Visualización de Sistemas no Lineales y Caóticos

Claudio Delrieux y Andrés Repetto
ICIC — Instituto de Ciencias e Ingeniería de Computación
Departamento de Ingeniería Eléctrica
Universidad Nacional del Sur
Alem 1253 — (8000) Bahía Blanca — ARGENTINA
e-mail:claudio@acm.org, arepetto@criba.edu.ar

Palabras Clave: COMPUTACIÓN GRÁFICA, VISUALIZACIÓN, SISTEMAS NO LINEALES Y CAÓTICOS.

Abstract

Los sistemas de visualización están actualmente revolucionando la metodología de investigación científica. El propósito de estos sistemas es producir una representación gráfica de un conjunto de datos (simulados u obtenidos por medición), de manera de hacer comprensible o evidente alguna característica específica de los mismos. Una de las áreas particularmente importantes donde es posible aplicar las técnicas de visualización es en el estudio de la dinámica de sistemas no lineales y caóticos. En este trabajo se reseñan algunas técnicas utilizadas con dicho propósito, y se desarrollan nuevos algoritmos, para los cuales se comparan los resultados y se discuten nuevas posibilidades.

1 Introducción

La *Visualización Científica* es una de las tecnologías derivadas de las Ciencias de la Computación que actualmente está revolucionando con mayor fuerza las metodologías de investigación científica en todos los campos [4, 5, 16]. Esto es así porque constituye una de las innovaciones de mayor aplicabilidad, al producir una adecuada representación gráfica de datos que, por una u otra razón, no pueden representarse en otros términos más convencionales [21, 22]. Por *visualización* se entiende el empleo de técnicas derivadas de la Computación Gráfica utilizadas para la representación de datos científicos de diverso tipo [1, 14], los cuales pueden provenir de simulaciones, mediciones, etc.

Los resultados de la visualización de estos datos no son meramente una representación cuantitativa de los mismos, es decir, no se busca necesariamente la presentación fiel de valores. Por el contrario, se busca un entendimiento global de determinadas propiedades del fenómeno, modelo o de la simulación que produjo los datos [10, 12]. Es decir, se busca una *comprensión cualitativa* del conjunto de datos representado. Esto muchas veces se consigue por medio del uso de metáforas visuales adecuadas (por ejemplo, representar una temperatura o altitud con un color), sumado al uso de técnicas que permitan la representación de datos multivariados (por ejemplo, volúmenes por medio de transparencias) y/o datos multivaluados o vectoriales

(por ejemplo, flechas para representar direcciones). Por dicha razón es que este tema es tan complejo y aplicable a tantas disciplinas.

Una de las áreas de aplicación más importante de la visualización es el análisis dinámico de sistemas complejos. En efecto, gran parte de los modelos matemáticos que describen los aspectos de la realidad que son objeto de estudio, se caracterizan por no tener soluciones analíticas cerradas. Es decir, ecuaciones del modelo describen la evolución temporal de sus variables, pero la complejidad de las mismas impide una adecuada predicción del comportamiento del sistema a partir de una determinada condición inicial. Es así que surge la simulación computacional como herramienta de observación.

En los sistemas no lineales o caóticos, sin embargo, una mera simulación numérica es insuficiente, dado que el comportamiento cualitativo del sistema puede comprenderse adecuadamente a partir de la estructuración topológica de la evolución temporal de sus trayectorias. Por dicha razón es de mayor utilidad un método gráfico, en el cual se cotejan las trayectorias con las estructuras matemáticamente posibles (puntos críticos, ciclos límite, atractores, etc.). Sin embargo, aún en sistemas relativamente sencillos esto puede no ser suficiente, dado que una representación gráfica "directa" de las trayectorias en el espacio de fases se puede realizar en sistemas de dos o a lo sumo tres variables, no pudiéndose representar la dependencia del comportamiento respecto de cambios de parámetros u otros fenómenos importantes. En este trabajo se explora la utilización de técnicas de visualización para la representación del comportamiento dinámico de sistemas no lineales y caóticos. Se reseñan los algoritmos presentados en la literatura, y se proponen modificaciones que producen sustanciales mejoras a los mismos.

