UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

Facultad de Ciencias

ANALISIS COMPUTACIONAL DE SISTEMAS DINAMICOS EN 2 DIMENSIONES

T E S I S

Que para obtener el título de:

M A T E M A T I C O

Presenta:

Salvador Malo Guzmán

1989

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

1. INTRODUCCION

La teoría de sistemas dinámicos† juega un papel fundamental en la modelación de fenómenos evolutivos deterministas, ya sea en forma continua o discreta. Sus orígenes se remontan a la mecánica celeste del siglo pasado y a los trabajos de Poincaré, Liapunov y Birkhoff, entre otros. Rápidamente se descubrió que por lo general es imposible integrar un sistema para encontrar las ecuaciones de todas sus órbitas o trayectorias. El enfoque geométrico cualitativo, iniciado por Poincaré, trata de responder a esta dificultad proporcionando en muchos casos aunque sea una descripción parcial de las características principales del espacio de trayectorias. En los últimos 25 años esta teoría ha tenido un desarrollo notable, debido en gran medida al uso cada vez más frecuente de las computadoras en las matemáticas. Ellas nos permiten explorar fácil y rápidamente una gama enorme de nuevos fenómenos prácticamente inaccesibles mediante las técnicas analíticas convencionales. Aquí se pretende ilustrar la utilidad del método computacional en el análisis de las características cualitativas de algunos tipos fundamentales de sistemas dinámicos en dos dimensiones. En particular se desea obtener información gráfica que ayude a determinar la estabilidad del atractor de Hénon, cuyo comportamiento asintótico constituye un ejemplo clásico de lo que se conoce como dinámica caótica. La idea es desarrollar técnicas que puedan ser implementadas en una computadora y que sean útiles para llegar a una mejor comprensión de los problemas teóricos, lo cual se logra muchas veces mediante el estudio de ejemplos particulares que se consideran representativos de un amplio rango de fenómenos físicos o matemáticos. Nos concentraremos en dos tipos de sistemas dinámicos: mapeos y flujos. Un mapeo F es simplemente una función $F : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ (que supondremos siempre lo suficientemente diferenciable para cualquier técnica que se vaya a aplicar). Ese mapeo genera un sistema dinámico simplemente por iteración, de forma que dado un punto $x_0 \in \mathbb{R}^n$, se tiene $x_1 = F(x_0)$, $x_2 = F(x_1)$, y $x_n = F(x_{n-1})$, $n \geq 1$. La *órbita positiva* de x_0

† A lo largo de este trabajo se identificará la noción de *sistema dinámico* con los modelos matemáticos que lo representen (*i.e.* ecuaciones diferenciales o difeomorfismos, según se trate de un modelo continuo o discreto).

es el conjunto de puntos $\{x_0, x_1, x_2, \ldots\}$. Es muy sencillo obtener numéricamente la órbita de cualquier punto evaluando la función en ese punto y después repitiendo el proceso. Un *flujo* es un sistema dinámico continuo dependiente del tiempo que está dado por las soluciones de una ecuación diferencial ordinaria. Restringiremos nuestra atención al caso de ecuaciones autónomas, de la forma $dx/dt = f(x)$, donde f es una función vector-valuada de $\mathbb{R}^n$ a $\mathbb{R}^n$ que se conoce usualmente como *campo vectorial*. El flujo definido por las soluciones de tales ecuaciones está dado por la función $\varphi : \mathbb{R}^n \times \mathbb{R} \longrightarrow \mathbb{R}^n$, donde $\varphi(x,t)$ es la solución de la ecuación diferencial al tiempo t con condición inicial x en $t = 0$. Así $\varphi(x,0) = x$, $\forall x$. La *órbita* o *trayectoria* por x es el conjunto de puntos $\{\varphi(x,t), -\infty < t < \infty\}$. Como los flujos están dados por ecuaciones diferenciales ordinarias, generalmente son más importantes que los mapeos desde el punto de vista de aplicaciones. Por otro lado son más difíciles de estudiar numéricamente, dado que se requiere de algún tipo de integración numérica del campo vectorial f para calcular la evolución temporal del flujo en la computadora. En §2.1 se dará un método sencillo para hacer esto.

Una de las nociones fundamentales en la teoría de sistemas dinámicos es la de estabilidad, de importancia central en este trabajo. El concepto conocido como *estabilidad estructural* fue introducido por Andronov y Pontryagin [5] en 1937, y ha jugado un papel crucial en el desarrollo de la teoría por más de 30 años, tanto en la escuela rusa como en las de Smale y Thom. A pesar de su utilidad, esta definición tiene algunos graves defectos, como lo son el hecho de que considera a todos los ciclos límites del plano como equivalentes, y el hecho de que atractores no hiperbólicos como los de Lorenz y Hénon no son estructuralmente estables. La definición de estabilidad que nos insteresará aquí es la de Zeeman [1], que se enuncia de forma precisa en §2.1. La idea general, sin embargo, es la siguiente. Dado un campo vectorial v en una variedad X, el flujo de v se obtiene resolviendo la ecuación diferencial ordinaria $\dot{x} = v$. El comportamiento asintótico de este flujo está descrito por sus atractores, y de forma más cuantitativa, por una medida sobre esos atractores. Dada $\varepsilon > 0$, se puede construir una ε-suavización u de esa medida, obteniéndose los picos más altos en las zonas en donde el flujo es más lento (las zonas de atracción). Para construir u de forma más precisa, se usa una ecuación diferencial parcial en X, conocida como la ecuación de Fokker-Planck para v con difusión ε. La función u es entonces el estado estacionario de esa ecuación.

Una vez que se ha construido u se define a dos campos vectoriales como ϵ-equivalentes si sus soluciones u son equivalentes como funciones. El punto clave de la construcción radica en la definición de equivalencia que se dé. Se define a un campo vectorial v como ϵ-estable si tiene una vecindad de ϵ-equivalentes, y como estable si es ϵ-estable para ϵ arbitrariamente pequeñas. En el caso de los sistemas discretos (difeomorfismos), las definiciones son análogas. El comportamiento asintótico del flujo discreto de un difeomorfismo θ está descrito por una medida sobre sus atractores. Dada $\epsilon > 0$, se construye una suavización u de esa medida, y se procede de manera similar al caso de los campos vectoriales. La diferencia de construcción está en la forma de definir la suavización u, que en este caso será la composición de una función de difusión con el mapeo inducido por el mismo difeomorfismo θ.

Antes de dar ejemplos de las formas en las que se puede aplicar el método computacional a estos resultados, habrá que entender bien la teoría. El tercer capítulo de este trabajo se dedicará entonces a su explicación detallada, particularmente en lo que se refiere a la teoría sobre sistemas discretos, en donde se desarrollarán los resultados de [1] para una clase más amplia de funciones. Así pues, se trata de cumplir con un doble propósito. Primero, explicar y extender algunos de los más recientes avances en la teoría de clasificación de sistemas dinámicos, y segundo, ejemplificar la forma en que el análisis computacional puede ayudar a determinar tanto la forma del flujo de un sistema como la naturaleza de sus atractores.

2. HERRAMIENTAS NUMERICAS PARA EL ANALISIS CUALITATIVO DE SISTEMAS BIDIMENSIONALES

§2.1. Retratos fase

Para obtener información cualitativa sobre un fenómeno matemático en la computadora se tienen que generar imágenes y gráficas que de alguna forma capturen las características cualitativas del sistema, y las haga resaltar claramente. En el análisis numérico de los sistemas dinámicos bidimensionales se tienen que resolver dos problemas distintos. El primero se refiere simplemente a la iteración de una función, dada una serie de condiciones iniciales (ya sea para seguir la trayectoria de un punto en particular, o bien para determinar el comportamiento global de toda una región del espacio). En general es muy sencillo llevar a cabo este proceso en la computadora, y por lo tanto sólo se discutirá (en §5) para un caso de especial interés.

El segundo problema es el de obtener información cualitativa sobre las soluciones de las ecuaciones diferenciales ordinarias asociadas a los sistemas continuos. Esto es mucho más que una simple integración numérica de la ecuación. No sólo se quiere conocer el valor de una solución particular a un tiempo dado. Se desea conocer el comportamiento global de todas las soluciones; se quiere determinar la localización de puntos fijos, la existencia de ciclos límites y la dirección del flujo para el conjunto de todas las soluciones de la ecuación. Se desea obtener lo que se conoce como un retrato fase de la ecuación diferencial. La primera dificultad que surge al tratar de hacer esto es la de aproximar un sistema continuo mediante cálculos numéricos discretos, ya que en la computadora no existe ninguna noción de espacio continuo. Dado un campo vectorial v, y una condición inicial x_0, se tiene que escoger una Δt y calcular un nuevo punto $x = x_0 + v\Delta t$ (según el esquema más sencillo de aproximación lineal). Evidentemente hay un error involucrado (de orden de Δt^2), y este error puede irse acumulando hasta llevar a una "falsa" trayectoria solución. Considérese, por ejemplo la siguiente ecuación:

$$\begin{cases} \dot{x} = -y \\ \dot{y} = x \end{cases}$$

cuyas soluciones son círculos concéntricos alrededor del origen. Si se toma como condición inicial $(x_0, y_0) = (1, 0)$ sobre la solución de radio 1 y se integra numéricamente, se obtiene

$$x_1 = x_0 + v_x(x_0, y_0)\Delta t = 1$$

$$y_1 = y_0 + v_y(x_0, y_0)\Delta t = \Delta t$$

de donde $\sqrt{x_1^2 + y_1^2} = \sqrt{1 + \Delta t^2} \neq 1$, de modo que (x_1, y_1) ya no está en el círculo, y la solución que se obtiene es una espiral en lugar de una órbita cerrada, lo cual es completamente distinto desde el punto de vista cualitativo. Surge la pregunta: ¿Cómo saber qué tan cerca está la solución numérica obtenida de la solución real? Evidentemente, si se usa una Δt más pequeña, el error disminuye. Sin embargo, el tiempo que lleva resolver la ecuación aumenta proporcionalmente, y en el caso de algunos campos vectoriales no lineales, la velocidad del flujo puede crecer muy rápidamente, en cuyo caso el escoger una Δt menor no corrige mucho los resultados. Incluso métodos de integración más precisos, como los de Runge-Kutta, pueden fallar.

La solución a este problema consiste en elaborar un programa "inteligente", que sepa reconocer cuándo está cerca de un punto fijo, cerca de un ciclo límite, o bien en una zona del campo vectorial donde la velocidad es muy alta. Esto requiere introducir lo que se denominará como una Δt dinámica, lo cual consiste en re-evaluar la velocidad del flujo en cada iteración y según el caso aumentar o disminuir la Δt que se esté utilizando. En la vecindad de un punto fijo, donde la velocidad del flujo es cercana a cero, si se usa una Δt fija se requiere un número muy grande de iteraciones y un tiempo muy largo para obtener una solución gráfica. Por otro lado, en zonas en donde el campo vectorial tiene un valor absoluto muy alto, una Δt fija puede ocasionar espaciamientos muy grandes entre los puntos calculados y una muy mala aproximación. El algoritmo más sencillo de evaluación y graficación para una trayectoria con Δt dinámica se da a continuación.

```
procedure Calcula_y_Grafica;
begin
  x1 := x2;
  y1 := y2;
  if abs(x2) + abs(y2) > 1 then
    t := t + dt;
    x2 := x1 + vx(x2, y2) * dt;
    y2 := y1 + vy(x2, y2) * dt;
    {Estas 2 líneas pueden sustituirse por un método
     más preciso de integración, como el de Runge-Kutta}
  if (abs(x1 - x2) < 1) and
    (abs(y1 - y2) < 1)
  then dt := 1.5 * dt;
  if (abs(x1 - x2) > 2) or
    (abs(y1 - y2) > 2)
  then dt := 0.2 * dt;
  draw(x1, y1, x2, y2);
end;
```

Para implementar este algoritmo hace falta considerar todavía las dimensiones de la pantalla, los factores de escalamiento, y el signo de t, ya que para obtener la gráfica de toda una trayectoria también hay que seguirla hacia atrás desde la condición inicial. En el Apéndice se incluye el programa en Pascal de un paquete completo de graficación de retratos fase de ecuaciones diferenciales en el plano. En las figuras 2.1 y 2.2 se puede observar la diferencia entre los retratos fase de la ecuación

$$\begin{cases} \dot{x} = x^2 - xy \\ \dot{y} = -y + x^2 \end{cases}$$

obtenidos mediante una Δt fija (Fig. 2.1) y una Δt dinámica (Fig. 2.2). El criterio usado para decidir qué Δt es la correcta en todo momento está basado en la resolución visual del monitor de la computadora. Se escoge Δt de forma que la distancia entre x_i y x_{i+1} sea, en términos del monitor, equivalente a la distancia de dos pixels de la pantalla. Una mejor precisión que ésta no serviría de nada pues no se distinguiría visualmente.

Finalmente, en algunas ocasiones es conveniente saber qué velocidad tiene el flujo en distintos puntos del retrato fase. Para esto se pueden asignar colores distintos a velocidades distintas y graficar las trayectorias usando estos colores. Esto será muy

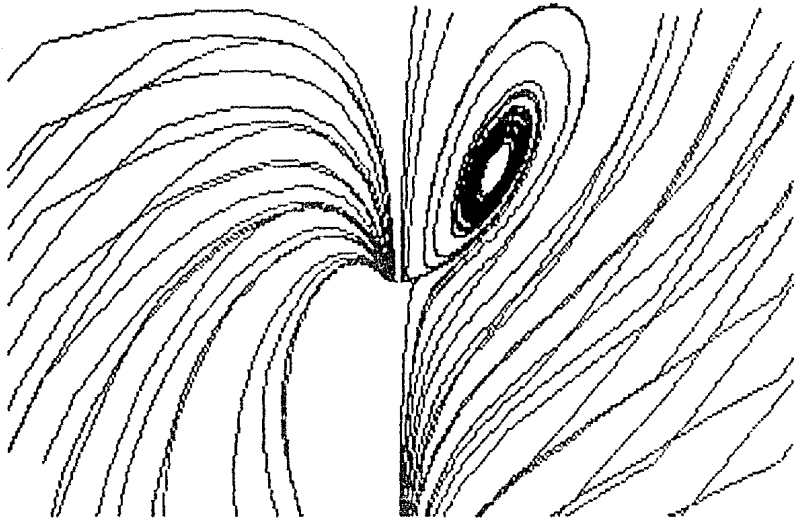

Figura 2.1. Retrato fase de la ecuación $(\dot{x}, \dot{y}) = (x^2 - xy, -y + x^2)$ usando una Δt fija.

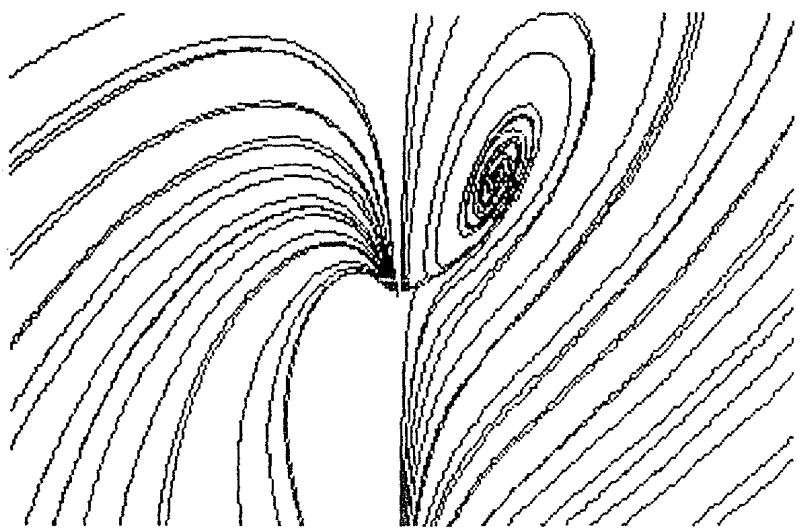

Figura 2.2. Retrato fase de la ecuación $(\dot{x}, \dot{y}) = (x^2 - xy, -y + x^2)$ usando una Δt dinámica.

útil en §4 para determinar la localización de los atractores en el campo vectorial de la ecuación de van der Pol.

§2.2. Diferencias finitas

En la teoría que se desarrolla en la tercera sección de este trabajo juegan un papel muy importante las ecuaciones del calor y de Fokker-Planck, ambas ecuaciones diferenciales parabólicas. En el análisis gráfico-numérico del tipo de sistemas dinámicos descritos aquí será por lo tanto necesario implementar algún método para resolver este tipo de ecuaciones diferenciales parciales. A diferencia de lo que sucede con las ecuaciones diferenciales ordinarias, para las ecuaciones parciales no hay programas, paquetes o métodos generales de solución, y en general su estudio es mucho más complicado. Sin embargo, existen esquemas numéricos generales que pueden adaptarse a los distintos tipos de ecuaciones. Uno de los más usados es el de diferencias finitas, que consiste en remplazar las derivadas parciales por aproximaciones en diferencias. Se analizará el método primero para el caso de una dimensión, ya que después el paso a dos variables es bastante sencillo. La ecuación diferencial parcial general en una variable espacial es

$$\frac{\partial u}{\partial t} = a(x,t)\frac{\partial^2 u}{\partial x^2} + b(x,t)\frac{\partial u}{\partial x} - c(x,t)u,$$

que puede escribirse como

$$\frac{\partial u}{\partial t} = L(x,t,D,D^2)u, \tag{2.1}$$

donde el operador L es lineal y $D = \partial/\partial x$.

Para obtener una ecuación en diferencias equivalente a (2.1) se cubre la región que se va a examinar con una red o malla rectilínea de puntos paralela a los ejes x,t y con separaciones entre los puntos de la malla de h y k para las direcciones x y t respectivamente. Los puntos (X,T) de la malla están dados por $X = mh$, $T = nk$, donde m y n son enteros y $m = n = 0$ es el origen. Las funciones que satisfacen las ecuaciones diferencial y en diferencias en los puntos $X = mh$, $T = nk$ se denotarán por u_m^n y U_m^n respectivamente. Las fórmulas en diferencias (que involucran a dos niveles de tiempo adyacentes) se obtienen de la expansión en serie de Taylor

$$u(x,t+k) = \Big(1 + k\frac{\partial}{\partial t} + \tfrac{1}{2}k^2\frac{\partial^2}{\partial t^2} + \cdots\Big)u(x,t)$$
$$= \exp\Big(k\frac{\partial}{\partial t}\Big)u(x,t).$$

Si denotamos $x = mh$, $t = nk$ y $u(mh, nk) = u_m^n$ entonces, si L es independiente de t, se tiene

$$u_m^{n+1} = \exp\Big(k\frac{\partial}{\partial t}\Big)u_m^n$$
$$= \exp(kL)u_m^n. \tag{2.2}$$

Dada u_m^n se desea conocer u_m^{n+1}, y todos los métodos por diferencias consisten en aproximar (2.2) de una forma u otra. Para cada ecuación en particular habrá que evaluar $\exp(kL)$. Sin embargo, como L generalmente contiene a D y D^2, será útil encontrar expresiones aproximadas para estos dos operadores desde ahora.

El desarrollo en serie de Taylor para $u(x+h,t)$ alrededor de (x,t) nos da

$$u(x+h,t) = u(x,t) + h\frac{\partial u}{\partial x}(x,t) + \frac{h^2}{2!}\frac{\partial^2 u}{\partial x^2}(x,t) + \frac{h^3}{3!}\frac{\partial^3 u}{\partial x^3}(x,t) + O(h^4) \tag{2.3}$$

lo cual, dividido por h resulta en

$$\frac{\partial u}{\partial x}(x,t) = \Big[u(x+h,t) - u(x,t)\Big]\Delta x + O(h).$$

La *diferencia hacia adelante* de la ecuación anterior es la aproximación de primer orden

$$\frac{\partial u}{\partial x} \simeq \Big[u(x+h,t) - u(x,t)\Big]/h + O(h)$$

para $\partial u/\partial x$ evaluada en (x,t). En la notación de subíndices la expresión anterior se escribe como

$$\frac{\partial u}{\partial x}\Big|_{n,m} = \frac{1}{h}[u_{m+1}^n - u_m^n] + O(h). \tag{2.4}$$

Una aproximación alternativa análoga a (2.4) está dada por la serie de Taylor para $u(x-h,t)$:

$$u(x-h,t) = u(x,t) - \frac{\partial u}{\partial x}(h) + \frac{1}{2!}\frac{\partial^2 u}{\partial x^2}(h)^2 - \frac{1}{3!}\frac{\partial^3 u}{\partial x^3}(h)^3 + O(h^4), \tag{2.5}$$

donde todas las derivadas se evaluan en (x,t). Dividiendo por h se obtiene la relación

$$\left.\frac{\partial u}{\partial x}\right|_{n,m} = \frac{1}{h}[u_m^n - u_{m-1}^n] + O(h)$$

que se conoce como *diferencia hacia atrás*, y que también es una aproximación de primer orden en el error $O(h)$.

