a objetos como "la piel" de una base de datos relacional para lograr una base de datos orientada a objetos parece ser la más atractiva pues ¿cómo arriesgar las grandes inversiones que hay en la tecnología relacional de bases de datos?

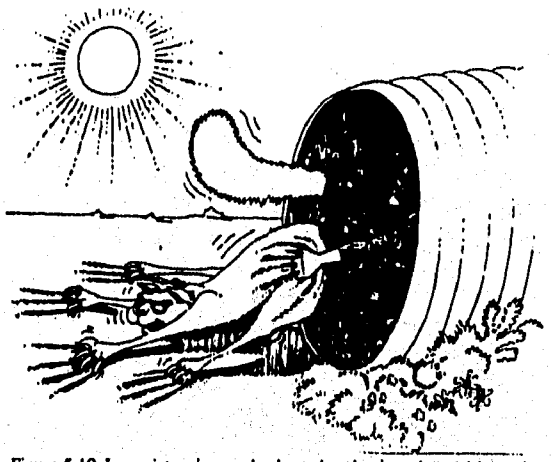


Figura 5.10. La peristencia guarda el estado y la clase de un objeto a lo largo del tiempo o espacio

Resumiendo, decimos que:

persistencia es la propiedad de un objeto por la cual su existencia trasciende en el tiempo (por ejemplo, el objeto continúa existiendo después de que su creador deja de existir) y/o espacio (el espacio físico en que fué creado inicialmente puede cambiar).

5.5 El Modelo de Objetos de BIFURCA

El programa BIFURCA fué desarrollado utilizando la metodología orientada a objetos. El principal motivo por el cual se usó dicha metodología es que permite crear módulos reutilizables. Por el hecho de estar tan estrechamente vinculados al *hardware*, los elementos correspondientes a la interfaz gráfica de BIFURCA: menúes, ventanas, botones y editores de texto, pueden estar sujetos a modificaciones drásticas. No es el caso de los procesos de uso menos general como los del desarrollo matemático los cuales fácilmente se pueden instrumentar en cualquier otro tipo de computadora que disponga de C++.

La Arquitectura de BIFURCA

Para asegurar la reusabilidad de los elementos matemáticos de BIFURCA, su arquitectura fué pensada en capas que hacen posible separar todas las clases para el manejo de los métodos numéricos necesarios de aquellas clases que contienen los elementos gráficos que ve el usuario como menúes, ventanas de edición e información, y botones. De igual forma, el módulo que contiene los elementos gráficos es independiente de la aplicación, en tanto que podría ser reutilizado por cualquier otra aplicación bajo el sistema operativo MS-DOS.

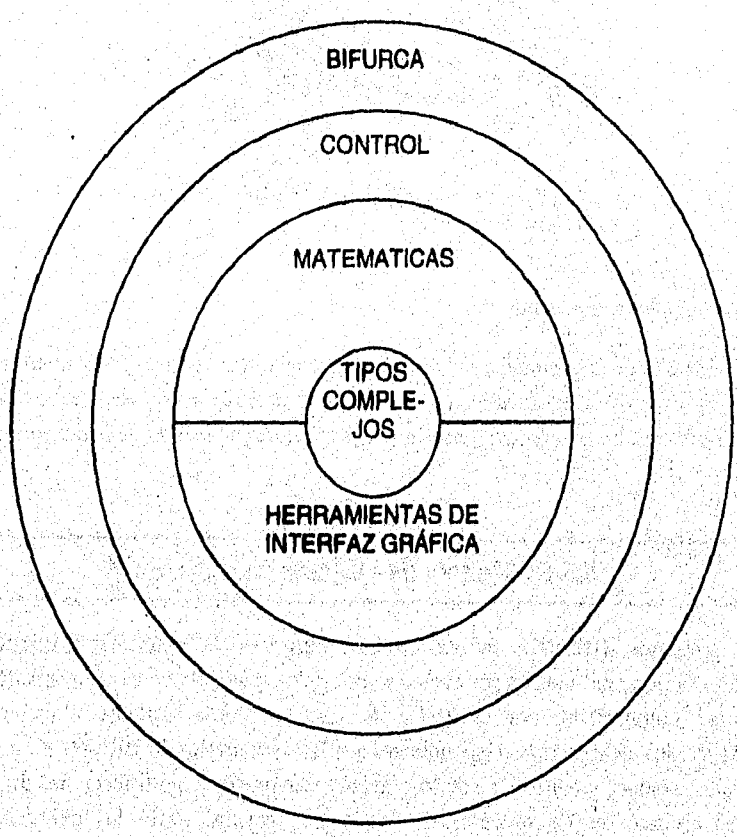


Figura 5.11 Arquitectura de BIFURCA

Las capas que componen el programa son las siguientes:

Bifurca

Esta es la capa más externa que establece la organización de todos los demás módulos para lograr el comportamiento global del programa y es, también, el "disparo" de inicio de éste. Esta capa no procesa los mensajes del usuario, ni toma ninguna clase de decisiones; solamente construye los objetos necesarios que darán vida al programa, los activa y coordina su funcionamiento.