2 Modelos básicos: representación de trayectorias

Uno de los procedimientos más usuales para facilitar la comprensión del comportamiento de un sistema dinámico consiste en representar gráficamente las trayectorias u órbitas. Esto se denomina *análisis gráfico* [8, 18]. Una manera adecuada de realizar un análisis gráfico en el espacio de fases en un sistema unidimensional discreto $\dot{x} = x_{k+1} = f(x_k)$ consiste en graficar la función f y la diagonal $x_{k+1} = x_k$. Partiendo de una condición inicial o semilla x_0, entonces, se comienza en el punto (x_0, x_0) y se recorre una línea vertical hasta $(x_0, f(x_0))$, luego una horizontal hasta $(f(x_0), f(x_0))$, y así sucesivamente (ver Figura 1(a)).

Uno de los problemas de este método gráfico, sin embargo, es que se vuelve ininteligible al cabo de unas pocas iteraciones (ver Figura 1(b)). Más aún, no parece posible representar la convergencia del comportamiento del sistema a un patrón estable para varias trayectorias simultáneamente. Estos dos objetivos, sin embargo, pueden conseguirse utilizando las técnicas de manejo de colores desarrolladas en los sistemas de visualización.

La idea básica consiste en ir disminuyendo la saturación e intensidad de la representación de una trayectoria, de modo que su influencia en el gráfico se acomode a un patrón de comportamiento. Si el sistema es atraído hacia un punto fijo, por ejemplo, entonces la trayectoria en el espacio de fases se representará con un color que irá tendiendo hacia un punto gris. Adoptando diversos colores para distintas condiciones iniciales, es posible representar el análisis gráfico de varias trayectorias para un valor fijo del parámetro r (ver Figura 2(a)).

Si, en cambio, para otro valor del parámetro, el sistema tiende a un ciclo de período 2, entonces el análisis gráfico de varias trayectorias simultáneas mostrará cómo algunas trayecto-

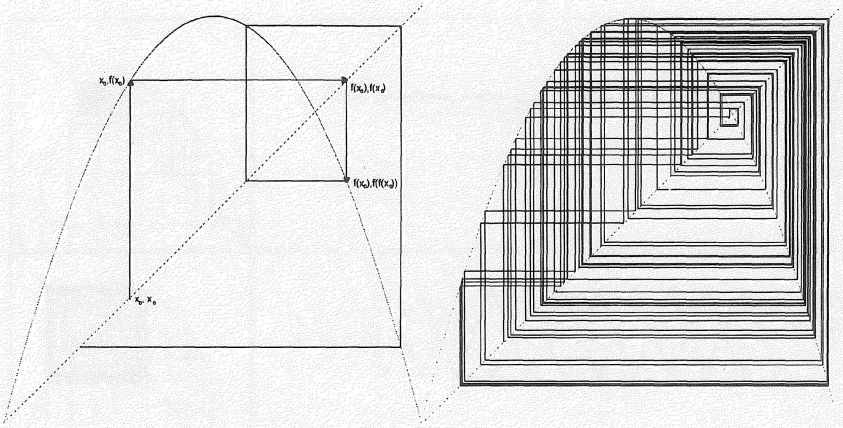

Figura 1: Análisis gráfico de trayectorias en el sistema $\dot{x} = x_{k+1} = rx_k(1 - x_k)$.

rias convergen "centrífugamente" y otras "centrípetamente" hacia dicho comportamiento (ver Figura 2(b)). Por último, si el sistema está en fase caótica, es posible observar de qué manera la condición inicial deja de ser importante y todas las trayectorias se dispersan en el espacio de fase de igual forma.