La forma más sencilla de obtener una aproximación de orden mayor para $\partial u/\partial x$ es restando la ecuación (2.5) de la ecuación (2.3). El resultado, con las derivadas evaluadas en (x,t) es

$$u(x+h,t) - u(x-h,t) = 2h\frac{\partial u}{\partial x} + \frac{h^3}{3}\frac{\partial^3 u}{\partial x^3} + O(h^5),$$

que dividido por $2h$ nos da

$$\left.\frac{\partial u}{\partial x}\right|_{n,m} = \frac{u_{m+1}^n - u_{m-1}^n}{2h} + O(h^2). \tag{2.6}$$

A partir de (2.3) y (2.5) se pueden obtener fácilmente aproximaciones para las derivadas parciales de segundo orden. Por ejemplo, si se suman esas dos ecuaciones se obtiene

$$\frac{1}{h^2}\{u(x+h,t) - 2u(x,t) + u(x-h,t)\} = \frac{\partial^2 u}{\partial x^2} + O(h^2).$$

En la notación de subíndices la expresión anterior se escribe como

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{n,m} = \frac{u_{m+1}^n - 2u_m^n + u_{m-1}^n}{h^2} + O(h^2). \tag{2.7}$$

En dos dimensiones los operadores pueden aproximarse de manera similar. Por ejemplo, el laplaciano, que se usará en §4, puede aproximarse de la siguiente forma:

$$\begin{aligned}\nabla^2 u|_{l,m} &= u_{xx}|_{l,m} + u_{yy}|_{l,m} \\ &= \frac{1}{h^2}\{u_{l+1,m} - 2u_{l,m} + u_{l-1,m} + u_{l,m+1} - 2u_{l,m} + u_{l,m-1}\} + O(h^2) \\ &= \frac{1}{h^2}\{u_{l+1,m} + u_{l-1,m} + u_{l,m+1} + u_{l,m-1} - 4u_{l,m}\} + O(h^2). \end{aligned} \tag{2.8}$$

Dada la frecuencia con que aparecen las aproximaciones para $\partial/\partial x$ y $\partial^2/\partial x$, será conveniente introducir una notación de *operadores de diferencias finitas* dados por

$$\delta_x u_m^n = u_{m+1/2}^n - u_{m-1/2}^n$$

y por

$$\delta_x^2 u_m^n = u_{m+1}^n - 2u_m^n + u_{m-1}^n.$$

En la práctica, para fines de cálculos numéricos, se puede substituir la expresión para $\delta_x u_m^n$ por $\frac{1}{2}(u_{m+1}^n - u_{m-1}^n)$, que es una aproximación equivalente (tomada de (2.6)) que tiene la ventaja de tener subíndices enteros. La expresión original $u_{m+1/2}^n - u_{m-1/2}^n$ será más útil para el desarrollo teórico formal, como se verá a continuación.

Ahora hay que establecer una relación entre δ_x y D. Para esto, defínase el operador de traslación E por $Ef_n = f_{n+1}$. La serie de Taylor

$$f(x+h) = f(x) + \frac{h}{1!}f'(x) + \frac{h^2}{2!}f''(x) + \cdots$$

puede escribirse como

$$\begin{aligned} f(x+h) &= Ef(x) \\ &= \left[1 + \frac{hD}{1!} + \frac{h^2D^2}{2!} + \cdots\right] f(x) \\ &= e^{hD} f(x). \end{aligned}$$

De aquí se deduce que $E = e^{hD}$. Ahora bien, $\delta_x = E^{1/2} - E^{-1/2}$, de modo que

$$\begin{aligned} D &= \frac{2}{h}\operatorname{senh}^{-1}\frac{\delta_x}{2} \\ &= \frac{1}{h}\left(\delta_x - \frac{1^2}{2^2 \cdot 3!}\delta_x^3 + \frac{1^2 \cdot 3^2}{2^4 \cdot 5!}\delta_x^5 \cdots\right). \end{aligned} \tag{2.9}$$

A manera de ejemplo, considérese la ecuación de calor en una dimensión

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial t^2}$$

En este caso

$$L = D^2,$$

y la ecuación (2.2) se vuelve

$$u_m^{n+1} = \exp(kD^2)u_m^n, \tag{2.10}$$

de donde, por (2.9)

$$D^2 = \frac{1}{h^2}(\delta_x^2 - \tfrac{1}{12}\delta_x^4 + \cdots).$$

Sustituyendo D^2 en (2.10) y desarrollando la serie de la exponencial,

$$u_m^{n+1} = [1 + r\delta_x^2 + \tfrac{1}{2}r(r - \tfrac{1}{6})\delta_x^4 + \tfrac{1}{6}r(r^2 - \tfrac{1}{2}r + 1/15)\delta_x^6 \ldots]u_m^n,$$

donde $r = k/h^2$ es el cociente de la malla de puntos. Si se toman las δ_x hasta segundo orden, se obtiene la fórmula

$$U_m^{n+1} = (1 + r\delta_x^2)U_m^n,$$

y al substituir δ_x^2 queda

$$U_m^{n+1} = (1 - 2r)U_m^n + r(U_{m+1}^n + U_{m-1}^n), \tag{2.11}$$

donde U_m^n es una aproximación de u_m^n.

Antes de tratar de implementar este método en la computadora, es necesario saber bajo cuáles condiciones funciona y bajo cuáles falla. Hay que establecer un *criterio de convergencia*. Considérese la fórmula (2.11). Denótese la diferencia entre la solución real y la solución aproximada en el punto (mh, nk) por

$$z_m^n = u_m^n - U_m^n.$$

Por el teorema de Taylor,

$$u_m^{n+1} = u_m^n + k\left(\frac{\partial u}{\partial t}\right)_m^n + \tfrac{1}{2}k^2\left(\frac{\partial^2 u}{\partial t^2}\right)_m^n + \cdots,$$
$$u_{m+1}^n = u_m^n + h\left(\frac{\partial u}{\partial x}\right)_m^n + \tfrac{1}{2}h^2\left(\frac{\partial^2 u}{\partial x^2}\right)_m^n + \tfrac{1}{6}h^3\left(\frac{\partial^3 u}{\partial x^3}\right)_m^n + \tfrac{1}{24}h^4\left(\frac{\partial^4 u}{\partial x^4}\right)_m^n + \cdots,$$

y

$$u_{m-1}^n = u_m^n - h\left(\frac{\partial u}{\partial x}\right)_m^n + \tfrac{1}{2}h^2\left(\frac{\partial^2 u}{\partial x^2}\right)_m^n - \tfrac{1}{6}h^3\left(\frac{\partial^3 u}{\partial x^3}\right)_m^n + \tfrac{1}{24}h^4\left(\frac{\partial^4 u}{\partial x^4}\right)_m^n + \cdots,$$

de modo que

$$
\begin{aligned}
u_m^{n+1} - (1-2r)u_m^n - r(u_{m+1}^n + u_{m-1}^n) = u_m^n &+ k\Big(\frac{\partial u}{\partial t}\Big)_m^n + \tfrac{1}{2}k^2\Big(\frac{\partial^2 u}{\partial t^2}\Big)_m^n - (1-2r)u_m^n \\
&- r\Big[u_m^n + h\Big(\frac{\partial u}{\partial x}\Big)_m^n + \tfrac{1}{2}h^2\Big(\frac{\partial^2 u}{\partial x^2}\Big)_m^n \\
&+ \tfrac{1}{6}h^3\Big(\frac{\partial^3 u}{\partial x^3}\Big)_m^n + \tfrac{1}{24}h^4\Big(\frac{\partial^4 u}{\partial x^4}\Big)_m^n + \cdots \\
&+ u_m^n - h\Big(\frac{\partial u}{\partial x}\Big)_m^n + \tfrac{1}{2}h^2\Big(\frac{\partial^2 u}{\partial x^2}\Big)_m^n \\
&- \tfrac{1}{6}h^3\Big(\frac{\partial^3 u}{\partial x^3}\Big)_m^n + \tfrac{1}{24}h^4\Big(\frac{\partial^4 u}{\partial x^4}\Big)_m^n + \cdots\Big] \\
= k\Big(\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2}\Big)_m^n &+ \tfrac{1}{2}k^2\Big(\frac{\partial^2 u}{\partial t^2} - \frac{1}{6r}\frac{\partial^4 u}{\partial x^4}\Big)_m^n + \cdots
\end{aligned}
$$

Como $\partial u/\partial t = \partial^2 u/\partial x^2$, el primer término de la ecuación anterior desaparece, y se tiene

$$
\begin{aligned}
z_m^{n+1} &= u_m^{n+1} - U_m^{n+1} \\
&= u_m^{n+1} - [(1-2r)U_m^n + r(U_{m+1}^n + U_{m-1}^n)] \\
&= (1-2r)(u_m^n - U_m^n) + r(u_{m+1}^n - U_{m+1}^n + u_{m-1}^n - U_{m-1}^n) \\
&\quad + u_m^{n+1} - (1-2r)u_m^n - r(u_{m+1}^n + u_{m-1}^n) \\
&= (1-2r)z_m^n + r(z_{m+1}^n + z_{m-1}^n) + \tfrac{1}{2}k^2\Big(\frac{\partial^2 u}{\partial t^2} - \frac{1}{6r}\frac{\partial^4 u}{\partial x^4}\Big)_m^n + \cdots
\end{aligned}
$$

Dado que todas las derivadas de u están acotadas, la expresión anterior se puede escribir como

$$
z_m^{n+1} = (1-2r)z_m^n + r(z_{m+1}^n + z_{m-1}^n) + O(k^2 + kh^2). \tag{2.12}
$$

El problema de convergencia para un método de diferencias finitas consiste en encontrar las condiciones tales que $z_m^n \to 0$ uniformemente para un punto fijo $X = mh$, $T = nk$ cuando $h, k \to 0$ y $n, m \to \infty$.

Lema. Si $0 < r \leq 1/2$, entonces $z_m^n \to 0$ bajo las condiciones dadas arriba.

Demostración. Si $0 < r \leq 1/2$, los coeficientes del lado derecho de (2.12) son todos mayores o iguales que cero, y por lo tanto

$$
\begin{aligned}
|z_m^{n+1}| &\leq (1-2r)|z_m^n| + r|z_{m+1}^n| + r|z_{m-1}^n| + A(k^2 + kh^2) \\
&\leq Z^{(n)} + A(k^2 + kh^2),
\end{aligned}
$$

donde A depende de las cotas superiores de $\partial^2 u/\partial t^2$ y $\partial^4 u/\partial x^4$, y $Z^{(n)}$ es el módulo máximo de z_m^n sobre las m. Así pues

$$Z^{(n+1)} \leq Z^{(n)} + A(k^2 + kh^2),$$

y si $Z^{(0)} = 0$ (*i.e.* si la condición inicial es igual para la ecuación diferencial y la aproximación en diferencias) entonces

$$\begin{aligned} Z^{(n)} &\leq nA(k^2 + kh^2) \\ &= TA(k + h^2) \\ &\to 0 \text{ cuando } k, h \to 0 \text{ para } X, T \text{ fijos.} \end{aligned}$$

Por lo tanto el método converge si $0 < r \leq 1/2$.

∎

El paso a dos dimensiones es inmediato. Considérese el operador L para la ecuación de calor dado por

$$L \equiv \frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2} \equiv D_1^2 + D_2^2,$$

donde

$$D_1 \equiv \frac{\partial}{\partial x_1} \text{ y } D_2 \equiv \frac{\partial}{\partial x_2}.$$

Como en el caso de una dimensión, se cubre la región del espacio (x_1, x_2, t) que se va a examinar con una malla rectilinea paralela a los ejes, con divisiones h, k sobre las direcciones del espacio y del tiempo respectivamente. Los puntos de la malla están dados por $X_1 = lh$, $X_2 = mh$, $T = nk$, donde l, m, n son enteros. La función que satisface la ecuación en diferencias en cada punto es $U_{l,m}^n$. La ecuación en diferencias correspondiente a (2.2) es

$$U_{l,m}^{n+1} = \exp(kL) U_{l,m}^n, \tag{2.13}$$

Substituyendo L en (2.13) se obtiene

$$U_{l,m}^{n+1} = \exp(kD_1^2)\exp(kD_2^2) U_{l,m}^n,$$

donde

$$D_1^2 = \frac{1}{h^2}(\delta_{x_1}^2 - \tfrac{1}{12}\delta_{x_1}^4 \ldots)$$

y

$$D_2^2 = \frac{1}{h^2}(\delta_{x_2}^2 - \tfrac{1}{12}\delta_{x_2}^4 \ldots).$$

Eliminando D_1^2 y D_2^2 queda

$$\begin{aligned} U_{l,m}^{n+1} &= [1 + r(\delta_{x_1}^2 + \tfrac{1}{2}r(r - \tfrac{1}{6})\delta_{x_1}^4 \ldots][1 + r(\delta_{x_2}^2 + \tfrac{1}{2}r(r - \tfrac{1}{6})\delta_{x_2}^4 \ldots]U_{l,m}^n \\ &= [1 + r(\delta_{x_1}^2 + \delta_{x_2}^2)]U_{l,m}^n + O(k^2 + kh^2). \end{aligned}$$

En §4 se aplicará esta teoría a una ecuación menos sencilla (i.e. la ecuación de Fokker-Planck), y se discutirá su implementación en la computadora.

3. ESTABILIDAD DE CAMPOS VECTORIALES Y DIFEOMORFISMOS

§3.1. Campos vectoriales y la ecuación de Fokker-Planck

Desde los inicios de la teoría sobre sistemas dinámicos se ha intentado establecer criterios para "clasificar" estos sistemas de alguna manera. Un *programa de clasificación* consiste de cuatro pasos fundamentales:

(*i*) Escoger una relación de equivalencia en el espacio V de todos los campos vectoriales diferenciables,

(*ii*) Definir un campo v como *estable* si tiene una vecindad de equivalentes,

(*iii*) Probar que los sistemas estables son densos, y

(*iv*) Clasificar las clases estables.

El primer esfuerzo realizado en esta dirección fue la teoría desarrollada por Thom para sistemas gradientes

$$v = -\nabla f, \quad f : X \to \mathbb{R},$$

donde X es una variedad diferenciable cualquiera. Se define $f \sim f'$ si existen difeomorfismos α, β tales que el siguiente diagrama conmuta:

$$\begin{array}{ccc} X & \xrightarrow{f} & \mathbb{R} \\ \downarrow{\scriptstyle \alpha} & & \downarrow{\scriptstyle \beta} \\ X & \xrightarrow{f'} & \mathbb{R} \end{array}$$

Según este esquema las funciones estables son las funciones de Morse (*i.e.* aquellas cuyos puntos críticos tienen un hessiano no singular).

Para el caso general (no sólo gradiente), se desarrolló el concepto de estabilidad estructural. Un *flujo* en X es una función $\varphi : X \times \mathbb{R} \to X$, $\varphi(x,t) = \varphi^t(x)$ tal que $t \mapsto \varphi^t$ es una acción de grupo de $\mathbb{R}$ en X. La *órbita* de x es $\varphi(x \times \mathbb{R})$. El campo vectorial v de φ está dado por $v = \partial\varphi/\partial t$, e inversamente φ es el flujo de v. La

topología C^∞ en el espacio de campos vectoriales en X induce una topología en el conjunto de flujos en X. Dos flujos φ, φ' son *topológicamente equivalentes* si existe un homeomorfismo $h : X \to X$ que manda órbitas de φ a órbitas de φ'. Un flujo es *estructuralmente estable* si tiene una vecindad de equivalentes topológicos en el espacio de flujos, y un campo vectorial es *estructuralmente estable* si su flujo lo es.

Esta teoría tiene varios defectos. Los principales son que los sistemas estructuralmente estables no son densos, y por lo tanto no llevan a ninguna clasificación, y que la equivalencia topológica está dada por homeomorfismos, lo cual hace que sistemas cuyo comportamiento es intuitivamente muy distinto caigan dentro de una misma clase de equivalencia. La noción de estabilidad estructural tiene entonces pocos usos en las matemáticas aplicadas.

Las definiciones y resultados que se presentan en este capítulo, y que fueron propuestos por Zeeman, constituyen un enfoque alternativo para este problema. Este enfoque se concentra en hacer resaltar la localización y la forma de los *atractores* de un sistema dado. Para lograr esto se usa un modelo que simule el comportamiento de una población arrastrada por el flujo del sistema. Evidentemente, la población se verá llevada hacia los atractores, en donde se irá acumulando. Si un atractor tiene medida cero (como sucede típicamente), entonces cuando $t \to \infty$ la población en el atractor será infinita. Esto no es muy deseable, ya que es muy incómodo trabajar con cantidades no acotadas. Entonces se introduce un término de difusión en el modelo, de forma que cuando la densidad de población en el atractor sea demasiado alta, ésta empiece a desbordarse. Así, los atractores quedan como cimas de montañas y los repulsores como valles en el espacio X. Este modelo se puede representar mediante una ecuación del tipo conocido como de reacción-difusión: la ecuación de Fokker-Planck.

La ecuación de Fokker-Planck surge en ciertos problemas físicos como ecuación de movimiento para la función de distribución de las variables macroscópicas de un sistema complejo, tales como la función de distribución probabilística del movimiento browniano de un conjunto muy grande de partículas [8]. También es útil sin embargo, para obtener información cualitativa "macroscópica" sobre un sistema dinámico cuyo comportamiento no puede ser descrito en forma exacta debido, al igual que en los sistemas físicos, a su complejidad. La construcción es como sigue.

Sea X una variedad orientada compacta n-dimensional de clase C^∞. Denótese con

$\mathbb{R}_+$ a los reales no negativos. Dado un campo vectorial v clase C^∞ en X, y dada $\varepsilon > 0$, la ecuación de Fokker-Planck para v con difusión ε es la ecuación diferencial parcial

$$\partial u/\partial t = \varepsilon \Delta u - \nabla \cdot (uv) \tag{3.1}$$

donde $u : X \times \mathbb{R}_+ \to \mathbb{R}_+$, $u(x,t) \geq 0$, $x \in X$, $t \geq 0$ y $\int_X u = 1$, $\forall t$.

La función u representa la densidad de una población en X arrastrada por v y sujeta a una difusión ε. El término uv es un campo vectorial que representa a la población u que se mueve con el flujo de v, de modo que la divergencia de uv representa un enrarecimiento de la población debido a v, lo cual contribuye negativamente al crecimiento local de u. Por otro lado, el término $\varepsilon \Delta u$ representa una difusión ε-fuerte, que contribuye positivamente a la velocidad de crecimiento, puesto que el laplaciano de u en x compara el valor promedio de u en puntos vecinos a X y por lo tanto mide el exceso de difusión que entra a partir de esos puntos y lo compara con lo que sale de x.

Sea $u^{v,\varepsilon} : X \to \mathbb{R}_+$ la solución de (3.1) dada por $\partial u/\partial t = 0$. A esta solución se le llama el *estado estacionario* de la ecuación. Como $\varepsilon \Delta u = \nabla \cdot (uv)$, entonces hay un equilibrio entre la fuerza del atractor y la fuerza de la difusión. La población ya no se mueve. Evidentemente, si ε es muy grande el estado de equilibrio será cercano a una función constante, mientras que si $\varepsilon \to 0$ entonces el estado de equilibrio será un pico muy grande sobre el atractor. En [1] se prueba la existencia y unicidad de $u^{v,\varepsilon}$ para X compacto, y se demuestra que es un atractor global de (1), es decir que dada cualquier densidad de población u^0 con $\int u^0 = 1$, se tiene que $u^0 \to u^{v,\varepsilon}$.

Definición. Dos funciones $u, u' : X \to \mathbb{R}$ son *equivalentes*, denotado $u \sim u'$, si existen difeomorfismos α, β en X y $\mathbb{R}$ tales que el diagrama siguiente conmuta

$$\begin{array}{ccc} X & \xrightarrow{u} & \mathbb{R} \\ \downarrow{\scriptstyle \alpha} & & \downarrow{\scriptstyle \beta} \\ X & \xrightarrow{u'} & \mathbb{R} \end{array}$$

Definición. Dos campos vectoriales v, v' en X son *ε-equivalentes* si $u^{v,\varepsilon} \sim u^{v',\varepsilon}$.