Control

A esta capa pertenecen los objetos mediante los cuales el usuario realiza los procesos matemáticos que desea, permitiendo la visualización de resultados. El Control recibe los mensajes del usuario y hace las peticiones de servicio necesarias a los objetos matemáticos. Para comunicar al usuario los resultados obtenidos por los expertos (cuando sea necesario), el control elige una de dos opciones: envía directamente la respuesta valiéndose de algún objeto presentador de ventanas o, invoca a un objeto graficador que siempre está activo a lo largo de la ejecución de BIFURCA.

Matemáticas

Esta capa está representada por los objetos que realizan cualquier proceso matemático requerido por el programa. Los objetos de dichas clases reciben mensajes únicamente desde el control, pues no poseen la funcionalidad necesaria para presentar sus resultados por sí mismos. Para efectuar sus procesos contienen objetos cuya funcionalidad está definida en un nivel aún más interno. Estos últimos objetos son los *tipos complejos* dados por: matrices, ecuaciones diferenciales y sistemas dinámicos.

Herramientas de Interfaz Gráfica

Encontramos aquí, todos los objetos que componen una interfaz gráfica de uso general: graficador, ventanas, editores de texto, botones, etc. Su funcionamiento está regido por las llamadas desde la capa de control y se usan para recibir y presentar resultados del usuario y para el usuario. Puede decirse que "esta capa es la que da la cara al usuario sin ser la responsable de sus propios actos". Además ésta es dependiente de la plataforma de *hardware* y del sistema operativo que se utilicen.

Tipos complejos

Ésta es la capa más interior del programa y contiene los objetos de tipo complejo como matrices y sistemas dinámicos.

Cada capa de la arquitectura de BIFURCA está determinada por un conjunto de módulos físicos. A continuación se describen tales módulos y se proporciona un diagrama de las dependencias existentes entre ellos. Dichas dependencias no son bidireccionales en el sentido de que no son recíprocas y por tanto se llega un momento en que hay módulos no dependientes, lográndose así la existencia de unidades totalmente independientes y reutilizables.

•`BIFURCA`. Éste es el módulo raíz que contiene las declaraciones necesarias que determinan la identidad del programa y la capa más externa; ésto es, los elementos contenidos en dicho módulo se encargan de integrar los demás módulos para lograr el comportamiento de BIFURCA tal como lo ve el usuario.

•`CONTROL`. Contiene las clases y objetos que dan origen a la capa de Control.

•`METMATE`. Aquí están incluidas todas las clases para manejo de los métodos numéricos contenidos en el programa.

•`MATRICES`. Se incluyen aquí las clases necesarias para matrices y vectores de cualquier tipo ya que está instrumentado mediante clases genéricas (llamadas *templates* en el contexto de C++).

•`SISTEMAS`. En este módulo se agruparon las clases que definen los objetos que componen un sistema dinámico: ecuaciones, sistemas de ecuaciones y parámetros.

•`GRAFICA3D`. Aquí están incluidas todas las clases y objetos necesarios para graficación de resultados. Incluye facilidades para graficación en tres dimensiones, rotación de gráficas y cambios de proyección.

•`UTILS`. En este módulo están agrupadas las clases que definen los objetos de la interfase gráfica con el usuario. Así, define clases controladoras del teclado, del mouse, de la pantalla, de ventanas, de botones, etc.

•`GETOBJEC`. Se incluyen aquí las clases y objetos necesarios para entrada de datos editados por el usuario en distintos formatos numéricos y de cadenas de caracteres normales. Este módulo está íntimamente relacionado con el módulo `UTILS`.