La implementación de estos resultados requiere, entre otras cosas, un uso adecuado de la paleta cromática [15]. En sistemas gráficos de tipo *true color*, es solamente necesario relacionar la crominancia del gráfico de la trayectoria con la condición inicial, y la saturación e intensidad con la cantidad de iteraciones. En los ejemplos de la Figura 2 utilizamos un mapeo lineal que asocia crominancia 0^o a $x_o = 0$ y 360^o a $x_o = 1$, mientras que la intensidad y saturación parten de 1 y se decrementan en 0.03 en cada iteración. El valor cromático calculado en el espacio CLS es luego convertido a RGB [9]. En sistemas gráficos de paletas limitadas (por ejemplo, 256 colores simultáneos), se utiliza una técnica de cuantización dentro del espacio cromático como la que se describe en [7].

3 Representación de trayectorias en sistemas bidimensionales

Los métodos gráficos también han sido extensamente utilizados para el estudio y análisis de sistemas lineales y no lineales de dos o más dimensiones, especialmente para el estudio de su comportamiento dinámico en el espacio de fases [18]. En este sentido, la herramienta gráfica es de gran utilidad en la caracterización del diagrama de fases de sistemas de gran complejidad, dado que permite localizar y caracterizar los puntos fijos y trayectorias cerradas y sus estabilidades asociadas.

Algunas de las técnicas mostradas en la Sección anterior pueden utilizarse en forma combinada para una representación adecuada de estos sistemas. En la Figura 3, por ejemplo, se observa el diagrama de fases del sistema

$$\begin{aligned} \dot{x} &= -y + \mu x + xy^2 \\ \dot{y} &= x + \mu y - x^2, \end{aligned}$$