Definición. Un campo vectorial es *ε-estable* si tiene una vecindad de ε-equivalentes en el espacio $v(X)$ de campos vectoriales C^∞ en X.

Definición. Un campo vectorial es *estable* si es ε-estable para una $\varepsilon > 0$ arbitrariamente pequeña.

Como ejemplo sencillo considérese $v(x) = -x$, $X = \mathbb{R}$. La ecuación de Fokker-Planck en este caso se vuelve

$$\frac{\partial u}{\partial t} = \varepsilon \frac{\partial^2 u}{\partial x^2} + \frac{\partial}{\partial x}(ux).$$

El estado estacionario está dado por

$$\varepsilon \frac{\partial^2}{\partial x^2} + \frac{\partial}{\partial x}(ux) = 0.$$

lo cual puede escribirse como

$$\frac{d}{dx}\left(\frac{du}{dx} + \frac{ux}{\varepsilon}\right) = 0.$$

Integrando se obtiene

$$\frac{du}{dx} + \frac{ux}{\varepsilon} = A,$$

donde A es una constante. Entonces

$$\frac{d}{dx}(ue^{x^2/2\varepsilon}) = Ae^{x^2/2\varepsilon}.$$

Integrando de nuevo queda

$$u = \left(A \int_0^x e^{y^2/2\varepsilon} dy + B\right) e^{-x^2/2\varepsilon}$$

donde B es una constante. Si $A < 0$ entonces la expresión entre paréntesis tiende a $-\infty$ cuando $x \to \infty$, de modo que $u < 0$ para valores de x suficientemente grandes. Por lo tanto $u \geq 0 \Rightarrow A = 0$. Por otro lado la condición $\int u = 1$ determina el valor de B, y entonces

$$u = \frac{1}{\sqrt{2\pi\varepsilon}} e^{-x^2/2\varepsilon}.$$

Por lo tanto en este caso el estado estacionario es simplemente la distribución normal con media 0 y variancia ε.

§3.2. Estabilidad de difeomorfismos y mapeos C^∞

Esta sección se dedicará a la demostración del teorema principal de este trabajo, el cual establece las condiciones necesarias para extender el programa de clasificación de sistemas continuos al caso discreto. El estudio de los sistemas dinámicos discretos tiene la ventaja de que para éstos los fenómenos caóticos aparecen en espacios de dos dimensiones, lo cual facilita enormemente su análisis. Las definiciones de equivalencia y estabilidad para difeomorfismos son análogas a las de los campos vectoriales, si bien la teoría no es tan elegante. Aunque los fundamentos sobre equivalencia, estabilidad y clasificación se plantean en [1] para difeomorfismos sobre variedades X sin frontera, aquí nos interesará demostrar uno de los teoremas fundamentales para funciones que no son invertibles, es decir para funciones C^∞ en X que no necesariamente son suprayectivas, y para variedades X con frontera. (Esto último es bastante sencillo.) En lugar de utilizar, como se hace en [1], una función de distribución probabilística para probar el resultado, se usará la ecuación de calor $u_t = \Delta u$. Antes que nada, haremos precisa la noción de variedad con frontera.

El *medio-espacio* (cerrado) $\mathbb{H}^n$ está dado por

$$\mathbb{H}^n = \{(x_1, \ldots, x_n) \in \mathbb{R}^n; x_n \geq 0\}.$$

Definición: Una *variedad con frontera* es un espacio métrico M tal que $x \in M \Rightarrow \exists\, V$ vecindad de x y $n \geq 0$ tales que V es difeomorfo ya sea a $\mathbb{R}^n$ o bien a $\mathbb{H}^n$.

Sea entonces X una variedad compacta C^∞. Sea U el espacio de funciones suaves de densidad de probabilidad en X, es decir de funciones $u : X \to \mathbb{R}$ clase C^∞ tales que $\int u = 1$, donde u representa la medida de una población en X. Denótese por $\mathcal{M}$ el espacio de funciones medibles en X. Sea $\theta : X \to X$ una función clase C^∞, tal que $\mu(\theta(X)) > 0$. Entonces θ induce el mapeo $\theta^* : U \to \mathcal{M}$, donde $\theta^* u$ es la medida u transformada por θ como se explica abajo. Dada $\varepsilon > 0$ se construye una función $S : \mathcal{M} \to U$ de suavización, y se considera la composición

$$\begin{array}{c} U \xrightarrow{\theta^*} \mathcal{M} \xrightarrow{S} U. \\ \hline L \end{array}$$

Se probará que L tiene un punto fijo único $u^{\theta,\epsilon}$, y que $\forall u \in U$, $L^n u \to u^{\theta,\epsilon}$ cuando $n \to \infty$. De forma análoga a §3.1, $u^{\theta,\epsilon}$ representa el *estado estacionario* de una población transformada por θ y difundida por ϵ. Dos mapeos θ, θ' en X son *ϵ-equivalentes* si $u^{\theta,\epsilon} \sim u^{\theta',\epsilon}$ como en §3.1.

Definición: $\theta : X \to X$ es *ϵ-estable* si tiene una vecindad de ϵ-equivalentes en el espacio de mapeos C^∞ en X, con la topología C^∞.

Definición: θ es *estable* si es ϵ-estable para $\epsilon > 0$ arbitrariamente pequeña.

Ahora se procederá a definir θ^* y S de manera precisa.

Sea $C(X)$ el espacio de Banach de las funciones reales continuas en X, con la norma del supremo

$$||f|| = \sup\{|f(x)|; x \in X\}.$$

Sea $C(X)^*$ el espacio dual, es decir el espacio de Banach de las funcionales lineales acotadas en $C(X)$. Si $\mu : C(X) \to \mathbb{R}_+$ es una medida en X, entonces $f \in C^\infty$ y X compacto $\Rightarrow \mu(f)$ es acotado. De hecho, $\mu(f) \leq \mu(X)||f||$. Además μ es lineal, por lo tanto las medidas en X pertenecen a $C(X)^*$. Sea $m : \mathcal{M} \to C(X)^*$ la función que asigna a cada $u \in \mathcal{M}$ la medida correspondiente $\mu \in C(X)^*$, dada por

$$mu(f) = \int u(x) f(x) dx, \quad f \in C(X).$$

Dada una función infinitamente diferenciable $\theta : X \to X$, sea $\tilde{\theta} : C(X)^* \to C(X)^*$ dada por

$$\tilde{\theta}\mu(f) = \mu(f\theta) \quad \mu \in C(X)^*, \quad f \in C(X).$$

Lema 3.1. El mapeo $\theta^* : U \to \mathcal{M}$ inducido por $\tilde{\theta}$ está dado, para $x \in \theta(X)$, por

$$\theta^* u(x) = \frac{u(\theta^{-1}x)}{J\theta(\theta^{-1}x)} \quad u \in U,$$

donde $J\theta(y)$ denota el módulo del jacobiano de θ en y.

Demostración. Sea θ^* el mapeo inducido por $\tilde{\theta}$ de forma que el siguiente diagrama conmute

$$\begin{array}{ccc} U & \xrightarrow{\theta^*} & \mathcal{M} \\ \downarrow m & & \downarrow m \\ C(X)^* & \xrightarrow{\tilde{\theta}} & C(X)^* \end{array}$$

Sea $x = \theta(y)$. Entonces $dx = J\theta(y)dy$. Para toda $u \in U$ y $f \in C(X)$ se tiene que

$$\begin{aligned}\int u(y)f(\theta y)dy &= \int u(y)f\theta(y)dy \\ &= mu(f\theta) \\ &= \tilde{\theta}mu(f) \\ &= m\theta^* u(f) \\ &= \int \theta^* u(x)f(x)dx \\ &= \int \theta^* u(x)f(\theta y)J\theta(y)dy.\end{aligned}$$

Como esto es cierto para toda $f \in C(X)$, entonces

$$u(y) = \theta^* u(x)J\theta(y).$$

Por lo tanto

$$\theta^* u(x) = \frac{u(y)}{J\theta(y)} = \frac{u(\theta^{-1}x)}{J\theta(\theta^{-1}x)} \qquad x \in \theta(X).$$

∎

Si θ no es suprayectiva en X, es necesario especificar θ^* para las $x \in X$ donde θ^{-1} no está definida. Sea entonces $\theta^* : U \to \mathcal{M}$ dada por

$$\begin{cases} \dfrac{u(\theta^{-1}x)}{J\theta(\theta^{-1}x)}, & x \in \theta(X) \\ 0 \qquad\quad , & x \notin \theta(X) \end{cases}$$

Para construir la función de suavización S usaremos la solución a tiempo ε de la ecuación de calor $\partial u/\partial t = \Delta u$ con condición inicial $u^0 = \delta_y$, la delta de Dirac † en y. (En [16] se demuestra que la ecuación de calor tiene soluciones para condiciones iniciales y de frontera Lebesgue-integrables, y en principio serviría cualquier u^0 tal que $\int u^0 = 1$). Sea $K_\varepsilon(x, y)$ esta solución. Entonces

$$\int K_\varepsilon(x, y)dx = 1,$$

† Sea $\{f_k\}$ una sucesión de funciones tales que: (i) $f_k \geq 0$ y $f_k = 0$ fuera de una vecindad N_k de un punto fijo y;

(ii) $\int_{N_k} f_k = 1, \ \forall k$;

(iii) $N_k \to \{y\}$ cuando $k \to \infty$.

Entonces la función δ de Dirac se define como $\lim_{k\to\infty} f_k$.

porque la solución de $\partial u/\partial t = \Delta u$ preserva $\int u = 1$. Por otro lado, dado que la solución de la ecuación de calor depende continuamente de la condición inicial, se tiene que $K_t(x,\cdot) \in C(X)$.

Sea $\tilde{S} : C(X)^* \to C(X)$ definida por

$$\tilde{S}\mu(x) = \mu(K_t(x,\cdot)) \qquad \mu \in C(X)^* \quad x \in X.$$

Para toda $\mu \in C(X)^*$, $\tilde{S}\mu$ es diferenciable respecto a x debido a la diferenciabilidad de K_t. Además, $\tilde{S}$ manda medidas de probabilidad a U, porque si μ es una medida de probabilidad entonces

$$\tilde{S}\mu(x) \geq 0 \qquad \forall x \in X$$

y

$$\begin{aligned}\int \tilde{S}\mu(x)dx &= \int \mu(K_t(x,\cdot))dx \\ &= \mu\Big(\int K_t(x,\cdot)dx\Big) \\ &= \mu(1) \\ &= 1.\end{aligned}$$

Lema 3.2. El mapeo $S : \mathcal{M} \to U$ inducido por $\tilde{S}$ está dado por

$$Su(x) = \int K_t(x,y)u(y)dy \qquad u \in U, \quad x \in X.$$

Demostración. Sea S el mapeo inducido, de forma que el siguiente diagrama conmute:

$$\begin{array}{ccc} \mathcal{M} & \xrightarrow{S} & U \\ \downarrow m & & \downarrow c \\ C(X)^* & \xrightarrow{\tilde{S}} & C(X) \end{array}$$

Si $u \in U$, entonces

$$\begin{aligned}Su(x) &= \tilde{S}mu(x) \\ &= mu(K_t(x,\cdot)) \\ &= \int K_t(x,y)u(y)dy.\end{aligned}$$

Ahora considérese la composición

$$C(X) \xrightarrow{\theta^*} M \xrightarrow{S} C(X) \,.$$
$$\xrightarrow[L]{}$$

Como θ^*, S son lineales, y $C(X)$ y M son espacios vectoriales sobre $\mathbb{R}$, entonces L puede verse como operador lineal sobre $C(X)$.

Definición. Un operador es *compacto* si manda conjuntos acotados en conjuntos relativamente compactos, donde por relativamente compacto se entiende un conjunto contenido en un subcojunto del espacio.

Lema 3.3. L es un operador compacto en $C(X)$.

Demostración. Sea F la esfera unitaria en $C(X)$ dada por

$$F = \{f; \|f\| = 1\}.$$

Hay que probar que la imagen de F es relativamente compacta en $C(X)$, ya que cualquier conjunto acotado puede encerrarse en un múltiplo de F, y entonces LF relativamente compacto $\Rightarrow$ L es operador compacto.

Por el teorema de Ascoli,† basta probar

(i) $\{\|Lf\|;\ f \in F\}$ es acotado, y

(ii) LF es equicontinua.

ya que entonces LF está totalmente acotado y como $C(X)$ es completo, la cerradura de LF es compacta y por lo tanto LF es relativamente compacto.

Por los lemas 3.1 y 3.2,

$$\begin{aligned} Lf(x) &= \int_X K_s(x,y)\theta^* f(y)dy \\ &= \int_{\theta(X)} K_s(x,y)\frac{f(\theta^{-1}y)}{J\theta(\theta - 1y)}dy \\ &\leq \int_{\theta(X)} \frac{K_s(x,y)}{J\theta(\theta^{-1}y)}dy. \end{aligned}$$

† **Teorema de Ascoli:** Sea X un espacio compacto, $C(X)$ el espacio de Banach de las funciones continuas en X con la norma del supremo, y $\Phi \subset C(X)$ acotado puntualmente y equicontinuo. Entonces Φ está totalmente acotado en $C(X)$, donde por totalmente acotado se entiende que está contenido en una unión finita de bolas abiertas de radio ϵ, $\forall \epsilon > 0$. (Ver [12]).

Como $J\theta$ es continuo y X es compacto, entonces $J\theta > \delta > 0$ para alguna δ. Por lo tanto $1/J\theta$ está acotado, digamos que por A. Sea $B = \int K_s(x,y)dy$. Como X es compacto y K_s es continua, $B < \infty$. Entonces $|Lf(x)| \leq AB$. Esto demuestra (i).

Para probar (ii) hay que demostrar que $\forall x \in X$, $\forall \eta > 0$, $\exists \delta > 0$ tal que

$$\forall f \in F,\ \forall x' \in X, d(x,x') < \delta \Rightarrow |Lf(x) - Lf(x')| < \eta.$$

Como K_s es C^∞ con respecto a x, existe una cota C (igual a la máxima pendiente de K_s) tal que

$$|K_s(x,y) - K_s(x',y)| \leq Cd(x,x').$$

Sea $\delta = \eta/ABC$. Entonces

$$\begin{aligned} \forall f \in F, \qquad |Lf(x) - Lf(x')| &= \Big| \int_{\theta(X)} (K_s(x,y) - K(x',y)) \frac{f(\theta^{-1}y)}{J\theta(\theta - 1y)} dy \Big| \\ &\leq ABCd(x,x') \\ &< \eta. \end{aligned}$$

Definición. Se define el *cono positivo* P en $C(X)$ por

$$P = \{f;\ f \geq 0\} = \{f;\ f_x \geq 0,\ \forall x \in X\}.$$

El interior de P está dado por

$$\text{int}(P) = \{f;\ f > 0\} = \{f;\ f_x > 0,\ \forall x \in X\}.$$

Se dice que un operador es *fuertemente positivo* si manda $P - \{0\}$ al interior de P.

Lema 3.4. L es fuertemente positivo.

Demostración. Hay que probar que si $f \geq 0$, $f \neq 0$, entonces $\forall x \in X$, $Lf(x) > 0$. Sea entonces

$$I = K_s(x,y) \frac{f(\theta^{-1}y)}{J\theta(\theta^{-1}y)},$$

de forma que $Lf(x) = \int_{\theta(X)} I dx$.

Obsérvese que $I \geq 0$, $\forall y \in \theta(X)$ y que I es continua en $\theta(X)$ por la continuidad de θ. Como $f \neq 0$, $\exists z \in X$ tal que $f(z) \neq 0$. Entonces $f(z) > 0$. Sea $y = \theta z$. Entonces $f(\theta^{-1}y) > 0$. Por lo tanto $I > 0$ para $y = \theta z$.

Como I es continua, entonces $\exists V \subset \theta(X)$ vecindad de y tal que $I > 0$ en V. Pero entonces

$$Lf(x) = \int_{\theta(X)} I dx > 0, \qquad \mu(\theta(X)) > 0.$$

Esto es cierto para toda $x \in X$. Por lo tanto $Lf > 0$.

∎

Observación: Aquí hay que tener mucho cuidado, ya que el lema anterior sólo es válido para funciones θ tales que $\mu(\theta(X)) > 0$, ya que de lo contrario la integral de $Lf(x)$ se estaría evaluando sobre un conjunto de medida 0 y entonces se tendría $Lf(x) = 0$, con lo cual se vendría abajo toda la teoría.

Definición. Un *álgebra de Banach* es un espacio de Banach A que a la vez es álgebra con identidad e, tal que $\forall x, y \in A$

$$||xy|| \leq ||x||\,||y||$$

y tal que $||e|| = 1$.

Definición: Sea A un álgebra de Banach. Sea $G(A)$ el conjunto de todos los elementos invertibles de A. Dado $x \in A$, el *espectro* $\sigma(x)$ de x es el conjunto de todos los números ξ tales que $\xi e - x$ *no* es invertible. A cada $\xi \in \sigma(x)$ se le conoce como *valor espectral* de x. El *radio espectral* de x es el número

$$\rho(x) = \sup\{|\xi| : \xi \in \sigma(x)\}.$$

Lema 3.5. Si $\xi \neq 0$ es un valor espectral de L, entonces ξ es un eigenvalor de L.

Demostración. Sea $L_\xi = L - \xi I$, y $\mathcal{N}(L_\xi)$ el espacio nulo de L_ξ. Si $\mathcal{N}(L_\xi) \neq \{0\}$, entonces ξ es eigenvalor de L. Supóngase que $\mathcal{N}(L_\xi) = \{0\}$, donde $\xi \neq 0$. Entonces $L_\xi x = 0 \Rightarrow x = 0$, y existe $L_\xi^{-1} : L - \xi(C(x)) \longrightarrow C(X)$. Como

$$\{0\} = \mathcal{N}(I) = \mathcal{N}(L_\xi^0) = \mathcal{N}(L_\xi) = \mathcal{N}(L_\xi^2) = \cdots,$$

entonces $X = L_\xi^0(X) = L_\xi(X)$. Por lo tanto L_ξ es biyeciva y L_ξ^{-1} existe en todo $C(X)$ y entonces ξ no es valor espectral, lo cual contradice la hipótesis.

∎

Teorema (Krein-Rutman).† Dado un cono positivo P y un operador lineal compacto L fuertemente positivo con respecto a P, entonces

† Para su demostración ver [6].

(a) L tiene un único eigenvector v interior a P : $\quad Lv = \lambda v \quad (\lambda > 0,\ |v| = 1)$.

(b) El eigenvalor λ asociado a v es de módulo mayor al de todos los otros eigenvalores de L.

Lema 3.6. L tiene un eigenvalor positivo de módulo máximo, y $C(X)$ puede escribirse como la suma de subespacios L-invariantes

$$C(X) = E + H,$$

donde

(i) E es el eigenespacio de λ,

(ii) $\dim(E) = 1$,

(iii) $E \cap \operatorname{int}(P) \neq \emptyset$,

(iv) $L|_H$ tiene radio espectral $< \lambda$.

Demostración. Si E es el eigenespacio correspondiente al vector v del teorema de Krein-Rutman, entonces quedan demostrados (i), (ii), (iii) y (iv).

Por el lema 3.5, los valores espectrales de L son eigenvalores de L, y los eigenvalores de $L|_H$ son todos menores que λ, con lo cual queda probado (v).

∎

Lema 3.7. $\lambda = 1$. *Demostración.* Sea $f \in E \cap \operatorname{int}(P)$. Sea $a = \int f$. Entonces $a > 0$ porque $f > 0$.

Sea $u = a^{-1} f$. Entonces $\int u = a^{-1} \int f = 1$. Como $u \in E$, entonces $Lu = \lambda u$. Por otro lado, Lu es suave porque S, y por ende L, son suavizantes. Entonces λu es suave, lo cual implica que $u \in U$, y $\lambda u = Lu \in LU \subset U$. Por lo tanto $\int \lambda u = 1$ y $\lambda = 1$. ∎

En el lema anterior, cuando X tiene frontera, hay que precisar lo que se entiende por una función suave en ∂X, es decir, hay que definir diferenciabilidad en la frontera de una variedad.