La forma en que se agrupan los módulos en cada capa del programa BIFURCA es:

CAPA	MÓDULOS
BIFURCA	BIFURCA
CONTROL	CONTROL
MATEMÁTICAS	METMATE
HERRAMIENTAS DE INTERFAZ GRÁFICA	UTILS GETOBJEC GRAFICA3D
TIPOS COMPLEJOS	MATRICES SISTEMAS

Las dependencias entre los módulos quedan establecidas como sigue:

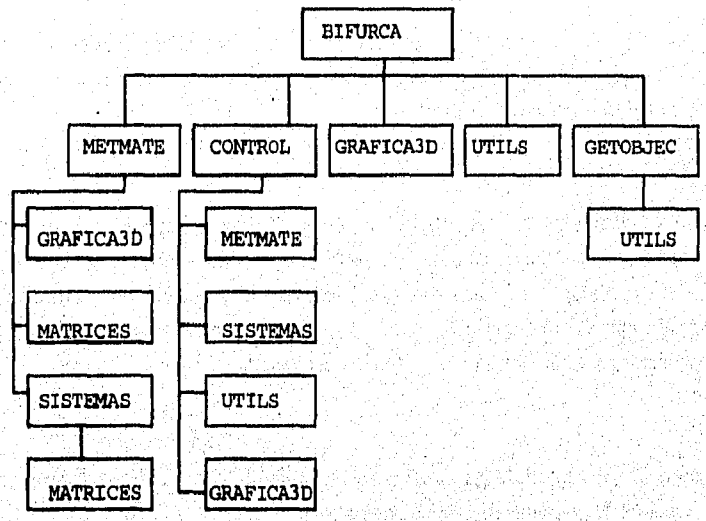


Figura 5.12. Dependencias entre módulos

En lo que resta de este documento, se tratarán los elementos más importantes de la definición y jerarquía de las clases relacionadas con las capas de control, matemáticas y tipos complejos que juntos conforman la parte matemática del programa BIFURCA. Se añade el módulo GRAFICA3D por representar la forma principal para la visualización de resultados.

Las Clases del Módulo MATRICES

Vector

Ésta es una clase genérica para manejar vectores de cualquier tipo. Redefine los operadores de igualdad, desigualdad, y otros más para responder a éstos en su forma particular, con lo que tenemos un ejemplo de polimorfismo.

`VectorNum`

Se trata de una clase genérica derivada (subclase) de la clase `Vector`, especializada en el manejo de cualquier tipo de vector numérico: de enteros cortos o largos, de reales de simple o doble precisión, y de complejos. Al especializarse en el manejo de números, introduce nuevas funciones para sumar, restar , multiplicar por escalar y efectuar producto interno entre vectores.

`Matriz`

Al igual que la de `Vector`, esta clase es genérica para cualquier tipo de dato (siempre que esté definido en el ámbito del programa). Las matrices en dicha clase son construidas a partir de vectores genéricos.

`MatrizNum`

Esta clase, se deriva de la clase `Matriz`. Dado que se especializa en matrices numéricas, introduce funciones para invertir matrices, obtener matrices transpuestas, multiplicar, sumar, restar, etc. Claro está que cualquier intento por parte del cliente por efectuar una operación cuando la matriz es inválida para ello, redundará en un retorno de error por parte de la función de la clase ejecutada.

`SLineal`

Se trata también de una clase genérica para cualquier tipo de número y define un objeto solucionador de sistemas lineales de ecuaciones.

Desde el momento de la instanciación, `SLineal` construye 2 matrices de factorización, por tanto, es muy rápido para obtener las soluciones ya que solamente efectúa un algoritmo de sustitución hacia atrás basado en el vector independiente recibido. Para su funcionamiento, contiene objetos tipo `VectorNum` y `MatrizNum`.

Las Clases del Módulo `SISTEMAS`

`Parametro`

Ésta clase contiene la definición de un parámetro en una ecuación. Así, básicamente contiene información sobre el valor del parámetro y su nombre. El

objetivo de expresar el parámetro como un objeto es porque son muy utilizados y susceptibles de cambios durante el programa.

`EcuacionSD`

Define los atributos de una ecuación del sistema dinámico. Entre dichos atributos, se define el vector de derivadas parciales de la ecuación respecto a sus distintas componentes.

`SistemaD`

Representa la definición de un sistema de ecuaciones diferenciales. Evidentemente debe contener parámetros y ecuaciones (un vector de `EcuacionSD`). Contiene las funciones necesarias para salvar su estado en disco y recuperarlo en corridas posteriores de BIFURCA. Envía los mensajes necesarios a sus objetos de ecuación y parámetros para evaluar el sistema de acuerdo a las peticiones que recibe.

Las Clases del Módulo GRAFICA3D

`AMBIENTE_GRAFICO_3D`

Define los elementos y atributos de un ambiente gráfico: ejes, graduación en dichos ejes, ángulo de rotación, ejes proyectados en caso de vistas en dimensión 2, etc. Las clases que dibujan los diferentes objetos gráficos definidos en este módulo, contienen un ambiente gráfico para regir la forma en que se presentan al usuario.