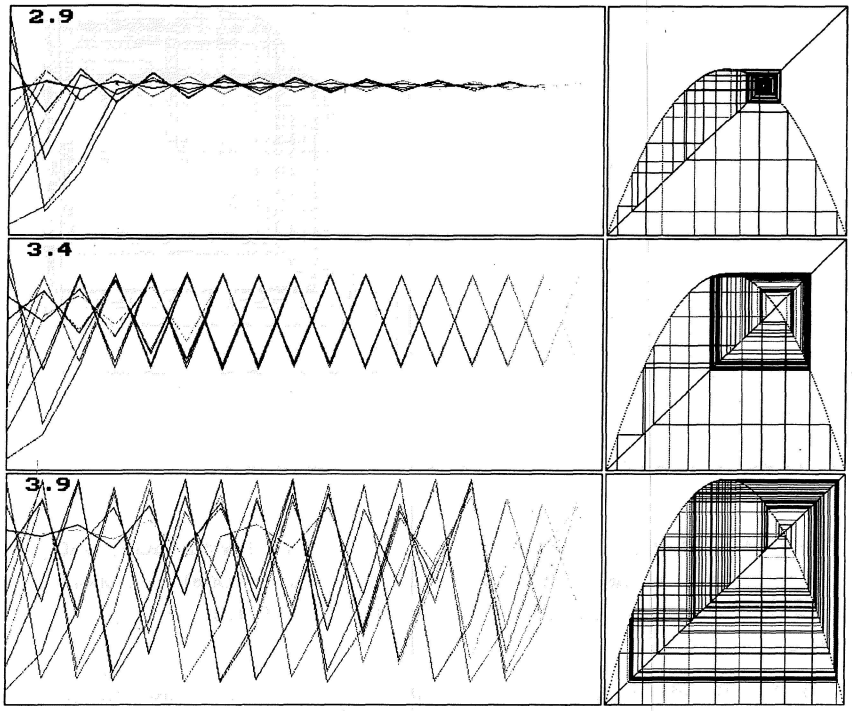

Figura 2: Análisis gráfico mejorado

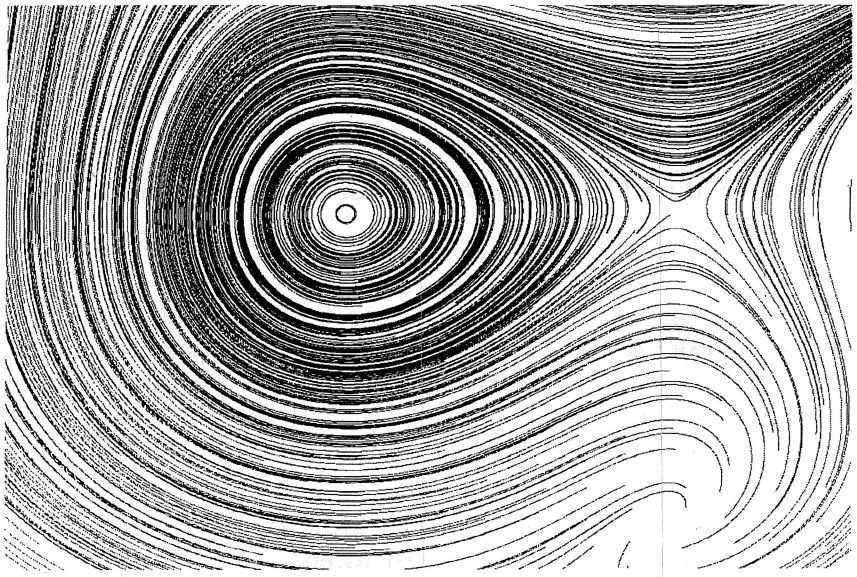

Figura 3: Trayectorias de un sistema bidimensional no lineal.

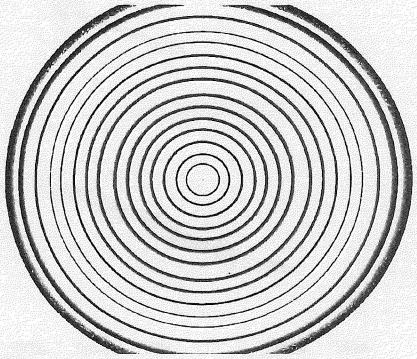

Figura 4: Una observación detallada muestra un ciclo límite inestable.

con $\mu = -0.01$ (ver [18]), valor para el cual se detecta inicialmente un centro en el origen, un punto silla a su derecha y una espiral o foco inestable abajo a la derecha. La graficación de las trayectorias se efectuó utilizando la técnica de disminuir la saturación del color a medida que aumentaba la cantidad de iteraciones, y de perturbar las semillas con una distribución de Poisson.

De esa manera, es posible determinar que el origen en realidad no es un centro sino un foco estable, dado que es posible notar que las trayectorias a su alrededor no son elípticas sino espiraladas, algo que graficándolas con el mismo color hubiese sido muy difícil de detectar. Del mismo modo, es posible ver que no hay trayectorias homoclínicas que incluyan al punto silla. De haber tomado una grilla rectangular de semillas, este hecho no se hubiese observado porque se superpondrían dos o más trayectorias, creando la sensación de una única trayectoria homoclínica. Por último, una observación cuidadosa de las trayectorias alrededor del origen muestra que las más cercanas (de menor radio) convergen hacia el punto fijo, mientras que las más lejanas divergen. Esto sólo puede explicarse con la existencia de un ciclo límite inestable alrededor del origen.

Una ampliación del diagrama de fases revela la existencia de un ciclo límite inestable de pequeño radio ($r \approx 0.02$). El mismo se evidencia por el hecho de que las trayectorias de menor radio convergen hacia el centro. La iteración de una semilla dentro de dicho radio comienza con color saturado (en este caso amarillo) y, a medida que la trayectoria evoluciona, se va volviendo gris (ver Figura 4). Por dicha razón estas trayectorias se observan como círculos gruesos, cuya saturación se pierde hacia el centro. Cuanto más grueso el círculo, más rápida la convergencia. A partir de cierto radio el comportamiento se invierte, siendo los círculos saturados en el interior y grises en el exterior. Esto significa que las trayectorias están divergiendo. Existe, sin embargo, un determinado radio para el cual las trayectorias no divergen ni convergen. Dichas trayectorias, entonces, pertenecen al ciclo límite.