Sea f una función suave de un abierto $V \subset \mathbb{H}^k$ a $\mathbb{R}$. Si $x \in V$, la derivada Df_x se define de la manera usual. Si $x \in \partial V$, entonces como f es suave, se puede extender a una función suave $\tilde{f}$ definida en una vecindad abierta de x en $\mathbb{R}^k$. Defínase Df_n como la derivada $D\tilde{f}_n : \mathbb{R}^k \to \mathbb{R}$. Si $\tilde{\tilde{f}}$ es otra extensión local de f, sea $\{x_i\}$ una sucesión de puntos en $\operatorname{Int}(V)$ que converge a x. Como $\tilde{f}$ y $\tilde{\tilde{f}}$ ambas coinciden con f en $\operatorname{Int}(V)$, entonces

$$D\tilde{f}_{x_i} = D\tilde{\tilde{f}}_{x_i}.$$

Por la continuidad de estas derivadas con respecto a x_i, $x_i \to x \Rightarrow D\tilde{f}_x = D\tilde{\tilde{f}}_x$. Ahora sea $X \subset \mathbb{R}^n$ una k-variedad con frontera, y defínase el *espacio tangente* $T_x(X)$ en $x \in X$ como la imagen de la derivada de una parametrización local de H^k a X alrededor de x.

Lema 3.8. $H = \{h \in C(X); L^n h \to 0 \text{ cuando } n \to \infty\} = \{h \in C(X); \int h = 0\}$.

Demostración. Sea $L_1 = L|_H$ y ρ el radio espectral de $L - 1$. Entonces $\rho < 1$ por el lema 3.6(v) y por el lema 3.7.

Escógase r tal que $\rho < r < 1$. Un resultado conocido para álgebras de Banach (ver por ejemplo Rudin [12]) es que

$$\rho = \lim_{n\to\infty} \sqrt[n]{||L_1^n||}.$$

Por lo tanto $\exists n_0$ tal que

$$\forall n > n_0, \quad \sqrt[n]{||L_1^n||} < r,$$

es decir $||L_1^n|| < r^n$.

Por la definición de la norma de un operador lineal

$$||L_1^n|| = \sup\{||L^n h||/||h||; \quad h \in H\}.$$

Por lo tanto, si $h \in H$ entonces $||L^n h||/||h|| \leq ||L_1^n|| < r^n$. Entonces

$$||L^n|| < r^n ||h||,$$

lo cual implica que $L^n h \to 0$ cuando $n \to \infty$.

Con esto se prueba que $\{h \in C(X); \int h = 0\} \subset H$.

Para demostrar la otra contención, supóngase que $f \in C(X)$ y que $L^n f \to 0$ cuando $n \to \infty$. Por el lema 3.6, sea $f = e + h$, $e \in E$, $h \in H$. Por el lema 3.7, $Le = e$, entonces $e = L^n = L^n f - L^n h \to 0$ cuando $n \to \infty$. Por lo tanto $f \in H$.

Ahora sea $f \in C(X)$. Entonces

$$\begin{aligned}
\int_X \theta^* f(x)dx &= \int_{\theta(X)} \frac{f(\theta^{-1}x)}{J\theta(\theta^{-1}x)}dx && \text{(por definición de } \theta^*) \\
&= \int_X \frac{f(y)}{J\theta(y)} J\theta(y)dy && \text{(haciendo } x = \theta y) \\
&= \int_X f(y)dy.
\end{aligned}$$

Por otro lado

$$\begin{aligned}\int_X Sf(x)dx &= \int_X \left(\int_X K_*(x,y) f(y) dy \right) dx \\ &= \int_X \left(\int_X K_*(x,y) dx \right) f(y) dy \\ &= \int_X f(y) dy.\end{aligned}$$

Entonces

$$\int Lf = \int S\theta^* f = \int \theta^* f = \int f$$

y por lo tanto

$$\int L^n f = \int f.$$

Si $h \in H$ entonces $L^n h \to 0$ cuando $n \to \infty$. Por lo tanto $\int L^n h \to 0$ cuando $n \to \infty$. Por lo tanto $\int h = 0$.

A la inversa supóngase $f \in C(X)$ y $\int f = 0$. Por el lema 3.6 sea $f = e + h$, $e \in E$, $H \in H$. Como $\dim(E) = 1$, se puede escribir

$$e = cu$$

donde u es el punto dado en el lema 3.7, y $c \in \mathbb{R}$.

Entonces $\int e = c \int u = c$. Además ya se probó que $\int h = 0$, entonces $c = \int e + \int h = \int f = 0$. Por lo tanto $e = 0$ y $f = h \in H$. ∎

Ahora se procede a demostrar el teorema principal, que plantea la existencia y unicidad de un estado atractor estacionario $u^{\theta,*}$ para la función θ.

Teorema. (i) $L|_U$ tiene un punto fijo único u.

(ii) Cualquier subconjunto acotado de U converge uniformemente a u.

Demostración. (i) En la demostración del lema 3.7 se construyó un punto $u \in E \cap U$. Como $Lu = u$, entonces u es punto fijo de $L|_U$.

Ahora sea u' un punto fijo cualquiera de $L|_U$. Entonces $u' \in E$, porque es un eigenvector del eigenvalor 1. Entonces $u' = bu$ para alguna $b \in \mathbb{R}$. Por lo tanto $b = \int bu = \int u' = 1$, porque $u' \in U$. Entonces $u' = u$, de forma que el punto fijo de $L|_U$ es único.

(ii) Sea G un subconjunto acotado de U. Hay que probar que $\forall \varepsilon > 0$, $\exists N$ tal que

$$\forall n > N,\ \forall g \in G,\ ||L^n g - u|| < \varepsilon.$$

Sea $G' = \{g - u;\ g \in G\}$. Entonces G' también es acotado, digamos que por k. Si $g \in G$, entonces $g - u \in H$ por el lema 3.8 ya que $\int(g-u) = \int g - \int u = 1 - 1 = 0$. Por lo tanto $G' \subset H$. Finalmente, tomando las mismas r, n_0 que en el lema 3.8, sea N tal que $N > n_0$ y $r^N k < \varepsilon$. Entonces para toda $n > N$ y $g \in G$,

$$\begin{aligned} ||L^n g - u|| &= ||L^n(g-u)|| \\ &\leq r^n ||g-u|| \\ &\leq r^n k \\ &< \varepsilon. \end{aligned}$$

4. SISTEMAS CONTINUOS Y EL CAMPO VECTORIAL DE LA ECUACION DE VAN DER POL

En esta sección se aplicarán los métodos numéricos de §2.1 y §2.2 a la teoría desarrollada en §3.1. Primero que nada se obtendrá una forma explícita para resolver la ecuación de Fokker-Planck en 2 dimensiones por diferencias finitas. Posteriormente se explicará la manera de implementar este método ya en la práctica en la computadora. Se verán algunos ejemplos sencillos en una dimensión y se dará el listado del programa en Pascal que resuelve la ecuación en este caso. Finalmente se analizará el caso bidimensional con el campo vectorial de la ecuación de Van der Pol, y se mostrarán los resultados obtenidos de su estudio numérico. Se escogió este campo vectorial en particular porque es un ejemplo clásico que ha tenido mucha importancia en el estudio de los sistemas dinámicos, y también porque es un caso muy ilustrativo del tipo de características cualitativas que pretende hacer resaltar la teoría que aquí se ha expuesto.

Considérese entonces la ecuación de Fokker-Planck

$$\frac{\partial u}{\partial t} = \varepsilon \Delta u - \nabla \cdot (uv). \tag{4.2}$$

Siguiendo la notación de §2.2, esta ecuación se puede escribir como

$$\frac{\partial u}{\partial t} = Lu, \tag{4.2}$$

en donde el operador L está dado por

$$L = \varepsilon D_x^2 + \varepsilon D_y^2 - aD_x - bD_y - c,$$

y donde $a = v_x$, $b = v_y$, $c = \dfrac{\partial v_x}{\partial x} + \dfrac{\partial v_y}{\partial y}$.

Los operadores D_x, D_y, D_x^2 y D_y^2 están dados por

$$D_x^2 = \tfrac{1}{h^2}(\delta_x^2 - \tfrac{1}{12}\delta_x^4 + \cdots), \qquad \delta_x^2 u_{l,m}^n = u_{l+1,m}^n - 2u_{l,m}^n + u_{l-1,m}^n$$

$$D_y^2 = \tfrac{1}{h^2}(\delta_y^2 - \tfrac{1}{12}\delta_y^4 + \cdots), \qquad \delta_y^2 u_{l,m}^n = u_{l,m+1}^n - 2u_{l,m}^n + u_{l,m-1}^n$$

$$D_x = \tfrac{1}{h}(\delta_x - \tfrac{1}{24}\delta_x^3 + \cdots), \qquad \delta_x u_{l,m}^n = \tfrac{1}{2}(u_{l+1,m}^n - u_{l-1,m}^n)$$

$$D_y = \tfrac{1}{h}(\delta_y - \tfrac{1}{24}\delta_y^3 + \cdots), \qquad \delta_y u_{l,m}^n = \tfrac{1}{2}(u_{l,m+1}^n - u_{l,m-1}^n)$$

Sustituyendo estas expresiones en la ecuación (2.13) se obtiene

$$\begin{aligned} u_{l,m}^{n+1} &= \Big\{1 + k\Big[\frac{\varepsilon}{h^2}(\delta_x^2 - \tfrac{1}{12}\delta_x^4 + \delta_y^2 - \tfrac{1}{12}\delta_y^4 + \cdots) \\ &\quad - \frac{a_{l,m}}{h}(\delta_x - \tfrac{1}{24}\delta_x^3 + \cdots) - \frac{b_{l,m}}{h}(\delta_y - \tfrac{1}{24}\delta_y^3 + \cdots) - c_{l,m}\Big]\Big\} u_{l,m}^n \\ &\simeq u_{l,m}^n + \Big[\varepsilon\Big(\frac{k}{h^2}(\delta_x^2 + \delta_y^2) - \frac{k}{h}a_{l,m}\delta_x - \frac{k}{h}b_{l,m}\delta_y - kc_{l,m}\Big]u_{l,m}^n. \end{aligned}$$

Sustituyendo las expresiones correspondientes para δ_x y δ_y y haciendo $\alpha = \varepsilon(\frac{k}{h^2})$ y $\beta = \dfrac{k}{2h}$ se obtiene la fórmula en diferencias finitas

$$\begin{aligned} U_{l,m}^{n+1} &= (1 - 4\alpha - kc_{l,m})U_{l,m}^n + (\alpha - \beta a_{l,m})U_{l+1,m}^n \\ &\quad + (\alpha + \beta a_{l,m})U_{l-1,m}^n + (\alpha - \beta b_{l,m})U_{l,m+1}^n + (\alpha + \beta b_{l,m})U_{l,m-1}^n \qquad (4.3) \end{aligned}$$

Antes de aplicar esta fórmula, hay que encontrar las condiciones tales que converge a la solución teórica para un punto fijo $X = lh$, $Y = mh$, $T = nk$ cuando $h, k \to 0$ y $l, m, n \to \infty$.

Teorema. La expresión (4.3) converge a la solución de (4.1) si $\alpha \leq \frac{1}{4}$ y $\frac{1}{2}hB \leq \varepsilon$, donde $B = max_{l,m}\{a_{l,m}, b_{l,m}\}$.

Demostración. Denótese la diferencia entre la solución real y la solución aproximada en el punto (lh, mh, nk) por

$$z_{l,m}^n = u_{l,m}^n - U_{l,m}^n.$$

Por el teorema de Taylor,

$$\begin{aligned} u_{l,m}^{n+1} &= u_{l,m}^n + k\Big(\frac{\partial u}{\partial t}\Big)_{l,m}^n + \tfrac{1}{2}k^2\Big(\frac{\partial^2 u}{\partial t^2}\Big)_{l,m}^n + \cdots, \\ u_{l+1,m}^n &= u_{l,m}^n + h\Big(\frac{\partial u}{\partial x}\Big)_{l,m}^n + \tfrac{1}{2}h^2\Big(\frac{\partial^2 u}{\partial x^2}\Big)_{l,m}^n + \tfrac{1}{6}h^3\Big(\frac{\partial^3 u}{\partial x^3}\Big)_{l,m}^n + \tfrac{1}{24}h^4\Big(\frac{\partial^4 u}{\partial x^4}\Big)_{l,m}^n + \cdots, \\ u_{l,m+1}^n &= u_{l,m}^n + h\Big(\frac{\partial u}{\partial y}\Big)_{l,m}^n + \tfrac{1}{2}h^2\Big(\frac{\partial^2 u}{\partial y^2}\Big)_{l,m}^n + \tfrac{1}{6}h^3\Big(\frac{\partial^3 u}{\partial y^3}\Big)_{l,m}^n + \tfrac{1}{24}h^4\Big(\frac{\partial^4 u}{\partial y^4}\Big)_{l,m}^n + \cdots, \\ u_{l-1,m}^n &= u_{l,m}^n - h\Big(\frac{\partial u}{\partial x}\Big)_{l,m}^n + \tfrac{1}{2}h^2\Big(\frac{\partial^2 u}{\partial x^2}\Big)_{l,m}^n - \tfrac{1}{6}h^3\Big(\frac{\partial^3 u}{\partial x^3}\Big)_{l,m}^n + \tfrac{1}{24}h^4\Big(\frac{\partial^4 u}{\partial x^4}\Big)_{l,m}^n + \cdots, \end{aligned}$$

y

$$u_{l,m-1}^n = u_{l,m}^n - h\Big(\frac{\partial u}{\partial y}\Big)_{l,m}^n + \tfrac{1}{2}h^2\Big(\frac{\partial^2 u}{\partial y^2}\Big)_{l,m}^n - \tfrac{1}{6}h^3\Big(\frac{\partial^3 u}{\partial y^3}\Big)_{l,m}^n + \tfrac{1}{24}h^4\Big(\frac{\partial^4 u}{\partial y^4}\Big)_{l,m}^n + \cdots,$$

de modo que

$$
\begin{aligned}
u_{l,m}^{n+1} &- (1-4\alpha-kc_{l,m})u_{l,m}^n - (\alpha-\beta a_{l,m})u_{l+1,m}^n \\
&- (\alpha+\beta a_{l,m})u_{l-1,m}^n - (\alpha-\beta b_{l,m})u_{l,m+1}^n - (\alpha+\beta b_{l,m})u_{l,m-1}^n \\
&= u_{l,m}^n + k\Big(\frac{\partial u}{\partial t}\Big)_{l,m}^n + \tfrac{1}{2}k^2\Big(\frac{\partial^2 u}{\partial t^2}\Big)_{l,m}^n + \cdots - (1-4\alpha-kc_{l,m})u_{l,m}^n \\
&- (\alpha-\beta a_{l,m})\Big[u + h\Big(\frac{\partial u}{\partial x}\Big) + \tfrac{1}{2}h^2\Big(\frac{\partial^2 u}{\partial x^2}\Big) + \tfrac{1}{6}h^3\Big(\frac{\partial^3 u}{\partial x^3}\Big) + \tfrac{1}{24}h^4\Big(\frac{\partial^4 u}{\partial x^4}\Big) + \cdots\Big]_{l,m}^n \\
&- (\alpha+\beta a_{l,m})\Big[u - h\Big(\frac{\partial u}{\partial x}\Big) + \tfrac{1}{2}h^2\Big(\frac{\partial^2 u}{\partial x^2}\Big) - \tfrac{1}{6}h^3\Big(\frac{\partial^3 u}{\partial x^3}\Big) + \tfrac{1}{24}h^4\Big(\frac{\partial^4 u}{\partial x^4}\Big) + \cdots\Big]_{l,m}^n \\
&- (\alpha-\beta a_{l,m})\Big[u + h\Big(\frac{\partial u}{\partial y}\Big) + \tfrac{1}{2}h^2\Big(\frac{\partial^2 u}{\partial y^2}\Big) + \tfrac{1}{6}h^3\Big(\frac{\partial^3 u}{\partial y^3}\Big) + \tfrac{1}{24}h^4\Big(\frac{\partial^4 u}{\partial y^4}\Big) + \cdots\Big]_{l,m}^n \\
&- (\alpha+\beta a_{l,m})\Big[u - h\Big(\frac{\partial u}{\partial y}\Big) + \tfrac{1}{2}h^2\Big(\frac{\partial^2 u}{\partial y^2}\Big) - \tfrac{1}{6}h^3\Big(\frac{\partial^3 u}{\partial y^3}\Big) + \tfrac{1}{24}h^4\Big(\frac{\partial^4 u}{\partial y^4}\Big) + \cdots\Big]_{l,m}^n \\
&= k\Big[\frac{\partial u}{\partial t} - \varepsilon\Big(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\Big) + a\frac{\partial u}{\partial x} + b\frac{\partial u}{\partial y} + cu\Big]_{l,m}^n \\
&\quad + \tfrac{1}{2}k^2\Big[\frac{\partial^2 u}{\partial t^2} + \frac{2\beta}{3k^2}\Big(a\frac{\partial^3 u}{\partial x^3} + b\frac{\partial^3 u}{\partial y^3}\Big) - \frac{\alpha}{6k^2}\Big(\frac{\partial^4 u}{\partial x^4} + \frac{\partial^4 u}{\partial y^4}\Big) + \cdots\Big]_{l,m}^n .
\end{aligned}
$$

Como el primer término satisface (4.1) entonces es igual a cero. Por lo tanto se tiene, tras un desarrollo similar al de §2.2, que

$$
\begin{aligned}
z_{l,m}^{n+1} &= (1-4\alpha-kc_{l,m})z_{l,m}^n + (\alpha-\beta a_{l,m})z_{l+1,m}^n \\
&\quad + (\alpha+\beta a_{l,m})z_{l-1,m}^n + (\alpha-\beta b_{l,m})z_{l,m+1}^n + (\alpha+\beta b_{l,m})z_{l,m-1}^n \\
&\quad + \Big[\tfrac{1}{2}k^2\frac{\partial^2 u}{\partial t^2} + \tfrac{\beta}{6}kh^2\Big(a\frac{\partial^3 u}{\partial x^3} + b\frac{\partial^3 u}{\partial y^3}\Big) - \tfrac{\varepsilon}{12}kh^2\Big(\frac{\partial^4 u}{\partial x^4} + \frac{\partial^4 u}{\partial y^4}\Big) + \cdots\Big]_{l,m}^n . \qquad (4.4)
\end{aligned}
$$

El último término de la expresión anterior puede escribirse como $O(k^2+kh^2)$, ya que tanto las derivadas parciales de u como el campo vectorial $a = v_x, b = v_y$ son acotados. (Nótese que la compacidad del espacio X en el que se está trabajando juega un papel crucial en todo momento.) Sea $Z^{(n)} = max_{l,m}\,|z_{l,m}^n|$, y sea $C = max_{l,m}\,|c_{l,m}|$. Si $\alpha \le \frac{1}{4}$, entonces $\alpha - \beta B = \frac{\varepsilon}{h^2}(\varepsilon - \frac{1}{2}hB) \ge 0$. Como todos los coeficientes de (4.4) son mayores o iguales que $\alpha - \beta B$ entonces todos son positivos. Por lo tanto

$$
\begin{aligned}
|z_{l,m}^{n+1}| &\le (1-4\alpha)|z_{l,m}^n| + kC|z_{l,m}^n| + (\alpha-\beta a_{l,m})|z_{l+1,m}^n| + (\alpha+\beta a_{l,m})|z_{l-1,m}^n| \\
&\quad + (\alpha-\beta b_{l,m})|z_{l,m+1}^n| + (\alpha+\beta b_{l,m})|z_{l,m-1}^n| + O(k^2+kh^2) \\
&\le (1-4\alpha)Z^{(n)} + kCZ^{(n)} + (\alpha-\beta a_{l,m})Z^{(n)} + (\alpha+\beta a_{l,m})Z^{(n)} \\
&\quad + (\alpha-\beta b_{l,m})Z^{(n)} + (\alpha+\beta b_{l,m})Z^{(n)} + O(k^2+kh^2) \\
&= (1+kC)Z^{(n)} + O(k^2+kh^2).
\end{aligned}
$$

Entonces

$$Z^{(n+1)} \leq (1+kC)Z^{(n)} + A(k^2+kh^2)$$

donde A depende de las cotas de $\partial^2 u/\partial t^2$, $\partial^4 u/\partial x^4$ y $\partial^4 u/\partial y^4$.

Como $Z^{(0)} = 0$ (*i.e.* la condición inicial real y la de la fórmula de diferencias son una misma) entonces

$$Z^{(n)} \leq \{1 + (1+kC) + (1+kC)^2 + \cdots + (1+kC)^{n-1}\} A(k^2+kh^2)$$

y como $nk = T$, y

$$(1+kC)^n = \left(1 + \frac{TC}{n}\right)^n < e^{TC},$$

entonces

$$Z^{(n)} \leq e^{TC} An(K^2+kh^2) = Te^{TC}A(k+h^2) \to 0 \text{ cuando } \quad h,k \to 0.$$

∎

Veamos algunos ejemplos en una dimensión. En este caso la ecuación (4.1) se reduce a

$$\frac{\partial u}{\partial t} = \varepsilon \frac{\partial^2 u}{\partial x^2} - v\frac{\partial u}{\partial x} - u\frac{\partial v}{\partial x}$$

y la fórmula equivalente a (4.3) es

$$U_m^{n+1} = (1 - 4\alpha - kc_m)U_m^n + (\alpha - \beta a_m)U_{m+1}^n + (\alpha + \beta a_m)U_{m-1}^n \tag{4.5}$$

donde $\alpha = \varepsilon k/h^2$, $\beta = k/2h$, $a_m = v\Big|_m$, y $c_m = \dfrac{\partial v}{\partial x}\Big|_m$.