`VECTOR_3D`

Esta clase define los objetos vector que se dibujan en un `AMBIENTE_GRAFICO_3D`. Todo lo que contienen tales objetos son sus coordenadas en el ambiente, su dimensión, su color y su función para dibujarse por sí mismos. Como se mencionó, un objeto de la clase `AMBIENTE_GRAFICO_3D` representa el contexto en que estarán dibujados estos vectores. Nótese que no necesariamente son vectores de 3 componentes y que es el ambiente gráfico el que contiene la información de cuáles de las componentes del vector serán proyectadas al ser dibujado.

`VECTOR_GORDO_3D`

Clase exactamente como la anterior, variando únicamente el estilo de los vectores.

`LINEA_3D`

Se definen aquí los objetos tipo línea en un espacio n-dimensional que son proyectados en cortes en 2 o 3 dimensiones. Cada línea está determinada por dos puntos (sus puntos extremos) dados por objetos de la clase VECTOR_3D.

`GRAFICADOR_OBJETOS_3D`

Esta es una clase genérica que contiene los elementos básicos de un graficador. Está diseñada para almacenar tantos objetos gráficos (vectores, líneas, puntos u otros que se definan) como el usuario desee mientras la memoria RAM de la computadora lo permita. Ello con el fin de redibujar la gráfica en diferentes proyecciones y distintos ángulos de rotación.

`GRAFICADOR_3D`

Representa el tipo de objeto que "hará frente" a los clientes. Es decir integra un ambiente gráfico con graficadores para líneas, puntos y vectores que será usado por los clientes de la clase para lograr dibujar gráficas y manipularlas.

`CURSOR_3D`

Define un objeto gráfico que será empleado por el usuario para moverse en el ambiente de graficación en 3 dimensiones. Las coordenadas indicadas por este objeto son consideradas como las condiciones iniciales para los métodos numéricos utilizados en BIFURCA.

Las Clases del Módulo METMATE

`NumericIntegra`

Esta clase define un integrador numérico cuya función es la obtención de las soluciones de un sistema dinámico y mandará graficar dichas soluciones. El integrador contiene un catálogo de métodos numéricos y mantiene la información del estado de sus atributos numéricos: número de iteraciones, tamaño de paso, y método numérico que está siendo utilizado.

`NumericIntegra1P`

Se trata de una clase derivada de `NumericIntegra` y se especializa en la integración mediante métodos de un paso (como los que se describieron en el capítulo 2). Ésta clase introduce la facilidad de variación del tamaño de paso por lo que, adicionalmente a su clase base, incluye atributos referentes a la tolerancia del error y a los límites permitidos para el tamaño de paso.

`NumericIntegraKP`

Ésta también es una clase derivada de `NumericIntegra`, solamente que su especialidad es en los métodos multipaso y no introduce la facilidad de variación en el tamaño de paso.

`NumericCero`

Define al manejador numérico para la obtención de ceroclinas. Dicho manejador permite obtener la ceroclina respecto a cada variable del sistema además de que puede hacerlo fijando diferentes variables en cada ceroclina (como se describe en el capítulo 3). Al igual que para la clase `NumericIntegra`, el manejador contiene un catálogo de los métodos numéricos disponibles, en este caso los que se emplean para encontrar soluciones de ecuaciones de una variable.

`NumericEigenVals`

Esta clase define el manejador numérico que obtiene los eigenvalores para una matriz de tamaño hasta de 3×3. Dicha clase puede se reutilizada para crear una nueva que resuelva el problema para cualquier tamaño. La única modificación requerida en tal caso es únicamente en la función `Calcula` de la clase.

`NumericSolEq`

Define el manejador numérico para localización de puntos de equilibrio de un sistema de ecuaciones no lineales. Igual que los otros manejadores, contiene un catálogo de los métodos numéricos disponibles, para la localización de ceros de funciones de dimensión mayor que 1. Contiene además un manejador de obtención de eigenvalores del tipo `NumericEigenVals`, lo cual también constituye un ejemplo de una relación de agregación entre clases.

`NumericBifurca`

Esta clase define el manejador numérico que sirve para obtener diagramas de bifurcación.

Recuérdese que para obtener un diagrama de bifurcación es necesario efectuar la continuación de puntos de equilibrio (como se explicó en el capítulo 4) por lo que la clase contiene un manejador del tipo `NumericSolEq` para localizar puntos de este tipo.

Este manejador grafica directamente los diagramas de bifurcación, así que utiliza la clase GRAFICADOR_3D contenida en el módulo GRAFICA3D, lo cual constituye un ejemplo de una relación de uso entre clases.