4 Visualización basada en texturas

Existen otras posibilidades para representar mapas vectoriales además de las utilizadas en la Sección anterior. La representación con trayectorias puede generalizarse a trazas por medio de partículas [20, 3], íconos [11, 17], o *hyperstreamlines*, es decir, trayectorias graficadas con un

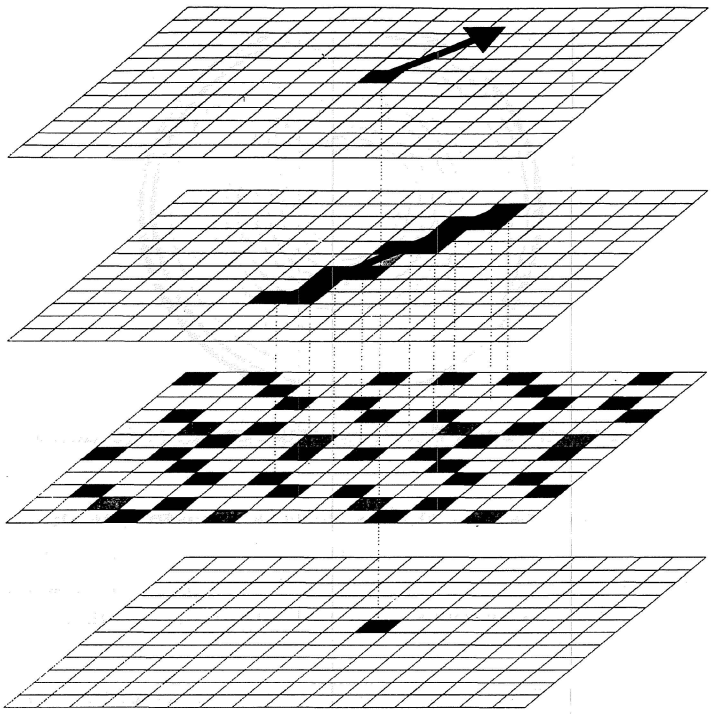

Figura 5: Convolución rectilínea entre una trayectoria y una textura.

espesor finito al cual se le agregan atributos que permiten representar magnitudes vectoriales adicionales [13, 6].

Desafortunadamente, estos métodos tienen un conjunto de desventajas. La calidad del resultado generalmente depende del posicionamiento inicial de las trayectorias (las "semillas"). La perturbación de Poisson sugerida en la Sección anterior suele ser adecuada para muestrear el mapa de una manera uniforme, pero no hay garantías de que queden zonas importantes sin explorar (por ejemplo, trayectorias homoclínicas). De todas maneras, el resultado siempre muestra que las zonas del mapa con bajo jacobiano están muy densamente cubiertas y a la inversa. La utilización de íconos, partículas u otras entidades geométricas agrega el problema de la gran resolución que se requiere, lo cual limita su uso a pequeñas zonas del mapa. Por útlimo, en general estos métodos de visualización requieren un cálculo muy complejo y sin posibilidades de ser reutilizado en distintas partes de su elaboración.

4.1 Convolución rectilínea

Los métodos basados en texturas no tienen estas desventajas, dado que por una parte representan uniformemente el flujo vectorial al margen de los gradientes locales, y, por otra parte, su cómputo es local, sencillo, y en general es posible reutilizarlo por medio de algoritmos incrementales [19, 2]. La idea básica de la visualización de campos vectoriales basada en texturas consiste en efectuar una convolución local entre una textura y un segmento de trayectoria, para cada punto del mapa. Esto garantiza un muestreo uniforme del mapa y la rapidez del cómputo por medio de funciones altamente optimizadas en el procesamiento de señales.