La variedad unidimensional compacta y sin frontera más sencilla es el círculo. Si se toma el intervalo $[0, 2\pi]$ y se divide en M partes, la fórmula (4.5) se puede aplicar para $i \in \{0, 1, \ldots, M\}$ identificando los puntos extremos $U_M^n = U_0^n$, de forma que se obtiene un ciclo de puntos que equivale a una representación discreta del círculo. Para simular este esquema numérico en la computadora hay que asignar valores iniciales a cada U_m y aplicar la fórmula (4.5) para todos los puntos, graficando el resultado después de cada iteración. Para lograr esto último se representa el intervalo $[0, 2\pi]$ en la dirección horizontal de la pantalla, y se divide la resolución horizontal H del monitor entre M, de forma que la solución se grafica como una poligonal con vértices $(i\frac{H}{M}, U_i), (i+1)\frac{H}{M}, U_i+1)$. Para M suficientemente grande esta poligonal adopta la apariencia de una curva suave.

El programa en Pascal que resuelve la ecuación de Fokker-Planck en una dimensión se lista a continuación, con parámetros y funciones correspondientes al campo vectorial $v = \cos(x)$.

```
Program  Ecuacion_de_Fokker_Planck_en_el_circulo;
{

                  du/dt   =  e*D(D(x)) +  D(u*v)

                                                                    }
const m = 105;       {numero de divisiones del intervalo espacial}
      n = 1000;      {numero de iteraciones en el tiempo}
      k = 0.03;      {intervalo dt de avance en el tiempo}
      e = 0.02;      {epsilon de difusion}

{ NOTA: Para asegurar una correcta convergencia del metodo, se debe
  cumplir la desigualdad
                                          2
                     4ek  <  (perimetro/m)
                                                                    }

var  U,nuevaU: array [0..m] of real; {aqui se guardan los valores de
                                la solucion numerica en cada punto}
     i,j: integer;
     campo: array[1..2,0..m] of real;   {aqui se guardan los valores
                             de la velocidad y divergencia del campo
                             en cada punto}
     x,integral,h,a,b,correccion,integralinicial: real;
     perimetro,factor: real; {el intervalo espacial es de 0 hasta
                              perimetro}

procedure condicion_inicial; {establece la condicion inicial para
                              empezar a calcular}
begin
     for i := 0 to m do U[i] := 0;
     U[m div 2] := m/perimetro; {funcion delta}
     factor := 100*perimetro/m; {amplificacion para graficacion}
     for i := 0 to m - 1 do     {dibuja la condicion inicial}
     draw (i*round(639/m),199 - round (U[i]*factor),
          (i+1)*round(639/m),199 - round (U[i+1]*factor),1);
end;

function v (x: real): real;  {campo vectorial}
begin
     v := cos(x);
end;
```

```
function Dv (x: real): real; {divergencia del campo vectorial}
begin
     Dv := -sin(x);
end;

BEGIN
     clrscr;
     hires;
     perimetro := 2*pi;
     h := perimetro/m;
     a := e*k/sqr(h);
     b := k/2*h;
     condicion_inicial;
     for i := 0 to m do
     begin
          x := i*perimetro/m;
          campo [1,i] := v(x);
          campo [2,i] := Dv(x);
     end;
          {calcula la integral inicial de la solucion para
          despues poder mantenerla constante}
          integralinicial := 0;
          for i := 0 to m do integralinicial :=
                                   integralinicial + U[i]*h;
     for j := 1 to n do
     begin

          for i := 1 to m - 1 do
          begin
               if keypressed then halt;
               {Algoritmo de diferencias finitas}
               nuevaU[i] := (a - b * campo[1,i]) * U[i+1] +
                            (1 - 2*a - k * campo[2,i])* U[i] +
                            (a + b*campo[1,i])* U[i-1];
          end;
               nuevaU[m] := (a - b * campo[1,m]) * U[0] +
                            (1 - 2*a - k * campo[2,m])* U[m] +
                            (a + b*campo[1,m])* U[m-1];
               nuevaU[0] := (a - b * campo[1,0]) * U[1] +
                            (1 - 2*a - k * campo[2,0])* U[0] +
                            (a + b*campo[1,0])* U[m];
           integral := 0;
           for i := 0 to m do integral := integral + nuevaU[i]*h;
           correccion := integralinicial/integral;
           for i := 0 to m do  nuevaU[i] := nuevaU[i]*correccion;
           for i := 0 to m - 1 do
           begin {borra la solucion anterior y dibuja la nueva}
           draw (i*round(639/m),199 - round (U[i]*factor),
               (i+1)*round(639/m),199 - round (U[i+1]*factor),0);
```

```
            if keypressed then halt;
            draw (i*round(639/m),199 - round (nuevaU[i]*factor),
                (i+1)*round(639/m),199-round (nuevaU[i+1]*factor),1);
            end;
            for i:= 0 to m do U[i] := nuevaU[i];
        end;
END.
```

El campo $v = \cos(x)$ tiene 2 ceros en el intervalo $[0, 2\pi]$. Sin embargo, si se considera la dirección del flujo, se observa fácilmente que sólo uno de ellos $(\pi/2)$ es atractor. Para resolver la ecuación de Fokker-Planck en la computadora hay que asignar valores a los parámetros que deben cumplir las relaciones $\alpha \leq \frac{1}{4}$ y $\frac{1}{2}hB \leq \varepsilon$. Para el caso de $v = \cos(x)$ se tiene que $B = 1$, y entonces los valores $\varepsilon = 0.03$, $h = \pi/52$ y $k = 0.025$ cumplen las desigualdades. (Los valores exactos se escogieron por comodidad y a base de prueba y error para obtener mejores resultados gráficos.) En la figura 4.1 se muestra el resultado de 1000 iteraciones para el campo $v = \cos(x)$ con la condición inicial

$$U_m = \begin{cases} 1/h, & m = M/2 \\ 0, & m \neq M/2 \end{cases}$$

que es un análogo acotado de la función δ y cuya integral es 1. Si se parte de otra condición inicial de integral 1, la gráfica que se obtiene después de 1000 iteraciones es indistinguible de la de la figura 4.1, lo cual no es de sorprenderse, ya que la teoría indica que todas las soluciones tienden a un mismo estado estacionario. Lo que se observa inmediatamente de la figura 4.1 es que hay un máximo justo sobre el punto $\pi/2$, lo cual hacer ver la eficacia del método para localizar los atractores de un sistema dado.

Un ejemplo menos trivial está dado por

$$v = \cos(\tfrac{3}{2}x) + Kx\cos(\tfrac{3}{2}x), \tag{4.6}$$

donde K es una constante $\ll 1$. Este campo tiene dos atractores, uno en $\pi/3$ y otro en $4\pi/3$. Sin embargo, debido al factor Kx, el segundo de ellos es "más fuerte" que el primero. Es interesante ver si la ecuación de Fokker-Planck hace resaltar o no esta diferencia. El segundo término del campo (4.6) tiene el problema de ser discontinuo en un punto, ya que $x\cos(2x)$ tiene valores distintos en 0 y 2π, y si se está considerando el intervalo $[0, 2\pi]$ con $u_0 = u_{2\pi}$ entonces habrá un salto en el valor del campo puesto que $v_0 \neq v_{2\pi}$. Desafortunadamente toda la teoría que se ha desarrollado aquí es para el caso

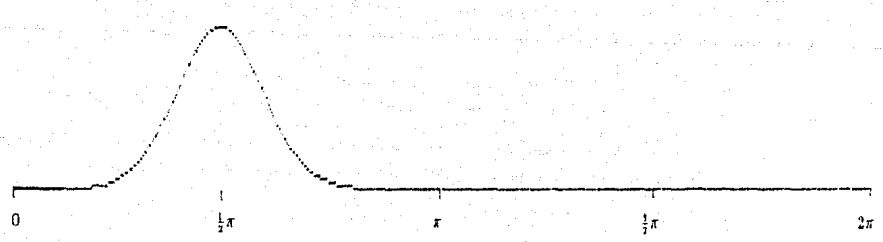

Figura 4.1. Solución de la ecuación de Fokker-Planck en el círculo para el campo $v = \cos(x)$.

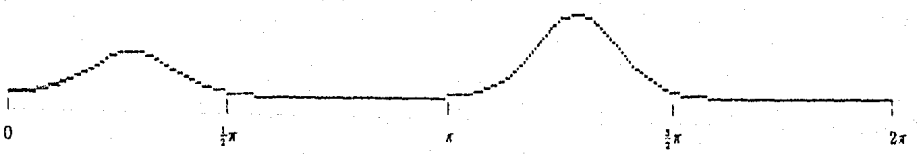

Figura 4.2. Ecuación de Fokker-Planck en el círculo para el campo $v = \cos(\frac{3}{2}x) + Kx\cos(\frac{3}{2}x)$.

donde v es continuo y diferenciable. Sin embargo, el enfoque computacional tiene una ventaja que no existe en los razonamientos teóricos: se puede experimentar, es decir se puede aplicar el programa a un campo discontinuo para ver si funciona. Aún así es recomendable analizar el problema y no proceder simplemente al tanteo.

Ignorando el punto de discontinuidad, lo que queda es el intervalo $(0, 2\pi)$ que no es compacto. Si se aplica la ecuación de Fokker-Planck a ciegas a cualquier campo v discontinuo en la frontera del intervalo, se pueden producir resultados indeseables. Como se indica en [1], la teoría expuesta en §3.1 puede aplicarse a conjuntos no compactos si se aplican ciertas restricciones a v en la frontera. En particular es necesario que la dirección del flujo de v en la frontera no vaya hacia afuera, ya que se produciría un falso atractor en esa zona y la población arrastrada por el flujo tendería a salirse del espacio X. En el caso de (4.6) sin embargo, la dirección del flujo en los extremos 0 y 2π es hacia adentro del intervalo, de modo que, con cierta precaución, podemos ignorar la discontinuidad. En la figura 4.2 se muestra la solución numérica de la ecuación de Fokker-Planck para 1000 iteraciones con $\Delta t = 0.025$. Como puede observarse claramente, hay dos máximos sobre los atractores, y el primero es efectivamente menor que el segundo. La discontinuidad aparentemente no causó problemas. Pasemos ahora al caso bidimensional.

Como ejemplo ilustrativo, sea v el campo vectorial dado por la ecuación de Van der Pol

$$\begin{cases} \dot{x} = y - x^3 + x \\ \dot{y} = -x \end{cases} \tag{4.7}$$

Esta ecuación, que modela un circuito eléctrico sencillo con una resistencia no lineal, ha jugado un papel muy importante en el estudio de la dinámica no lineal, sirviendo como prototipo para distintos fenómenos. Si v es un campo vectorial cualquiera en el plano con flujo ϕ_t y D es una región del plano, entonces [18] se tiene

$$\frac{d}{dt}\Big(\text{area } \phi_t(D)\Big) = \int_D \nabla \cdot v.$$

Esto quiere decir que la divergencia de v mide la velocidad a la que se expanden las áreas conforme se siguen las trayectorias solución. Si se aplica esta fórmula a la ecuación

(4.7) con D_r el disco de radio r, se obtiene

$$\begin{aligned}\frac{d}{dt}\Big(\text{area } \phi_t(D_r)\Big) &= \int_{D_r}(1-3x^2)\\ &= \int_0^r\int_0^{2\pi}(1-3r^2\cos^2\theta)r\,dr\,d\theta\\ &= \pi r^2(1-\frac{3}{4}r^2)\end{aligned}$$

De aquí se ve que sólo el círculo de radio $2/\sqrt{3}$ tiene un área que no cambia con el flujo. Esto indica que la ecuación (4.7) sólo tiene una órbita periódica que está cerca del círculo de radio $2/\sqrt{3}$.

Al tratar de aplicar la fórmula (4.3) con el campo v dado por la ecuación de Van der Pol surge el problema de que la ecuación (4.7) está definida en $\mathbb{R}^2$, mientras que (4.3) sólo se puede aplicar a conjuntos acotados (porque sólo se puede manejar un número finito de U_m^n). Se puede optar por uno de dos caminos. O bien tomar una región rectangular K del plano y trabajar en un conjunto compacto aunque la teoría de §3.1 no incluya variedades con frontera, o bien tomar una variedad sin frontera isomorfa a un rectángulo (*i.e.* un toro) y proyectar el campo (4.7) a esa variedad y aplicar (4.3), lo cual lleva a la misma dificultad que para el ejemplo (4.6) en una dimensión, puesto que el campo vectorial sería discontinuo en las dos líneas del toro correspondientes a la frontera de K. Se pueden encontrar justificaciones para optar por ambas opciones, y al hacer los cálculos en la computadora para los dos procedimientos se observan exactamente los mismos resultados, lo cual es muy alentador. Si se toma $K = [-2,2]\times[-2,2]$ y se combinan las técnicas descritas en §2.1 y §2.2 con un método adecuado de graficación, se puede obtener información sobre los atractores del sistema a través de imágenes como la de la figura 4.3. Aquí se representa la velocidad del flujo del sistema mediante colores, desde azul para las zonas más rápidas hasta violeta para las zonas cercanas a los puntos fijos. Encima del retrato fase se representa la solución de la ecuación de Fokker-Planck correspondiente a este campo vectorial. Inmediatamente destacan dos características de esta solución. La primera es que la densidad de población se acumula en toda la zona correspondiente al ciclo límite, lo cual indica que este ciclo es efectivamente un atractor global del sistema. La segunda es que hay dos máximos en la solución. Por lo tanto no todo el ciclo es igualmente "atractor". Si se observa el retrato fase, se puede ver que el ciclo límite no tiene un color único correspondiente a

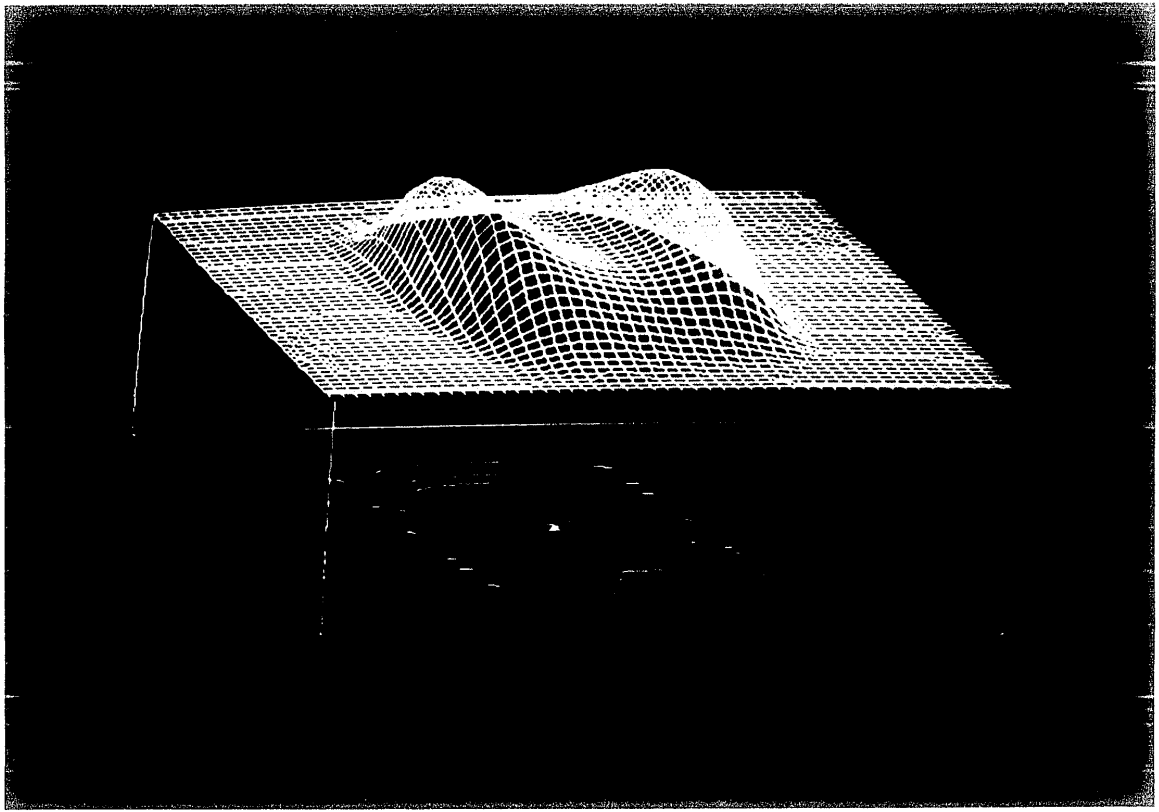

Figura 4.3. Retrato fase de la ecuación de Van der Pol y solución de la ecuación de Fokker-Planck para ese campo vectorial.

una sola velocidad, sino que hay dos ramas lentas y dos ramas rápidas. Los máximos en la densidad de población corresponden a las ramas lentas. El comportamiento de este sistema dinámico es cualitativamente distinto al de otro sistema con un ciclo límite de velocidad uniforme. Las definiciones de equivalencia dadas en §3.1 distinguen claramente entre los dos, mientras que el concepto tradicional de equivalencia topológica no lo hace.

5. ESTUDIO NUMERICO DEL ATRACTOR DE HENON

El progreso de la teoría de sistemas dinámicos puede verse como el desarrollo simultáneo de dos líneas de investigación. Por un lado, la búsqueda de *simplicidad*, estabilidad, determinismo. Por otro lado, el descubrimiento de *complejidad*, inestabilidad, caos. El objetivo general de la teoría es entender qué sucede a tiempos largos; saber cuál es el comportamiento asintótico de los sistemas. Naturalmente, esto lleva a enfocar el estudio hacia el entendimiento de la forma que tienen los atractores.

Uno de los primeros y más famosos ejemplos de dinámica caótica es el siguiente sistema de ecuaciones concebido por Lorenz en 1963:

$$\begin{cases} \dot{x} = -10x + 10y \\ \dot{y} = 28x - y - xz \\ \dot{z} = -\frac{8}{3}z + xy \end{cases}$$

que es una simplificación de un problema relacionado con fenómenos de hidrodinámica. Lorenz descubrió que la mayoría de las trayectorias parecen oscilar en forma impredecible alrededor de dos puntos fijos, y que cualesquiera dos trayectorias con condiciones iniciales muy cercanas tarde o temprano se separan totalmente y adoptan comportamientos completamente distintos. Además, existe una región acotada dentro de la que todas las trayectorias acaban por quedar atrapadas. Todas las órbitas tienden a un conjunto de medida cero que tiene una estructura local de un producto de una 2-variedad por un conjunto de Cantor. Esto se conoce como un *atractor extraño*. La definición precisa se da más adelante.

En 1976, Hénon [3] propuso un sistema dinámico discreto que fuera lo más sencillo posible pero tuviera características esenciales de comportamiento análogas a las de las ecuaciones de Lorenz. El sistema de Hénon tiene dos ventajas obvias. Primero, que es bidimensional, lo cual hace más sencillo su estudio, y segundo, que es un sistema discreto, por lo que los cálculos numéricos son más precisos que para un sistema continuo como el de Lorenz. Los sistemas continuos y los discretos están relacionados de la siguiente manera. Si se consideran exclusivamente las intersecciones de las trayectorias en $\mathbb{R}^3$ con una sección transversal Y, se puede definir una función θ en Y de la siguiente forma: dada $y \in Y$, se sigue la trayectoria que se origina en y hasta que

vuelve a intersectar a Y, denotando por $\theta(y)$ a la nueva intersección. Esta función θ se conoce como *mapeo de Poincaré*. A la inversa, dado un difeomorfismo $\theta : Y \to Y$ de una variedad compacta Y, sea Y_θ el espacio cociente de $Y \times \mathbb{R}$ bajo la relación de equivalencia $(y,s) \sim (\theta^n(y), s-n)$. A partir de esto se obtiene un flujo suave $\{\varphi_t\}_{t\in\mathbb{R}}$ en Y_θ haciendo $\varphi_t[y,s] = [y, s+t]$, donde $[y,s] \in Y_\theta$ denota la clase de equivalencia de $(y,s) \in Y \times \mathbb{R}$.