Las Clases del Módulo CONTROL

Se mencionó que el módulo de control determina la capa de interfaz entre el usuario y los métodos matemáticos. Así que las clases aquí mencionadas definen objetos controladores con el "poder" de mandar los objetos gráficos y matemáticos a hacer lo que sea necesario para satisfacer los deseos del usuario. Son clases clientes y servidores a la vez. Cabe mencionar que la misma metodología en capas fué aplicada aquí, de tal manera que existe una clase `InterfazMatematica` que contiene un objeto de cada tipo especiallzado de interfaz y que será la que reciba directamente los mensajes del usuario.

`ControlColors`

Esta clase sólo contiene la definición de colores para cada tipo de gráfica: curvas de solución, ceroclinas, diagramas de bifurcación, o puntos de equilibrio. Igualmente contiene un método encargado de recibir del usuario los colores deseados para cada caso.

`ControlSistemaD`

Es la clase que se comunica con los sistemas dinámicos directamente para abrirlos, cerrarlos, modificarlos o evaluar sus ecuaciones de acuerdo a las entradas de información del usuario. Obviamente, contiene entre sus atributos un objeto del tipo `SistemaD`.

`InterfazIntegra`

Contiene entre sus atributos objetos de tipo `NumericIntegra1P` y `NumericIntegraKP` para comunicarse con las rutinas de cálculo y graficación de soluciones.

`InterfazCeroc`

Igual que las anteriores, esta clase contiene un controlador numérico (`NumericCero`) para la localización de ceroclinas.

`InterfazEqui`

Es totalmente análoga a las clases de interfaz anteriores pero con un controlador tipo `NumericSolEq` para encontrar puntos de equilibrio.

`InterfazBifurca`

Es totalmente análoga a las clases de interfaz anteriores pero con un controlador `NufmericBifurca` para dibujar diagramas de bifurcación.

`InterfazMatematica`

Esta es la interfaz que recibe directamente las entradas del usuario. Como parte de ella tiene una interfaz de cada tipo para controlar los distintos procesos numéricos, un controlador del color, y un objeto graficador de tipo `GRAFICADOR_3D` para la presentación de los resultados de algunos procesos numéricos.

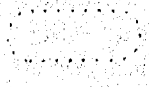
El lector interesado en ver la forma en que se instrumentaron las definiciones de los módulos de **BIFURCA**, puede mirar los archivos que tienen el mismo nombre que los módulos pero con extensión ".h". Por ejemplo, para consultar las definiciones hechas en el módulo **CONTROL**, ábrase el archivo CONTROL.H

5.6 La Jerarquía de Clases de bifurca

En esta sección se proporciona al lector el diseño jerárquico de la parte matemática de BIFURCA, incluyéndose relaciones de herencia, uso y agregación. Después de eso, se muestra la parte esencial de la definición de las clases en el lenguaje C++.

La notación gráfica que se utilizará para los diagramas de jerarquía de clases es la siguiente:

Dentro de un óvalo punteado se representa una clase común



Dentro de un óvalo punteado se representa una clase genérica

Herencia. La dirección de la flecha apunta hacia la superclase

Contención por valor. El círculo apunta hacia la clase contenida

Contención por referencia. El círculo apunta hacia la clase contenida

Relación de uso. El círculo se encuentra del lado de la clase usada

<tipo> Se incluye esta especificación para indicar el tipo con que se instanciará una clase