La manera más sencilla de implementar una convolución rectilínea es utilizar un segmento de recta que pase tangente a la trayectoria en el punto de muestra. De esa manera, se utiliza una ventana de convolución con una función y ancho predeterminado (*kernel*), se recorre la textura a lo largo de dicho segmento efectuando la convolución, y se pinta el punto de muestra con el color resultante. En primera instancia, un recorrido rectilíneo de la textura, desde el punto de inicio *hacia adelante* en la dirección de la derivada, produce el efecto de "derrapar" la textura hacia adelante, lo cual puede ser deseable pero no es necesariamente lo correcto. Para tener en cuenta la dinámica local del sistema en un punto es necesario conocer parte de su historia, además de su futuro, por lo que el segmento de recta debe tener su punto medio en el lugar en el que se evalúa la derivada de la trayectoria (ver Figura 5).

Esta implementación puede ser muy eficiente, pero es esencialmente incorrecta, dado que en la medida en que el radio local de curvatura de la trayectoria es comparable o menor a la longitud de la ventana de convolución, la aproximación de la trayectoria con un segmento de recta es más errónea. Por dicha razón es que se utilizan métodos más elaborados, como los que veremos a continuación.

4.2 Convolución de integral lineal

Es inmediato que esta aproximación puede ser fácilmente mejorada si en vez de efectuar la convolución a lo largo de un segmento de recta, ésta se efectúa a lo largo de la trayectoria propiamente dicha. En esto consiste el algoritmo de convolución de integral lineal (LIC)[1] propuesto por Cabral y Leedom [2]. De esa manera se corrigen las características insatisfactorias de la convolución rectilínea. En el trabajo original de Cabral y Leedom se propone discretizar la trayectoria del siguiente modo:

1. A partir de un punto de la textura (texel) ya visitado sobre la trayectoria, se evalúa la dirección de la derivada de la misma.
2. Se recorre en dicha dirección desde el centro del mismo hasta que se abandona el texel y se ingresa en un texel vecino.
3. Dicho texel es el próximo a visitar y computar en la convolución.
4. Estos pasos se realizan hacia adelante y hacia atrás en la dinámica del sistema, para tener en cuenta el pasado y el futuro del punto sobre la trayectoria.

El costo adicional, sin embargo, es el de tener que recorrer la trayectoria a partir del punto de inicio correspondiente. Cabral y Leedom reconocen que este algoritmo es un orden de magnitud más lento que la convolución rectilínea. Además, la implementación es bastante engorrosa por la cantidad de casos particulares que pueden ocurrir (derivadas nulas, errores numéricos, singularidades), las cuales se subsanan agregando términos correctivos artificiales. Por último, si bien el recorrido de la trayectoria es menos aproximado que con la convolución rectilínea (y produce resultados de mejor calidad), la aproximación que se realiza puede ser grosera en muchos casos.

[1] Se denomina de esa manera porque el recorrido de la trayectoria es una *integral lineal*, la cual es, a su vez, convolucionada con la textura.

El ejemplo más evidente es que cuando la trayectoria ingresa a un texel, se evalúa su derivada en el centro del mismo, es decir, en un punto cercano pero no exacto del mapa vectorial. Además, con radios de curvatura pequeños, es posible que se evalúe incorrectamente el próximo texel si la derivada local en el centro del mismo apunta en una dirección pero la trayectoria abandona el texel en otro sentido. Por todas estas razones, además de la complejidad de los algoritmos y el costo en tiempo, es que proponemos nuevas alternativas para evaluar la LIC en lo que sigue de esta Sección.

4.3 LIC por diferencial finito

La manera usual de recorrer una trayectoria en un sistema dinámico es por diferenciales finitos, utilizando algún método para reducir el error sistemático acumulado, como por ejemplo por Runge-Kutta. De esa manera, la trayectoria se recorre agregando al punto actual un factor finito ϵ multiplicando a la derivada del mapa. Cada punto así obtenido se considera parte de una poligonal, la cual, con valores muy pequeños de ϵ, aproxima arbitrariamente la trayectoria buscada (con costos computacionales crecientes). De esa manera fueron obtenidas las trayectorias en la Sección anterior. ¿Podremos utilizar una técnica similar para recorrer los texels de una trayectoria?