Dado un difeomorfismo $\theta : \mathbb{R}^n \to \mathbb{R}^n$ con un punto fijo hiperbólico $\bar{x}$ (i.e. un punto fijo tal que $D\theta(\bar{x})$ no tiene eigenvalores de módulo 1), y una vecindad U de $\bar{x}$ se definen las variedades locales estable e inestable $W^s_{\text{loc}}(\bar{x})$ y $W^u_{\text{loc}}(\bar{x})$ como

$$W^s_{\text{loc}}(\bar{x}) = \{x \in U | \theta^n(x) \to \bar{x} \text{ cuando } n \to \infty, \text{ y } \theta^n(x) \in U,\ \forall n \geq 0\},$$

$$W^u_{\text{loc}}(\bar{x}) = \{x \in U | \theta^{-n}(x) \to \bar{x} \text{ cuando } n \to \infty, \text{ y } \theta^{-n}(x) \in U,\ \forall n \geq 0\}.$$

Las variedades globales estable e inestable $W^s(\bar{x})$ y $W^u(\bar{x})$ se definen como

$$W^s(\bar{x}) = \bigcup_{n\geq 0} \theta^{-n}(W^s_{\text{loc}}(\bar{x})),$$

$$W^u(\bar{x}) = \bigcup_{n\geq 0} \theta^n(W^u_{\text{loc}}(\bar{x})).$$

Definición: Dado un flujo continuo o discreto en $\mathbb{R}^n$, un punto p es un punto *ω-límite* de x si existen puntos $\varphi_{t_1}(x)$, $\varphi_{t_2}(x), \ldots$ en la órbita de x tales que $\varphi_{t_i}(x) \to p$ y $t_i \to \infty$. El conjunto ω-límite de x, denotado $\omega(x)$, es la unión de todos sus puntos ω-límite.

Definición: Dado un difeomorfismo θ en $\mathbb{R}^n$ con un punto fijo p, un *punto homoclínico* es un punto $q \neq p$ tal que $q \in W^u(p) \bigcap W^s(p)$. La órbita $\{\theta^n(q)\}$ de q es una *órbita homoclínica*.

Dada una ecuación diferencial en $\mathbb{R}^n$, con un mapeo de Poincaré θ con un punto fijo p, se dice que hay una *órbita transversal homoclínica* si las variedades $W^s(p)$ y $W^u(p)$ se intersecan transversalmente.

Definición: Un *atractor* es un conjunto cerrado invariante Λ con la propiedad de que, dada $\varepsilon > 0$, $\exists U$ en la ε-vecindad de Λ con $\mu(U) > 0$ y tal que $x \in U \Rightarrow \omega(x) \subset \Lambda$ y $\varphi_t(x) \in U$, $\forall t \geq 0$.

A partir de resultados numéricos obtenidos del sistema de Lorenz, Hénon observó que un volumen en $\mathbb{R}^3$ se estira en una dirección al mismo tiempo que se dobla sobre sí mismo en el transcurso de una revolución completa. De estas observaciones pudo simular el mismo efecto mediante la composición de tres mapeos sobre el plano x, y. Considérese una región alargada en la dirección del eje x (Figura 5.1.a). El doblado del área puede hacerse mediante

$$T' : x' = x, \quad y' = y + 1 - ax^2,$$

que produce la forma de la figura 5.1.b; a es un parámetro ajustable. El doblado se completa mediante una contracción sobre el eje x :

$$T'' : x'' = bx', \quad y'' = y',$$

que produce la forma de la figura 5.1.c; b es otro parámetro, que debe ser menor que 1 en valor absoluto. Finalmente se vuelve a la orientación original sobre el eje x mediante

$$T''' : x''' = y'', \quad y''' = x'',$$

de lo que resulta la figura 5.1.d.

Sea $T = T'''T''T'$. Escribiendo (x_i, y_i) en lugar de (x, y) y (x_{i+1}, y_{i+1}) en lugar de (x''', y''') se obtiene el sistema dinámico discreto en $\mathbb{R}^2$,

$$T : x_{i+1} = y_i + 1 - ax_i^2, \quad y_{i+1} = bx_i. \tag{5.1}$$

Una diferencia entre el comportamiento de este sistema y el de las ecuaciones de Lorenz es que los puntos sucesivos que se obtienen mediante la iteración de (5.1) no siempre convergen a un atractor. En ocasiones escapan al infinito. Esto se debe a que el término cuadrático de T domina para distancias grandes del origen.

Se ha visto que el comportamiento del sistema (5.1) puede ser radicalmente distinto para valores distintos de los parámetros a y b. La función T tiene dos puntos invariantes, dados por

$$x = \frac{1}{2a}\left[-(1-b) \pm \sqrt{(1-b)^2 + 4a}\right], \quad y = bx.$$

Estos puntos son reales para

$$a > a_0 = -(1-b)^2/4.$$

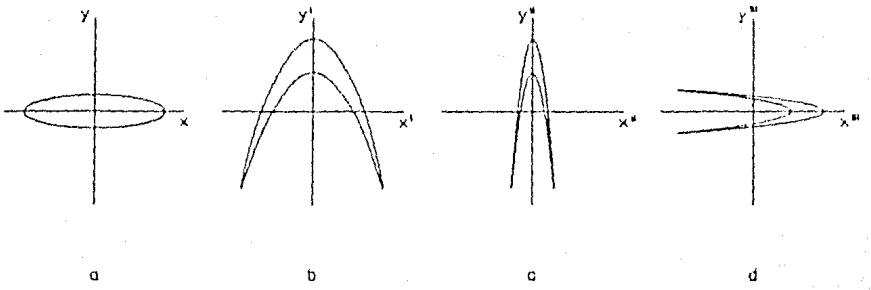

Figura 5.1. El area inicial a es mapeada por T' a b, por T'' a c, y finalmente por T''' a d.

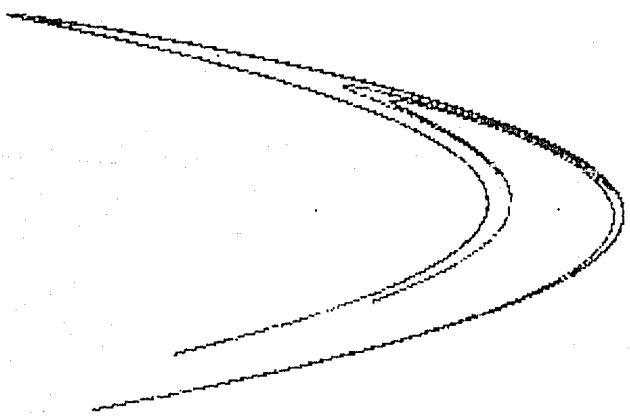

Figura 5.2. 10 000 iteraciones de la función T partiendo de $x_0 = 0$, $y_0 = 0$.

Cuando esto se cumple, uno de los puntos es siempre inestable, mientras que el otro es inestable para

$$a > a_1 = 3(1-b)^2/4.$$

Si se fija el valor de b (por ejemplo $b = 0.3$), entonces para $a < a_0$ todos los puntos tienden a infinito, de modo que no hay atractor. Lo mismo sucede para $a > a_3$ (donde $a_3 = 1.55$ para $b = 0.3$). Cuando se incrementa a por encima de a_0, al principio el atractor sigue siendo sencillo y consiste de un conjunto periódico de p puntos. Conforme aumenta el valor de a, el valor de p sufre una serie de *bifurcaciones* y tiende a infinito cuando $a \to a_2$, donde $a_2 \approx 1.06$ para $b = 0.3$. En cambio, para $a_2 < a < a_3$, el atractor consiste de una infinidad de puntos sin periodicidad y con un comportamiento caótico. En la figura 5.2 se ilustra el resultado de graficar 10000 iteraciones de T para $a = 1.4$, empezando con $x_0 = 0$, $y_0 = 0$. El atractor en este caso parece consistir de una serie de "curvas" más o menos paralelas. Los puntos tienden a distribuirse densamente sobre esas curvas. La *estructura longitudinal* del atractor (a lo largo de las curvas) parece ser sencilla, puesto que cada curva es esencialmente una variedad 1-dimensional. La *estructura transversal* en cambio, es mucho más complicada. Si se amplifica una zona que contenga una sección de curva, se puede observar que se divide en dos o más componentes paralelas, y cada una de éstas a su vez es en realidad un conjunto infinito de curvas. Existe una sucesión jerárquica de "niveles" en la estructura del atractor que es análoga a la estructura de un conjunto de Cantor.

El hecho de que después de miles de iteraciones los puntos no hayan escapado sugiere la existencia de regiones en el plano invariantes bajo T. Efectivamente, tales regiones existen par cada valor de a. Por ejemplo, para el valor $a = 1.4$ usado para calcular la figura 5.2, el cuadrilátero $ABCD$ dado por

$$x_A = -1.33,\ y_A = 0.42,\ x_B = 1.32,\ y_B = 0.133,$$
$$x_C = 1.245,\ y_C = -0.14,\ x_D = -1.06,\ y_D = -0.5,$$

es invariante. La imagen de $ABCD$ es una región acotada por cuatro arcos de parábola, y está totalmente contenida en el cuadrilátero. La figura 5.3 muestra la imagen de $ABCD$ para una, dos y tres iteraciones de T. Nótese que para la tercera iteración (Fig. 5.3.c) la región transformada ya es muy similar al atractor en sí (compárese con la figura 5.2).

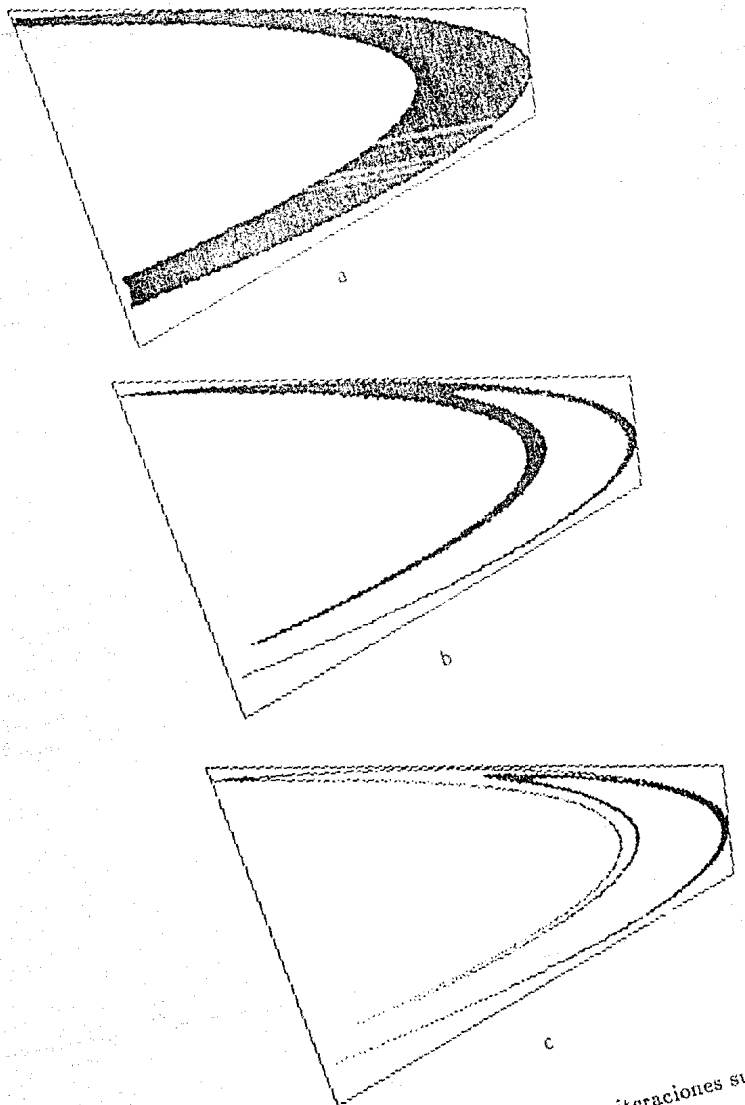

Figura 5.3. Imágenes del cuadrilatero $ABCD$ bajo tres iteraciones succsivas de la funci

A continuación nos interesará averiguar si a partir de esta información numérica se puede concluir algo acerca de la estabilidad del sistema. El problema de la estabilidad de este tipo de atractores ha tenido interés desde hace más de diez años, y hasta la fecha todos los resultados obtenidos en este sentido han sido negativos, ya que se ha usado la noción de estabilidad estructural, según la cual el atractor de Hénon es inestable. [20] La existencia de regiones invariantes permite restringir el estudio de los atractores a conjuntos compactos. Sin embargo, la función T restringida a estos compactos no es suprayectiva, y por eso fue necesario desarrollar la teoría dada en §3.2 para garantizar la existencia de funciones $u^{T,\epsilon}$ que permitieran establecer su estabilidad.

Dado un valor para el parámetro a, y un compacto K invariante (que dependerá de a y será similar al cuadrilátero $ABCD$ descrito arriba), considérese la composición de T con una función de suavización S (i.e. la solución de la ecuación de calor restringida a K y aplicada por un tiempo ε como se indica en §3.2), el estado estacionario $u^{T,\epsilon}$ se puede aproximar numéricamente aplicando T y S repetidas veces al conjunto K. Para implementar la función de suavización S en la computadora se puede utilizar el mismo esquema de diferencias finitas usado para resolver la ecuación de Fokker-Planck. La implementación de T se hace de la siguiente manera. Se examinan los puntos de una región de $\mathbb{R}^2$ que contenga al conjunto invariante K y se revisa se están o no en este conjunto. De ser así se calcula y se grafica la imagen del punto bajo T. El resultado final es la imagen completa del cuadrilátero K. El programa que hace esto, y que fue utilizado para calcular las imágenes de la figura 5.3, se lista a continuación.

```
Program  Atractor_de_Henon;

const a =       {asignar aqui un valor para el parametro a};
      b = 0.3; {valor fijo de b usado para los calculos (puede cambiarse)}
      color = 3;

type matriz = array [0..320,0..100] of integer;
     apuntador = ^matriz;

var espacio1,espacio2,espacioant1,espacioant2,vacio: apuntador;
    {Como el numero de puntos que se manejan es superior a los 64000,
     es necesario utilizar apuntadores para que las estructuras
     quepan en la memoria}
    veces: integer;

function en_cuadrilatero(i,j: integer): boolean;
```

```
 {Decide si el punto que se esta examinando pertenece al cuadrilatero
 invariante K}
var x,y: real;
begin
     x := (i - 159)/107;
     y := (99-j)/200;
     en_cuadrilatero := true;
     if x < -0.2934782*y - 1.2067391 then en_cuadrilatero := false;
     if x > 0.2747252*y  + 1.2834615 then en_cuadrilatero := false;
     if y > -0.1083018*x + 0.2759586 then en_cuadrilatero := false;
     if y < 0.1561822*x  - 0.3344468 then en_cuadrilatero := false;
end;

procedure dibuja_cuadrilatero; {Dibuja el marco de la region invariante}
begin
     draw(round(159+107*(-1.33)),round(99-200*0.42),
          round(159+107*1.32),round(99-200*0.133),color);
     draw(round(159+107*(-1.33)),round(99-200*0.42),
          round(159+107*(-1.06)),round(99-200*(-0.5)),color);
     draw(round(159+107*1.245),round(99-200*(-0.14)),
          round(159+107*(-1.06)),round(99-200*(-0.5)),color);
     draw(round(159+107*1.245),round(99-200*(-0.14)),
          round(159+107*1.32),round(99-200*0.133),color);
end;

procedure condicion_inicial; {Asigna un valor positivo a todos los puntos
                              del espacio}
var i,j: integer;
begin
     new(vacio);
     for j := 0 to 200 do
         for i := 0 to 320 do
         begin
              vacio^[i,j] := 0;
              if j <= 100 then espacio1^[i,j] := 1;
              if j >= 100  then espacio2^[i-100,j] := 1;
         end;
end;

procedure itera;
var i,j,k,coord1,coord2,valant: integer;
    x,y: real;
begin
     espacioant1^ := espacio1^;
     espacioant2^ := espacio2^;
     espacio1^ := vacio^;
     espacio2^ := vacio^;
        for i := 1 to 319 do
            for j := 1 to 199 do
```

```
            begin
                 if keypressed then halt;
                 {Toma cada punto (i,j) de la pantalla, y si corresponde a
                  un punto del cuadrilatero lo convierte a coordenadas reales
                  para calcular su imagen}
                 if en_cuadrilatero(i,j) then
                 begin
                    if j <= 100 then valant := espacioant1^[i,j];
                    if j >= 100 then valant := espacioant2^[i,j];
                    x := (i - 159)/107;
                    y := (99-j)/200;
                    x := y + 1 - a*x*x;
                    y := b*(i-159)/107;
                    coord1 := round(159 + 107*x);
                    coord2 := round(99 - 200*y);
                    {Asigna el valor de cada punto a su imagen}
                    if coord2 <= 100 then espacio1^[coord1,coord2] := valant;
                    if coord2 >= 100 then espacio2^[coord1,coord2] := valant;
                   {dibuja solo los valores distintos de cero, es decir
                   aquellos que corresponden a la imagen del espacio anterior}
                    if j <= 100 then if (espacioant1^[i,j] > 0) then
                                        plot(coord1,coord2,color);
                    if j >= 100 then if (espacioant2^[i,j] > 0) then
                                        plot(coord1,coord2,color);
               end;
            end;
end;


BEGIN
     graphcolormode;
     palette(1);
     new(espacio1); {inicializa los apuntadores}
     new(espacio2);
     new(espacioant1);
     new(espacioant2);
     condicion_inicial;
     for veces := 1 to 3 do {tres iteraciones para lograr las tres imagenes
                             de la figura 5.3}
     begin
          dibuja_cuadrilatero;
          itera;
          if veces < 4 then clearscreen;
     end;
END.
```

Cuando dos parámetros a y a' están suficientemente cerca, se puede escoger un

$u^{T,\epsilon}$ puedan compararse correctamente. Esto hace posible la noción de "vecindad de funciones equivalentes" a una determinada $u^{T,\epsilon}$.

Considérese en primer lugar el intervalo de parámetros $a_0 < a < a_1$. Para estos valores de a el atractor consiste de un solo punto. Evidentemente, dada una perturbación T' de T, las funciones $u^{T,\epsilon}$ y $u^{T',\epsilon}$ tendrán ambas un solo máximo de naturaleza similar y serán difeomorfas, es decir equivalentes. Se puede concluir entonces que para estos valores del parámetro a, T es estable. Para los valores $a_1 \leq a < a_2$, se observan una serie de bifurcaciones de doblamiento de período. Podría pensarse a primera vista que los valores de bifurcación son inestables. Sin embargo no es así. Considérese la primera bifurcación. Para a menor que cierto valor a_i se tiene un período $p = 1$ correspondiente a un punto fijo $\bar{x}$. Para $a > a_i$ se tiene $p = 2$, y hay dos atractores x_1 y x_2. Si $a > a_i$, se observa que cuando $a \searrow a_i$, $x_1 \to \bar{x}$ y $x_2 \to \bar{x}$, de modo que $|x_1 - x_2|$ puede hacerse tan pequeño como se quiera escogiendo a cerca de a_i. Dada $\varepsilon > 0$, si los dos puntos x_1 y x_2 están suficientemente cerca, entonces al aplicar una función de suavización los dos atractores se reflejarán como un solo máximo en la función $u^{T,\epsilon}$. Esto puede entenderse como sigue. Si se usa la solución de la ecuación de calor como función de suavización, entonces dada una condición inicial de dos funciones delta muy cercanas (Fig. 5.4.a) la solución de la ecuación de calor se encarga de ir llenando el espacio entre ellas conforme avanza el tiempo (Fig. 5.4.b). Para un tiempo suficientemente largo, los dos picos iniciales se habrán transformado en un solo máximo (Fig. 5.4.c). Ahora bien, dada $\varepsilon > 0$ tan pequeña como se quiera, es posible tomar dos deltas lo suficientemente cercanas como para que aparezca un solo máximo en un tiempo menor que ε. Pero esto es justo lo que se necesita para que T sea estable en el valor de parámetro a_i, ya que para cualquier $\varepsilon > 0$ existe una vecindad de parámetros alrededor de a_i para los cuales la función $u^{T,\epsilon}$ es casi idéntica, ya que dos atractores cercanos se comportan como uno solo bajo la suavización. Como el argumento se aplica al resto de las bifurcaciones, se puede concluir que T es estable para todos los valores de a entre a_1 y a_2.