Jerarquía para el módulo MATRICES

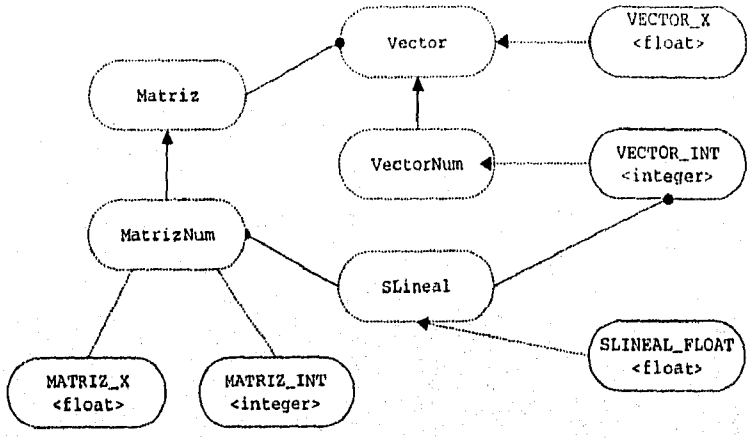


Figura 5.13

Jerarquía para el módulo SISTEMAS

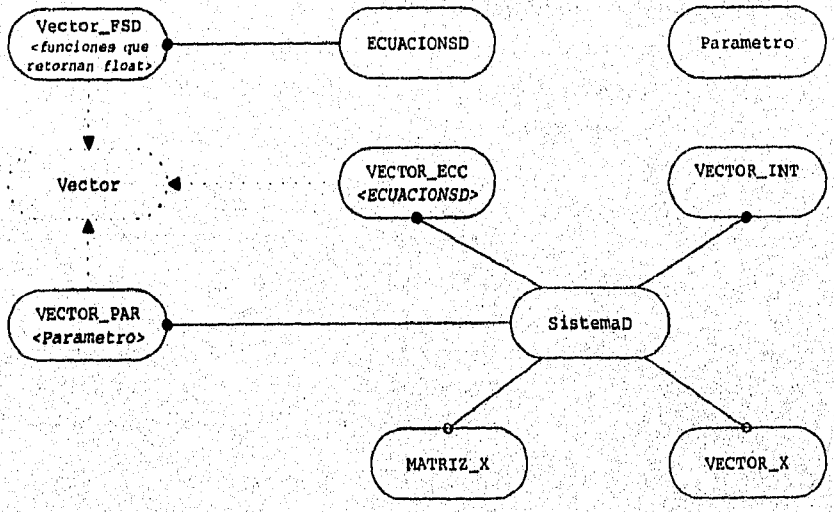


Figura 5.14

Jerarquía para el módulo GRAFICA3D

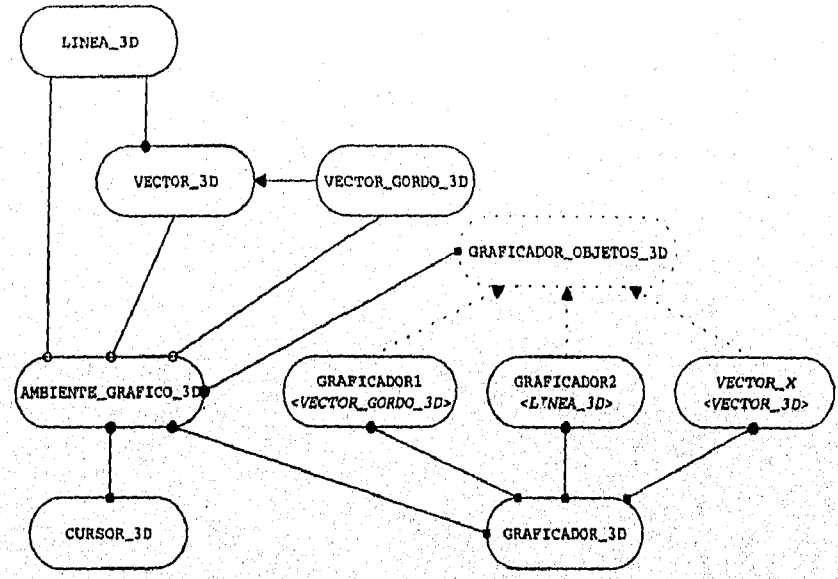


Figura 5.15

Jerarquía para el módulo METMATE

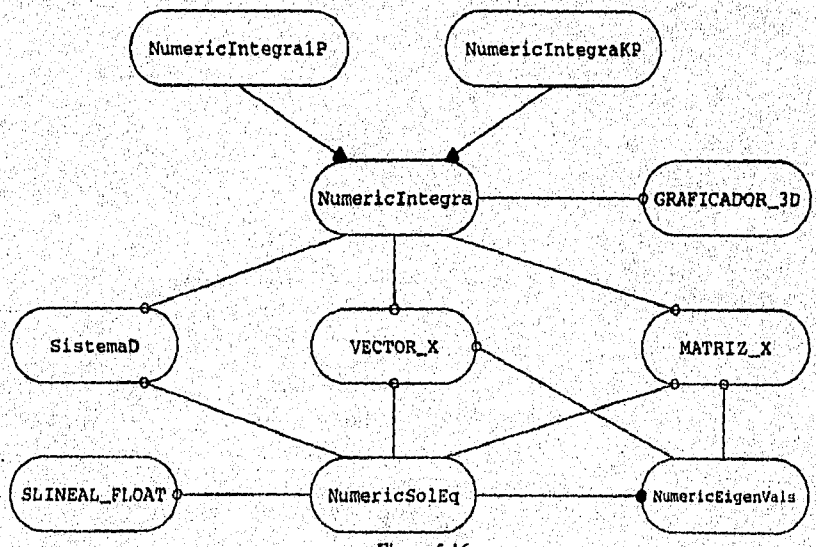


Figura 5.16

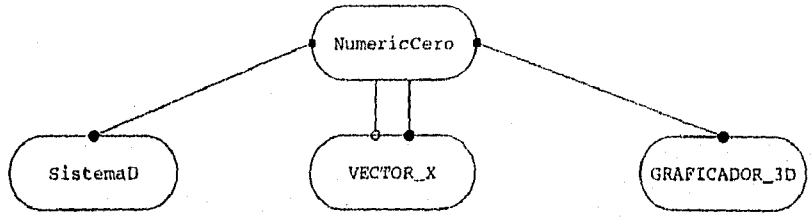


Figura 5.17

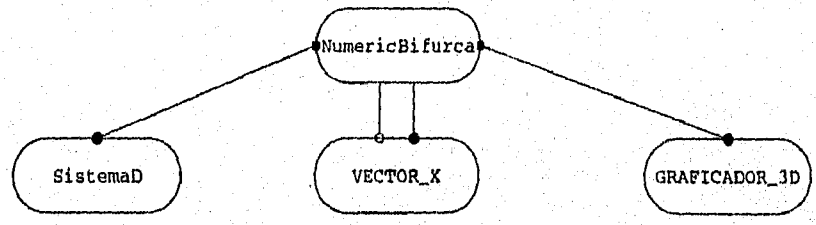


Figura 5.18

Jerarquía para el módulo CONTROL

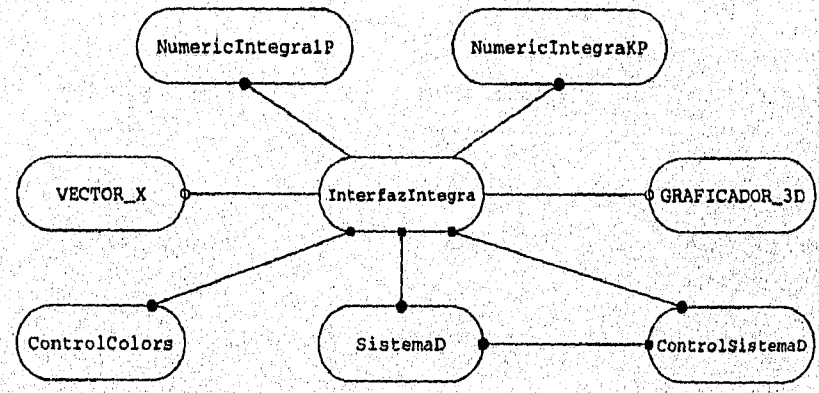


Figura 5.19

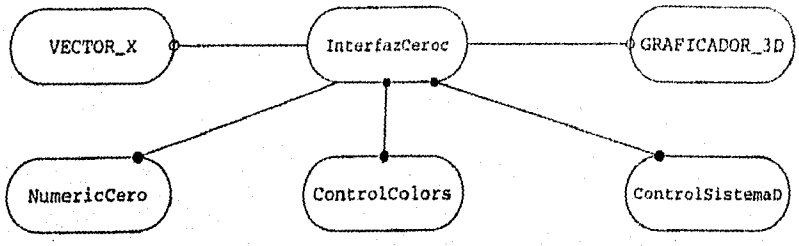


Figura 5.20

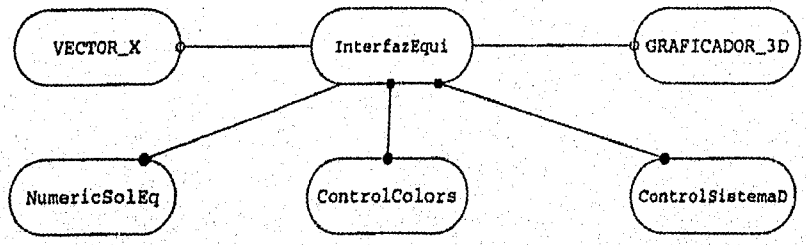


Figura 5.21

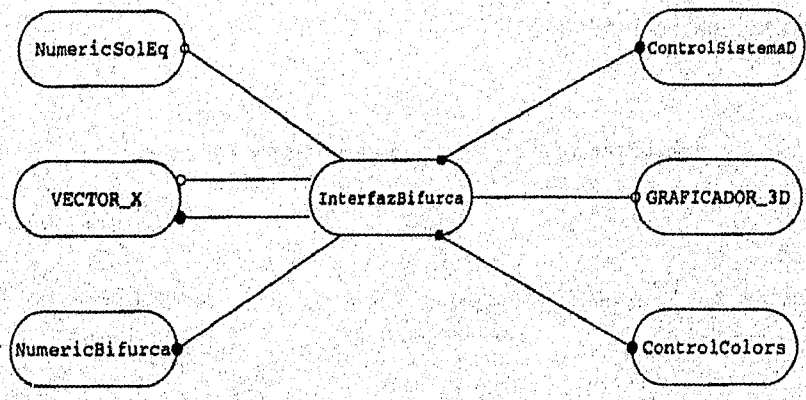


Figura 5.22

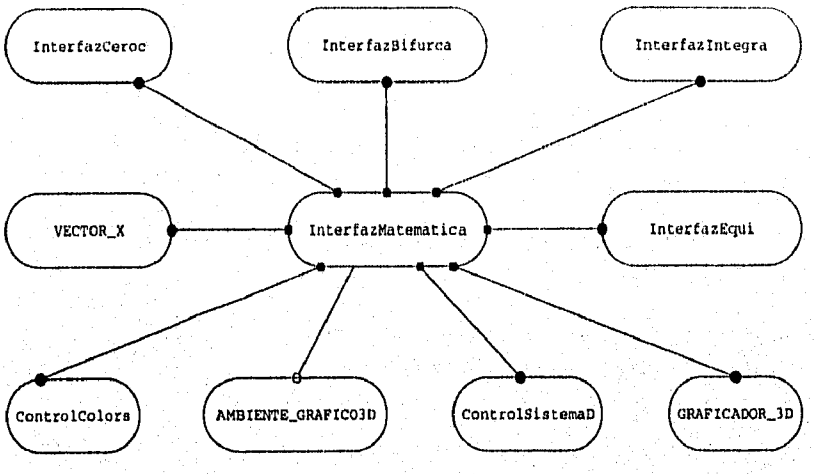


Figura 5.23

El lector interesado podrá ver el código utilizado en C++ para la instrumentación de la jerarquía de clases antes descrita en los archivos de los módulos correspondientes.

BIBLIOGRAFIA

[1] BJARNE STROUSTRUP, *El Lenguaje C++ de Programación,* 2a Edición, Addison-Wesley/Díaz de Santos, Méx. D.F., 1993

[2] GRADY BOOCH, *Object Oriented Analysis and Design with Applications*, 2ª. Edición, The Benjamin Cummins Publishing Company, Santa Clara, California, 1994.