Para $a_2 \leq a < a_3$ se observa un comportamiento sumamente complejo. Parece haber regiones de caos intercaladas con "ventanas" de periodicidad. Este fenómeno no es del todo nuevo, ya que algo muy similar se observa en el caso de los mapeos unidimensionales como el siguiente:

$$f_a(y) = 1 - ay^2, \quad 0 < a \leq 2,$$

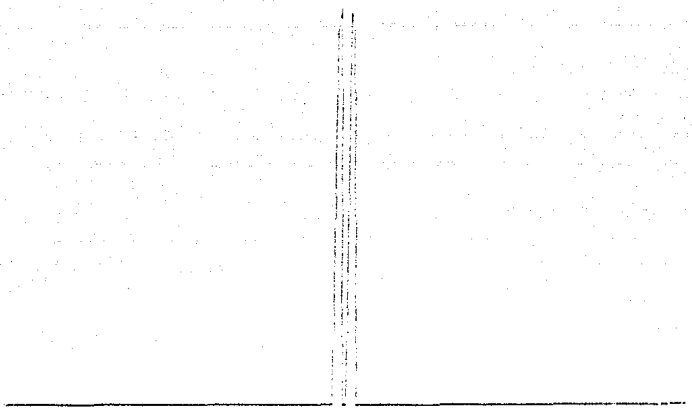

Figura 5.4.a. Condición inicial de dos funciones δ para la ecuación de calor en un intervalo compacto.

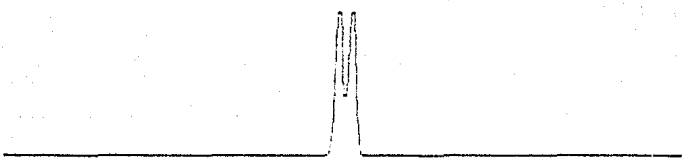

Figura 5.4.b. Solución después de 0.075 unidades de tiempo.

Figura 5.4.c. Solución después de 0.2 unidades de tiempo.

que han sido estudiados bastante más que los sistemas en dos dimensiones.[21] Aunque pocos de los resultados en una dimensión pueden aplicarse directamente al caso bidimensional, hay algunas características que se manifiestan de forma muy similar en ambos tipos de sistemas. Por ejemplo, se observa numéricamente que parece haber una infinidad de ventanas de comportamiento periódico en el espacio de parámetros. Por otro lado, se observa también lo que parecen ser regiones de caos, sin puntos fijos ni periodicidad. A partir de simples observaciones gráficas no se pueden hacer deducciones formales ni teoremas. Sin embargo, se pueden analizar las distintas posibilidades de comportamiento del sistema (que gracias a los cálculos numéricos se han reducido bastante), y sacar conclusiones para cada caso.

Supóngase que existen intervalos de la forma $I = (a_i, a_j)$ en el espacio de parámetros tales que para $a \in I$ hay exclusivamente órbitas caóticas. Dado un valor cualquiera a para el cual el sistema manifieste un comportamiento irregular, y experimentando numéricamente, se observa lo siguiente. Existen perturbaciones de a suficientemente pequeñas tales que la imagen gráfica del atractor no cambia. Parece ser que lo único que varía es la manera en que los puntos se distribuyen sobre el atractor, y no el atractor en sí. Una vez más, hay que recalcar que estas observaciones son puramente visuales, y podría ser que el sistema tuviera cambios drásticos en su estuctura transversal imperceptibles a bajas resoluciones. Aún así, debido a la pequeña escala a la que se tendrían que manifestar, y dadas las características de la función de suavización ya descritas al discutir los puntos de bifucación , estos cambios no se reflejarían en la función $u^{T,\epsilon}$ para una perturbación suficientemente pequeña. Por lo tanto, si existen intervalos de caos "puro", estos intervalos son regiones estables de T.

Desgraciadamente, esto es lo más que se puede decir en favor de la estabilidad del atractor de Hénon. A partir de aquí, todas las observaciones que se pueden hacer indican la existencia de un conjunto infinito de parámetros para los cuales el sistema es inestable. Se trata de aquellos puntos de frontera entre las regiones periódicas y las caóticas. En estos puntos la menor perturbación precipita la función $u^{T,\epsilon}$ de tener un número finito de máximos a tener la forma de una distribución suave encima de unas curvas parabólicas. La pregunta es, qué tan grande es este conjunto de parámetros inestables. ¿Es denso?

Existen dos formas alternativas de comportamiento para el sistema. La primera

posibilidad sería que los valores del parámetro a correspondientes a un comportamiento caótico fueran sólo la frontera entre dos regiones de periodicidad. En ese caso sólo se tendría un comportamiento caótico para un conjunto de parámetros de medida cero. Los experimentos numéricos parecen contradecir esta hipótesis, puesto que se observa "caos" para mucho más valores de a de lo que se esperaría si el conjunto fuera de medida cero. Sin, embargo, esto podría deberse simplemente a errores de aproximación de la computadora, los cuales harían que un punto saltara a través de un número indefinido de periodicidades, pasando por órbitas que en realidad no le corresponden, y dando la falsa impresión de caos. También podría suceder que las ventanas de periodicidad fueran en la mayoría de los casos tan estrechas, o de períodos tan altos, que fueran esencialemente indistinguibles de una órbita caótica.

La segunda alternativa sería que efectivamente hubiera conjuntos abiertos de valores de a con comportamiento exclusivamente caótico. Ya se ha visto que para esos puntos el atractor sería estable, como también son estables los intervalos correspondientes a órbitas periódicas. Sólo quedarían, como en el caso anterior, los puntos de frontera entre regiones de caos y regiones de periodicidad, los cuales podrían seg ir siendo densos en ciertos intervalos. en ambos casos los valores problemáticos se reducen a un conjunto de medida cero.

Como se puede ver, no hay evidencia suficiente que ayude a probar la densidad del conjunto de valores inestables del sistema. Se sabe muy poco todavía acerca del atractor de Hénon. De hecho, la existencia misma de un atractor extraño para T sigue siendo un problema abierto.[4] Sin embargo, a manera de conclusión, se puede establecer la siguiente

Conjetura: El atractor de Hénon (5.1) es estable según la definición de §3.2 para un conjunto denso de parámetros, y los valores inestables forman un conjunto de medida cero.

Por muchos años el trabajo en sistemas dinámicos se ha desarrollado bajo un principio conocido como el *dogma de estabilidad*, según el cual un modelo de un sistema físico es útil sólo si sus propiedades cualitativas son invariantes bajo perturbaciones. Generalmente se exigía la condición de estabilidad estructural para considerar a un sistema como un "buen" modelo de un fenómeno dado. Recientemente se descubrieron defectos e insuficiencias tanto en el concepto de estabilidad estructural como en el dogma de

estabilidad, y ya no se confía en ellos tan ciegamente. Sin embargo, se sigue buscando una cierta robustez en los modelos, y asícomo la estabilidad, definida de un modo o de otro, seguirá jugando un papel primordial en la teoría de los sistemas dinámicos, la computación seguirá siendo el instrumento ideal para su estudio.

BIBLIOGRAFIA

[1] Zeeman E C 1988 Stability of dynamical systems *Nonlinearity* **1** 115–55

[2] López de Medrano S, Watts C & Zeeman E C 1988 Stable diffeomorphisms are dense *Preprint, University of Warwick*

[3] Hénon M 1976 A two-dimensional mapping with a strange attractor *Commun. Math. Phys.* **50** 69–77

[4] Guckenheimer J & Holmes P 1983 *Nonlinear oscillations, dynamical systems, and bifurcations of vector fields* (New York: Springer)

[5] Andronov A A & Pontryagin L 1937 Systèmes grossiers *Dokl. Acad. Nauk. SSSR* **14** 247–51

[6] Krein M G & Rutman M A 1950 Linear operators leaving invariant a cone in a Banach space *Amer. Math. Soc. Translations* **10** 199-325

[7] Ames W F 1977 *Numerical methods for partial differential equations* (New York: Academic)

[8] Risken H 1984 *The Fokker-Planck equation* (Berlin: Springer)

[9] Garabedian P R 1964 *Partial differential equations* (New York: Wiley)

[10] Mitchell A R & Griffiths D F 1980 *The finite difference method in partial differential equations* (Chichester: Wiley)

[11] Conway J 1985 *A course in functional analysis* (New York: Springer)

[12] Rudin W 1973 *Functional analysis* (New York: McGraw-Hill)

[13] Courant R & Hilbert D 1962 *Methods of mathematical physics, Vol. II* (New York: Wiley)

[14] Kreyszig E 1978 *Introductory functional analysis* (New York: Wiley)

[15] Guillemin V & Pollack A 1974 *Differential topology* (New Jersey: Prentice-Hall)

[16] Widder D V 1975 *The heat equation* (New York: Academic)

[17] Gerald C F & Wheatley P O 1984 *Applied numerical analysis* (Massachussetts: Addison-Wesley)

[18] Guckenheimer J 1980 Dynamics of the Van der Pol Equation *IEEE Trans. Circ. & Syst.* **27** 983-9

[19] Hirsch M W 1984 The dynamical systems approach to differential equations *Bull. Am. Math. Soc.* **11** 1-64

[20] Mañé R 1987 A proof of the C^1 stability conjecture *Publ. Math. IHES*

[21] Collet P & Eckmann J P 1980 *Iterated maps of the interval as dynamical systems* (Boston: Birkhauser)

APENDICE: PAQUETE DE GRAFICACION DE RETRATOS FASE

```
Program    Graficacion_de_ecuaciones_diferenciales_en_el_plano;
           {Version en Turbo Pascal}

type cadena_de_100 = string[100];
     arreglo_polaco = array [1..100] of cadena_de_100;

label otra_vez;

Var  Matriz: Array [1..9,1..50] of Real;
     num_imag,num_archivos,Numero_de_condiciones,
     Densidad,Punto,recsread,ya: Integer;
     origenx,origeny,amplificacion,s,
     cota,Factor1,Factor2,a,b,e,k,contador_de_flecha: Real;
     polaca_x, polaca_y: arreglo_polaco;
     screenbuffer: array [1..16128] of byte;
     pantalla: file;
     tecla: char;
     campo_x,campo_y,campo_vectorial_x, campo_vectorial_y : cadena_de_100;
     sobrepuestas,flechasino,numero,filename, nombre_de_archivos: string[12];

procedure Graphics;                                    external 'graph.bin';
procedure HiRes;                                       external Graphics[6];
procedure HiResColor(Color: Integer);                  external Graphics[9];
procedure GraphBackground(Color: Integer);             external Graphics[15];
procedure GraphWindow(X1,Y1,X2,Y2: Integer);           external Graphics[18];
procedure Plot(X,Y,Color: Integer);                    external Graphics[21];
procedure Draw(X1,Y1,X2,Y2,Color: Integer);            external Graphics[24];
procedure GetPic(var Buffer; X1,Y1,X2,Y2: Integer);    external Graphics[36];
procedure PutPic(var Buffer; X,Y: Integer);            external Graphics[39];
function  GetDotColor(X,Y: Integer): Integer;          external Graphics[42];
procedure ClearScreen;                                 external Graphics[60];

{-------------------------------------------------------------------------------}

function  SUFIJA (expresion: cadena_de_100): cadena_de_100;

var  infija,              {expresion infija original}
     polaca,             {expresion sufija final}
     temp,
     temp2:  string[100];  {cadenas temporales}
     alphab: string[100];  {cadena de caracteres alfabeticos}
     i,psp: integer;       {contadores}
     invalida: boolean;     {indicador de expresion invalida}

type cadena = string[100];

{Las siguientes funciones (f,g,r) determinan el orden de precedencia
 de los caracteres de la expresion}

{-------------------------------------------------------------------------------}
```

```
function  f (simbolo: cadena): integer;

var simbolo_char: cadena_de_100;  {simbolo que se va a analizar}
begin
     if pos (simbolo,alphab) <> 0
     then f := 7
     else begin
               simbolo_char := copy (simbolo,1,1);
               case simbolo_char of
                    '+','-' :  f := 1;
                    '*','/' :  f := 3;
                        '^' :  f := 6;
                        '(' :  f := 9;
                        ')' :  f := 0;
                        ' ' :  f := -1;
               end
          end
end;

{-------------------------------------------------------------------------------}

function  g (simbolo: cadena): integer;

var simbolo_char: cadena_de_100;
begin
     if pos (simbolo,alphab) <> 0
     then g := 8
     else begin
               simbolo_char := copy (simbolo,1,1);
               case simbolo_char of
                    '+','-' :  g := 2;
                    '*','/' :  g := 4;
                        '^' :  g := 5;
                        '(' :  g := 0
               end
          end
end;

{-------------------------------------------------------------------------------}

function  r (simbolo: cadena): integer;
begin
     if pos (simbolo,alphab) <> 0
     then r := 1
     else r := -1                  {distingue las variables de los operadores}
end;

{-------------------------------------------------------------------------------}

function unaria (simbolo,infijita: cadena): cadena;
begin
     if (simbolo = '-') or (simbolo = 'l') or
        (simbolo = 'e') or (simbolo = 'c') or (simbolo = 's') then
     begin
          unaria := 'no';
          case simbolo of
               '-': if i = 1 then unaria := '\';
               'l': begin
                         if copy (infijita,1,1) = 'n' then
                         unaria := '&';
                    end;
               'e': begin
                         if copy (infijita,1,1) = 'x' then
```

```
                    if copy (infijita,2,1) = 'p' then
                    unaria := '$';
               end;
          'c': begin
                    if copy (infijita,1,1) = 'o' then
                    if copy (infijita,2,1) = 's' then
                    unaria := '>';
               end;
          's': begin
                    if (copy (infijita,1,1) = 'i') or
                    (copy (infijita,1,1) = 'e') then
                    if copy (infijita,2,1) = 'n' then
                    unaria := '<';
               end;
     end;
end
     else unaria := 'no';
end;

{-------------------------------------------------------------------------}

procedure  convierte (infija: cadena; var polaca: cadena);


var  stack: array[1..20] of string[100];   { el "stack" }
     nuevo: string[100];                   {simbolo que se esta analizando}
     tope,                                 {tope del stack}
     contador: integer;

{sub-procedure de convierte}
procedure  detecta_operadores_unarios;

          {Detecta el menos unario, asi como las funciones sen,
                                                        cos,
                                                        ln,
                                                        exp}
begin  {detecta_operadores_unarios}
                    {revisa si es un operador unario}
                    temp := unaria (nuevo,infija);
                    if temp <> 'no' then
                    {si es unaria entonces dependiendo del tipo de
                     funcion quita los caracteres necesarios de infija}
                    begin
                         case temp of
                              '&': infija := copy (infija,2,100);
                        '<','>','$': infija := copy (infija,3,100);
                         end; {del case}
                         repeat
                         begin
                              nuevo := copy (infija,1,1);
                              infija := copy (infija,2,100);
                         end;
                         until nuevo <> ' ';
                         if nuevo = '(' then
                         begin
                              i := 0;
                              psp {parentesis sin pareja} := 1;
                              while psp > 0 do
                              begin
                                   i := i + 1;
                                   temp2 := copy (infija,i,1);
                                   if temp2 = '(' then psp := psp + 1;
                                   if temp2 = ')' then psp := psp - 1;
                              end;
```

```
                    temp2 := sufija (copy (infija,1,i-1));
                    polaca := concat (polaca,temp2);
                    polaca := concat (polaca,temp);
                    nuevo := copy (infija,i + 1, 1);
                    infija := copy (infija,i + 2,100);
                    if nuevo = ' ' then
                    repeat
                    begin
                         nuevo := copy (infija,1,1);
                         infija := copy (infija,2,100);
                    end;
                    until nuevo <> ' ';
                    i := 0;
               end {del if nuevo = '('}
               else begin
                         if temp <> '\' then
                         begin
                              invalida := true;
                              textcolor(12);
                              writeln ('Expresion invalida');
                              textcolor(15);
                         end
                         else begin
                                   polaca := concat (polaca,nuevo);
                                   polaca := concat (polaca,temp);
                                   nuevo := ' ';
                                   repeat
                                   begin
                                      nuevo := copy (infija,1,1);
                                      infija := copy (infija,2,100);
                                   end;
                                   until nuevo <> ' ';
                              end;
                    end;

                    {Al finalizar esta rutina, debe considerarsele
                     como si hubiera sido un caracter alfabetico}
                    contador := contador + 1;

          end; {del if temp <> 'no'}
end;


begin {convierte}
     {inicializa el stack}
     tope := 1;
     stack[tope] := '(';

     {inicializa la expresion sufija}
     polaca := '';
     contador := 0;
     i := 0;
     nuevo := ' ';

     {toma el primer simbolo que no sea espacio en blanco}
     repeat
     begin
          nuevo := copy (infija,1,1);
          infija := copy (infija,2,100);
          if nuevo = '-' then i := 1;
     end;
     until nuevo <> ' ';

     {traduce la expresion infija}
```

```
while (nuevo <> '#') and not invalida do
begin
     {quita del stack los simbolos de mayor prioridad}
     if tope = 0
     then begin
               invalida := true;
               textcolor(12);
               writeln ('Expresion invalida');
               textcolor(15);
          end
     else while ( f (nuevo) < g (stack[tope]))
           and not invalida do
           begin
                temp := stack[tope];
                tope := tope - 1;
                polaca := concat (polaca,temp);
                contador := contador + r (temp);
                if contador < 1
                then begin
                          invalida := true;
                          textcolor (12);
                          writeln ('Expresion invalida');
                          textcolor (15);
                     end
            end;

     {revisa si la expresion es valida}
     if not invalida
     then begin
          detecta_operadores_unarios;
          i := i - 1;
          {revisa si el siguiente simbolo es un parentesis derecho}
          if f (nuevo) <> g (stack[tope])
          then
               begin
                    tope := tope + 1;
                    stack[tope] := nuevo;
               end
          else tope := tope - 1;

          {toma el siguiente simbolo que no sea espacio en blanco}
          if length (infija) > 0
          then begin
                    nuevo := ' ';
                    repeat
                    begin
                         nuevo := copy (infija, 1, 1);
                         infija := copy (infija, 2, 100);
                         detecta_operadores_unarios;
                         if nuevo = '(' then i := 2 else i := 0;
                    end; {del repeat}
                    until nuevo <> ' ';
               end {del if length (infija) > 0}
          else nuevo := '#';
          end {del if not invalida}
end;

{revisa si la expresion es valida}
if not invalida
then if (tope > 0) or (contador <> 1)
     then begin
          invalida := true;
          textcolor (12);
          writeln ('Expresion invalida');
```

```
                textcolor (15);
                end
end;

{-------------------------------------------------------------------------------}

begin {sufija}

     {inicializa el indicador de invalido y la cadena de alfabeto}

     invalida := false;
     alphab := 'abekxy123456789';

     {convierte la expresion a notacion polaca}
     infija := concat(expresion,')');
     convierte (infija,polaca);
     if not invalida
     then sufija := polaca + '#'
     else sufija := '#';
end; {sufija}

{-------------------------------------------------------------------------------}

procedure  preprocess (cadena: cadena_de_100; var polacarray: arreglo_polaco);

var x: string[1];
    i,posicion: integer;

begin
     textcolor(14);
     writeln;
     posicion := 0;
     for i := 1 to length (cadena)  - 1 do
     begin
          posicion := posicion + 1;
          case copy (cadena,posicion,1) of
               'k': if k = 0 then
                    begin
                         write ('k = ');
                         readln (k);
                         str (k: 20, polacarray [posicion]);
                    end
                    else str (k: 20, polacarray [posicion]);
               'a': if a = 0 then
                    begin
                         write ('a = ');
                         readln (a);
                         str (a: 20, polacarray [posicion]);
                    end
                    else str (a: 20, polacarray [posicion]);
               'b': if b = 0 then
                    begin
                         write ('b = ');
                         readln (b);
                         str (b: 20, polacarray [posicion]);
                    end
                    else str (b: 20, polacarray [posicion]);
               'e': if e = 0 then
                    begin
                         write ('e = ');
                         readln (e);
                         str (e: 20, polacarray [posicion]);
                    end
                    else str (e: 20, polacarray [posicion]);
```

```
          else polacarray [posicion] := copy (cadena,posicion,1);
          end; {case}
          x := copy (polacarray [posicion],1,1);
          while x = ' ' do
          begin
               polacarray [posicion] := copy (polacarray [posicion],2,100);
               x := copy (polacarray [posicion],1,1);
          end;
      end;  {for to}
      polacarray [posicion + 1] := '#';
end; {preprocesa}

{-------------------------------------------------------------------------}

var tope,i,j,basura: integer;
    simbolo: cadena_de_100;
    stack: array [1..100] of real;

function f1(x,y: real): real;

begin
     tope := 0;
     i := 1;
     simbolo := campo_x[i];
     while simbolo <> '#' do
     begin
          case simbolo of
               'x': begin
                         tope := tope + 1;
                         stack [tope] := x;
                    end;
               'y': begin
                         tope := tope + 1;
                         stack [tope] := y;
                    end;
               'e': begin
                         tope := tope + 1;
                         stack [tope] := e;
                    end;
               'a': begin
                         tope := tope + 1;
                         stack [tope] := a;
                    end;
               'b': begin
                         tope := tope + 1;
                         stack [tope] := b;
                    end;
               'k': begin
                         tope := tope + 1;
                         stack [tope] := k;
                    end;
               '+': begin
                         stack [tope -1] := stack [tope -1] + stack [tope];
                         tope := tope - 1;
                    end;
               '-': begin
                         stack [tope -1] := stack [tope -1] - stack [tope];
                         tope := tope - 1;
                    end;
               '*': begin
                         stack [tope -1] := stack [tope -1] * stack [tope];
                         tope := tope - 1;
                    end;
               '\': stack [tope] := - (stack [tope]);
```

```
            '^': if frac(stack [tope]) = 0 then
                 begin
                      stack [tope + 1] := stack [tope - 1];
                      for j := 1 to round (stack [tope]-1) do
                      stack [tope - 1]:= stack [tope - 1]*stack [tope + 1];
                      tope := tope - 1;
                 end
                 else begin
                 if stack [tope - 1] > 0 then
                 stack [tope -1] := exp(stack [tope] * ln(stack [tope -1]))
                 else if stack [tope - 1] = 0 then
                 stack [tope - 1] := 0
                 else if frac (stack [tope]/2) <> 0 then
                 stack [tope -1] := -exp(stack [tope]*ln(-stack [tope -1]))
                 else
                 stack [tope -1] := exp(stack [tope]*ln(-stack [tope -1]));
                 tope := tope - 1;
                 end;
            '/': begin
                      stack [tope -1] := stack [tope -1] / stack [tope];
                      tope := tope - 1;
                 end;
            '<': stack [tope] := sin (stack [tope]);
            '>': stack [tope] := cos (stack [tope]);
            '&': stack [tope] := ln (stack [tope]);
            '$': stack [tope] := exp (stack [tope]);
            else begin
                      tope := tope + 1;
                      val (simbolo,stack [tope],basura);
                 end;
        end; {case}
    i := i + 1;
    simbolo := campo_x [i];
    end; {while}
    f1 := stack [tope];
end; {f}

function f2(x,y: real): real;

begin
    tope := 0;
    i := 1;
    simbolo := campo_y[i];
    while simbolo <> '#' do
    begin
        case simbolo of
            'x': begin
                      tope := tope + 1;
                      stack [tope] := x;
                 end;
            'y': begin
                      tope := tope + 1;
                      stack [tope] := y;
                 end;
            'e': begin
                      tope := tope + 1;
                      stack [tope] := e;
                 end;
            'k': begin
                      tope := tope + 1;
                      stack [tope] := k;
                 end;
            'a': begin
                      tope := tope + 1;
```

```
                    stack [tope] := a;
                end;
            'b': begin
                    tope := tope + 1;
                    stack [tope] := b;
                end;
            '+': begin
                    stack [tope -1] := stack [tope -1] + stack [tope];
                    tope := tope - 1;
                end;
            '-': begin
                    stack [tope -1] := stack [tope -1] - stack [tope];
                    tope := tope - 1;
                end;
            '*': begin
                    stack [tope -1] := stack [tope -1] * stack [tope];
                    tope := tope - 1;
                end;
            '\': stack [tope] := - (stack [tope]);
            '^': if frac(stack [tope]) = 0 then
                begin
                    stack [tope + 1] := stack [tope - 1];
                    for j := 1 to round (stack [tope]-1) do
                    stack [tope - 1]:= stack [tope - 1]*stack [tope + 1];
                    tope := tope - 1;
                end
                else begin
                if stack [tope - 1] > 0 then
                stack [tope -1] := exp(stack [tope] * ln(stack [tope -1]))
                else if stack [tope - 1] = 0 then
                stack [tope - 1] := 0
                else if frac (stack [tope]/2) <> 0 then
                stack [tope -1] := -exp(stack [tope]*ln(-stack [tope -1]))
                else
                stack [tope -1] := exp(stack [tope]*ln(-stack [tope -1]));
                tope := tope - 1;
                end;
            '/': begin
                    stack [tope -1] := stack [tope -1] / stack [tope];
                    tope := tope - 1;
                end;
            '<': stack [tope] := sin (stack [tope]);
            '>': stack [tope] := cos (stack [tope]);
            '&': stack [tope] := ln (stack [tope]);
            '$': stack [tope] := exp (stack [tope]);
            else begin
                  tope := tope + 1;
                  val (simbolo,stack [tope],basura);
             end;
        end; {case}
    i := i + 1;
    simbolo := campo_y [i];
    end; {while}
    f2 := stack [tope];
end; {f}

{-------------------------------------------------------------------------}

procedure parametros;
begin
    a := 0;
    b := 0;
    e := 0;
    k := 0;
```

```
     campo_x := sufija(campo_vectorial_x);
     campo_y := sufija(campo_vectorial_y);
     if (campo_x <> '#') and (campo_y <> '#') then
     begin
     preprocesa (campo_x, polaca_x);
     preprocesa (campo_y, polaca_y);
     end;
end;

{-------------------------------------------------------------------------}

procedure  ecuacion_nueva;

var
    i: integer;

label no;

Begin
     no:
     clrscr;
     textcolor(9);
     writeln ('-Escribir cualquier numero racional como p/q.');
     writeln ('-Usar las constantes a,b para expresar numeros mayores que 10');
     writeln (' y numeros en forma decimal');
     writeln ('-Expresar los parametros o constantes de perturbacion con k,e.');
     writeln;
     writeln (' EJEMPLO:   dx/dt = (3/2)*x - a*y');
     writeln;
     writeln ('            dy/dt = y - k*x^2');
     writeln;
     writeln ('            a = 20');
     writeln ('            k = 0.0154');
     writeln;
     textcolor(15);
     writeln;
     write ('dx/dt = ');
     readln (campo_vectorial_x);
     writeln;
     write ('dy/dt = ');
     readln (campo_vectorial_y);
     textcolor(14);
     parametros;
     if (campo_x = '#') or (campo_y = '#') then
     begin
          delay (2000);
          goto no;
     end;
End;

{-------------------------------------------------------------------------}

Procedure  Evalua;
Begin
     matriz [5,punto] := matriz [5,punto] + matriz [6,punto];
     Matriz [2,Punto] := Matriz [2,Punto] + s * Matriz [9,Punto] *
                         f1(Matriz [2,Punto]/s + origenx,
                           Matriz [4,Punto]/s + origeny
                           ) * Matriz [6,Punto];

     Matriz [4,Punto] := Matriz [4,Punto] + s * Matriz [9,Punto] *
                         f2(Matriz [2,Punto]/s + origenx,
                           Matriz [4,Punto]/s + origeny
                           ) * Matriz [6,Punto];
```

```
End;

{-------------------------------------------------------------------------------}

Procedure  Dibuja;
Begin
     Draw (2*Round(Matriz [2,Punto]) + 319, 99 - Round(Matriz [4,Punto]),
           2*Round(Matriz [1,Punto]) + 319, 99 - Round(Matriz [3,Punto]),1);
End;

Procedure  Condiciones_iniciales;
var a,b: integer;

Begin
     Punto := 0;
     Numero_de_condiciones := SQR(2 * Densidad + 1);
     For a := -Densidad to Densidad do
     Begin
          For b := -Densidad to Densidad do
          Begin
               Punto := Punto + 1;
               Case Densidad of
               1: Begin
                       Factor1 := 100;
                       Factor2 := 60;
                  End;
               2: Begin
                       Factor1 := 70;
                       Factor2 := 45;
                  End;
               3: Begin
                       Factor1 := 50;
                       Factor2 := 30;
                  End;
               End;
               Matriz [2,Punto] := a * Factor1 + 5;
               Matriz [4,Punto] := b * Factor2 - 5;
               Matriz [6,Punto] := 0.0005;
          End;
     End;
     For Punto := 1 to Numero_de_condiciones do
     Begin
          Matriz [9,Punto] := 1;
          Matriz [5,Punto] := 0;
     End;
     Matriz [7] := Matriz [2];
     Matriz [8] := Matriz [4];
End;

{-------------------------------------------------------------------------------}

Procedure  Ajusta_dt;
Begin
     If (Abs(Matriz [1,Punto] - Matriz [2,Punto]) < 1) and
        (Abs(Matriz [3,Punto] - Matriz [4,Punto]) < 1) then
        Matriz [6,Punto] := 1.5 * Matriz [6,Punto];
     If (Abs(Matriz [2,Punto] - Matriz [1,Punto]) > 2) or
        (Abs(Matriz [3,Punto] - Matriz [4,Punto]) > 2) then
        Matriz [6,Punto] := 0.2 * Matriz [6,Punto];
     If (Abs(Matriz [2,Punto] - Matriz [1,Punto]) > 5) or
        (Abs(Matriz [3,Punto] - Matriz [4,Punto]) > 5) then
        Matriz [4,Punto] := 100;
End;
```

```
{-------------------------------------------------------------------------------}

Procedure Ejes;
var i: integer;
Begin
     Hires;
     Hirescolor(15),
     begin
          for i := 0 to 159 do plot (4*i,99,1);
          for i := 0 to 99  do plot (319,2*i,1);
     end;
End;

{-------------------------------------------------------------------------------}

Procedure  Va_de_regreso;
Begin
     Matriz [9,Punto] := Matriz [9,Punto] - 2;
     Matriz [5,Punto] := 0;
     Matriz [2,Punto] := Matriz [7,Punto];
     Matriz [4,Punto] := Matriz [8,Punto];
     Matriz [6,Punto] := 0.0005;
End;

{-------------------------------------------------------------------------------}

procedure flechas;
var i: integer;
    x,y: real;
begin
     for i := 1 to numero_de_condiciones do
     begin
          if (sqr(matriz[4,i] - matriz [3,i]) > 0) and (Abs(Matriz [6,i]) <0.1)
          then
          begin
          x := ((matriz[2,i] - matriz [1,i])*cos(pi/4) -
               (matriz [4,i] - matriz [3,i])*sin(pi/4)) * 4 /
               sqrt(sqr(matriz[2,i] - matriz [1,i]) +
                    sqr(matriz[4,i] - matriz [3,i]));
          y := ((matriz[2,i] - matriz [1,i])*sin(pi/4) +
               (matriz [4,i] - matriz [3,i])*cos(pi/4)) * 4 /
               sqrt(sqr(matriz[2,i] - matriz [1,i]) +
                    sqr(matriz[4,i] - matriz [3,i]));
          if matriz [9,i] = 1 then begin
                                        x := -x;
                                        y := -y;
                                   end;
          draw (2*round(x+matriz[2,i]) + 319, 99 - round(y+matriz[4,i]),
                2*round(matriz[2,i]) + 319, 99 - round(matriz[4,i]),1);
          x := ((matriz[2,i] - matriz [1,i])*cos(-pi/4) -
               (matriz [4,i] - matriz [3,i])*sin(-pi/4)) * 4 /
               sqrt(sqr(matriz[2,i] - matriz [1,i]) +
                    sqr(matriz[4,i] - matriz [3,i]));
          y := ((matriz[2,i] - matriz [1,i])*sin(-pi/4) +
               (matriz [4,i] - matriz [3,i])*cos(-pi/4)) * 4 /
               sqrt(sqr(matriz[2,i] - matriz [1,i]) +
                    sqr(matriz[4,i] - matriz [3,i]));
          if matriz [9,i] = 1 then begin
                                        x := -x;
                                        y := -y;
                                   end;
          draw (2*round(x+matriz[2,i]) + 319, 99 - round(y+matriz[4,i]),
                2*round(matriz[2,i]) + 319, 99 - round(matriz[4,i]),1);
          end;
```

```
     end;
end;

{-------------------------------------------------------------------------}

Procedure  Calcula_y_Grafica;
Begin
     contador_de_flecha := contador_de_flecha + 1;
     if (frac (contador_de_flecha/60) = 0) and (flechasino ='SI') then flechas;
     Matriz [1] := Matriz [2];
     Matriz [3] := Matriz [4];
     For Punto  := 1 to Numero_de_condiciones do
     Begin
          If (Matriz [9,Punto] > -2) and
             (Abs(Matriz [2,Punto]) + Abs(Matriz [4,Punto]) > 1)
             and (abs(matriz[6,punto]) < 10)
             then
          Begin
               Evalua;
               Ajusta_dt;
               If (Round(Abs(Matriz [2,Punto])) <= 319) and
                  (Round(Abs(Matriz [4,Punto])) <=  99) and
                  (Abs(Matriz [6,Punto]) < 10) and
                  (Abs(Matriz [5,Punto]) < cota)
               then Dibuja
               Else Va_de_regreso;
           End
           Else Va_de_regreso;
     End;
End;

{-------------------------------------------------------------------------}

procedure  proyecta_secuencia;
begin
     clrscr;
     textcolor(2);
     writeln ('PROYECCION DE UNA SECUENCIA DE IMAGENES.');writeln;
     textcolor(15);
     write ('Nombre generico de los archivos de pantallas : ');
     readln (nombre_de_archivos);
     write ('Desde 1 hasta ');
     readln (num_archivos);
     hires;
     for num_imag := 1 to num_archivos do
     begin
          str (num_imag,numero);
          assign (pantalla,concat(nombre_de_archivos,'.',numero));
          reset (pantalla);
          blockread (pantalla,screenbuffer,126,recsread);
          close (pantalla);
          putpic (screenbuffer,0,199);
     end;
     repeat until keypressed;
end;

{-------------------------------------------------------------------------}

procedure  savescreen;
begin
     write ('Nombre del archivo (NOMBRE.nnn , donde nnn = 1 a 999) : ');
     readln (filename);
     assign (pantalla,filename);
     rewrite (pantalla);
```

```
     blockwrite (pantalla,screenbuffer,126,recsread);
     close (pantalla);
end;

{-------------------------------------------------------------------------}

procedure solucion_particular;

label no;
begin
     clrscr;
     no:
     write ('Condicion inicial x = ');
     readln (matriz [2,1]);
     write ('Condicion inicial y = ');
     readln (matriz [4,1]);
     writeln;
     matriz [6,1] := 0.0005;
     s := 40 * Amplificacion;
     Matriz [9,1] := 1;
     Matriz [5,1] := 0;
     matriz [2,1] := (matriz [2,1] - origenx) * s;
     matriz [4,1] := (matriz [4,1] - origeny) * s;
     if (abs(matriz[2,1]) > 3e4) or (abs(matriz[4,1]) > 3e4)
                                 or (f1(Matriz [2,Punto]/s + origenx,
                                     Matriz [4,Punto]/s + origeny)
                                     * Matriz [6,Punto] > 3e4)
                                 or (f2(Matriz [2,Punto]/s + origenx,
                                     Matriz [4,Punto]/s + origeny)
                                     * Matriz [6,Punto] > 3e4)
     then
     begin
          textcolor(12);
          writeln ('Valores fuera de rango.');
          textcolor(15);
          writeln;
          goto no;
     end;
     matriz [7,1] := matriz [2,1];
     matriz [8,1] := matriz [4,1];
     ejes;
     if sobrepuestas = 'SI' then putpic (screenbuffer,0,199);
     numero_de_condiciones := 1;
     contador_de_flecha := 0;
     repeat calcula_y_grafica until keypressed;
end;

{-------------------------------------------------------------------------}

procedure retrato;
begin
     clrscr;
     densidad := 0;
     Repeat Write ('Densidad de flujo (1,2 o 3) ');
            Readln (Densidad);
     until  Densidad in [1,2,3];
     s := 40 * Amplificacion;
     condiciones_iniciales;
     writeln;
     write ('Dibuja ejes (S/N) ? ');
     read (kbd,tecla);
     if (tecla = 's') or (tecla = 'S') then ejes
     else begin
               hires;
```

```
                hirescolor(15);
            end;
        contador_de_flecha := 0;
        repeat calcula_y_grafica until keypressed;
end;

{-------------------------------------------------------------------------}

procedure cambia_amplificacion;
begin
        Repeat clrscr;
               writeln ('Amplificacion = ',amplificacion:4:1);
               writeln;
               Write ('Nueva amplificacion (min 0.1) = ');
               Readln (Amplificacion);
        until  (Amplificacion >= 0.1);
        writeln;
        writeln ('Tiempo maximo de recorrido = ',cota:1:0);
        writeln;
        write ('Nuevo valor = ');
        readln (cota);
end;

{-------------------------------------------------------------------------}

procedure mueve_el_origen;
begin
        clrscr;
        writeln;
        writeln ('El origen esta en (',origenx:5:3,' , ',origeny:5:3,')');
        writeln;
        write ('Nueva coordenada x del origen = ');
        readln (origenx);
        write ('Nueva coordenada y del origen = ');
        readln (origeny);
        sobrepuestas := 'NO';
end;

{-------------------------------------------------------------------------}

procedure menu;
begin
        writeln;
        textcolor(2);
        write ('F1:');
        textcolor(15);
        writeln ('  NUEVA ECUACION');
        textcolor(2);
        write ('F2:');
        textcolor(15);
        writeln ('  GRAFICA SOLUCION PARTICULAR');
        textcolor(2);
        write ('F3:');
        textcolor(15);
        writeln ('  RETRATO FASE COMPLETO');
        textcolor(2);
        write ('F4:');
        textcolor(15);
        writeln ('  SALVA LA PANTALLA EN DISCO');
        textcolor(2);
        write ('F5:');
        textcolor(15);
        writeln ('  CAMBIA LOS PARAMETROS O CONSTANTES');
        textcolor(2);
```

```
     write ('F6:');
     textcolor(15);
     writeln ('  ORIGEN (',origenx:3:1,' , ',origeny:3:1,')');
     textcolor(2);
     write ('F7:');
     textcolor(15);
     writeln ('  CAMBIA TIEMPO MAX (',cota:1:0,
              ') Y/O AMPLIFICACION (',amplificacion:4:1,')');
     textcolor(2);
     write ('F8:');
     textcolor(15);
     writeln ('  SOLUCIONES PARTICULARES SOBREPUESTAS (',sobrepuestas,')');
     textcolor(2);
     write ('F9:');
     textcolor(15);
     writeln ('  DIBUJA FLECHAS EN EL FLUJO (',flechasino,')');
     textcolor(2);
     write ('F10:');
     textcolor(15);
     writeln (' PROYECTA IMAGENES DE DISCO');
     textcolor(2);
     write ('ESC:');
     textcolor(15);
     writeln (' TERMINA');
end;

{-------------------------------------------------------------------------------}

procedure mensaje;
begin
     textmode(c80);
     textcolor(15);
     clrscr;
     writeln;
     writeln ('GRAFICADOR DE ECUACIONES DIFERENCIALES EN EL PLANO');
     writeln ('(c) Salvador Malo G.,  1987-88, IMUNAM');
     writeln;
     write ('Oprima ESC para salir o RETURN para continuar');
     read (kbd,tecla);
     if tecla = #27 then halt;
end;

{-------------------------------------------------------------------------------}

function funckey: boolean;
begin
     if (tecla = #27) and keypressed then
     begin
          read (kbd,tecla);
          funckey := true;
     end;
end;

{-------------------------------------------------------------------------------}

BEGIN
     mensaje;
     flechasino := 'NO';
     sobrepuestas := 'NO';
     ya := 0;
     origenx := 0;
     origeny := 0;
     amplificacion := 1;
     cota := 5;
```

```
    otra_vez:
    if ya = 1 then getpic (screenbuffer,0,0,639,199);
    textmode(c80);
    textcolor(15);
    clrscr;
    menu;
    ya := 0;
    read (kbd,tecla);
    if funckey then
    case tecla of
         #61: begin
                   retrato;
                   ya := 1;
              end;
         #68: proyecta_secuencia;
         #66: begin
                   if sobrepuestas = 'SI' then sobrepuestas := 'NO'
                   else sobrepuestas := 'SI';
                   goto otra_vez;
              end;
         #65: cambia_amplificacion;
         #60: begin
                   solucion_particular;
                   ya := 1;
              end;
          #67: begin
                    if flechasino = 'SI' then flechasino := 'NO'
                    else flechasino := 'SI';
                    goto otra_vez;
               end;
          #63: parametros;
          #62: savescreen;
          #59: begin
                    ecuacion_nueva;
                    goto otra_vez;
               end;
          #64: mueve_el_origen;
          else if tecla <> #27 then goto otra_vez else halt;
     end;
     goto otra_vez;
END.
```