[3] MIGUEL KATRIB MORA, *Programación Orientada a Objetos en C++*, Infosys, Méx. D.F., 1994

[4] TED FAISON, *Borland C++ 3.1, Programación Orientada a Objetos*, 2ª Edición, Prentice Hall Hispanoamericana, Méx. D.F., 1993.

CONCLUSIONES

Los objetivos del presente trabajo fueron concluidos con éxito, en tanto que la instrumentación de un paquete integrado de programas cuyo fin común es el análisis de sistemas dinámicos fué concluida. Se dispone ahora de la aplicación de software BIFURCA con todas las facilidades que permiten el estudio experimental de sistemas dinámicos. La mayoría de métodos fueron implantados satisfactoriamente, a excepción de aquel para encontrar eigenvalores de una matriz. El hecho es que los métodos numéricos encontrados para dichos fines, solamente eran capaces de encontrar eigenvalores reales y, en el caso de los sistemas de ecuaciones diferenciales es muy común tener eigenvalores complejos. La manera en que se resolvió el problema es particularizando a dimensiones 4, 3 y 2 y resolviendo para los polinomios característicos cuyas fórmulas fueron determinadas algebraicamente.

Por su parte, la metodología orientada a objetos para el desarrollo de *software*, pudo ser muy bien aplicada al tipo de problemas que se deseaban resolver. De esta manera, llegué a consolidar bastantes conocimientos obtenidos a lo largo de la licenciatura y lo mejor de todo es que pudieron ser presentados en una manera integral como parte de un todo. La última idea es importante, ya que es muy común que mientras estudia, el alumno no logra asimilar el vínculo que existe entre temas aparentemente diferentes y, como consecuencia es muy probable que no se sienta atraido hacia éstos más que por el reto que implica estudiarlos.

Aunque la aplicación que se hizo de las matemáticas no está directamente vinculada con algún problema del mundo real, considero que representa un ejemplo de lo que son las matemáticas aplicadas ya que indirectamente resuelve problemas reales en la medida en que los sistemas dinámicos son su representación.

En tanto que para incursionar en las matemáticas aplicadas es necesaria una buena dosis de matemática formal, de análisis numérico, y de computación, el egresado de Matemáticas Aplicadas de la ENEP Acatlán tiene una buena formación. Desgraciadamente, en México no se tienen muchas oportunidades de aplicar en la realidad los conocimientos adquiridos, pues no existe una estructura de comunicación debidamente vinculada entre la Industria y la Investigación Científica.