El problema de utilizar diferenciales finitos para discretizar curvas no lineales es bien conocido. Valores pequeños de ϵ determinan un recorrido muy lento, lo cual crea computación redundante de valores que en definitiva permanecen dentro del mismo texel. Por el contrario, valores grandes determinan que en vez de pasar de un texel al vecino correspondiente, nuestra evaluación "salta" a cierta distancia, lo cual produce un efecto de *aliasing* que deteriora la calidad de la visualización. No es posible encontrar un valor justo para el diferencial finito, dado que un mismo sistema puede recorrer lentamente el mapa en algunos lugares (cerca de un punto fijo, por ejemplo) y muy rápidamente en otros.

Pero es posible pensar en una asignación *adaptativa* de ϵ. A partir de un valor *a priori* (conservadoramente bajo), se computa un próximo punto en el mapa por Runge-Kutta. Si dicho punto yace sobre un texel vecino al actual, el valor del diferencial fue correcto y seguimos recorriendo la integral lineal. Si el punto yace sobre el texel actual, el diferencial fue pequeño y se lo incrementa para computar un nuevo punto. Si el punto yace sobre un texel no conexo con el actual, el diferencial fue grande y se lo decrementa para computar nuevamente un punto.

Este algoritmo es conceptualmente más sencillo, y de implementación más directa. No introduce factores de corrección artificiales, es poco sensible a los errores numéricos, y tiene un único caso particular cuando la trayectoria es muy cercana a un punto fijo. En este caso, el gradiente del mapa en la trayectoria es muy pequeño, y por lo tanto el algoritmo adaptativo intentará encontrar un ϵ cada vez más grande, amplificando el error numérico. Para corregir este caso, se establece un límite al valor del diferencial (normalmente por debajo de la unidad), y si una trayectoria con dicho diferencial, al cabo de una determinada cantidad de iteraciones (por ejemplo 100) no abandona el texel, se considera que la trayectoria "muere" en dicho texel. Podemos comparar el resultado de este algoritmo, con una textura de ruido blanco y pequeños cuadrados de colores, en la Figura 6.

Figura 6: Convolución por diferencial finito del sistema dinámico de la Figura 3.

4.4 El LIC Workbench

Para poder evaluar la calidad de los resultados obtenidos, fue necesario unir en un único programa la posibilidad de combinar diversos sistemas dinámicos, con sus parámetros propios, junto con diferentes texturas, y la posibilidad de computar las LIC para diferentes valores de kernel, transformaciones, etc. De esa manera se implementó el `LIC Workbench`[2]. El mismo permite seleccionar en forma interactiva todas las combinaciones mencionadas, lo cual posibilita su uso con fines exploratorios.

Podemos ver en la Figura 7 la interfase con el usuario, donde se observa a la izquierda la ventana de diálogo para elegir la textura y el lienzo donde se muestra la misma. Más abajo podemos ver la ventana donde se eligen los parámetros de evaluación de la convolución, en particular el valor inicial del diferencial finito y el tamaño del kernel. La ventana de diálogo inferior izquierda permite elegir el sistema dinámico en un menú donde figuran algunos ejemplos representativos, junto con los parámetros de transformación (centrado y escala) para poder ubicar correctamente los puntos críticos y trayectorias significativas. Una vez elegido el sistema dinámico, se muestra la ecuación diferencial del mismo, y se inicializan los valores en las ventanas de diálogo mencionadas a los valores por omisión que parecen más adecuados.

Una vez elegidos todos los valores, se activa el botón de comienzo, y el programa comienza a mostrar en el lienzo de la derecha el resultado de la convolución de la textura elegida con el sistema dinámico seleccionado, a partir de los parámetros dados. Dos botones más permiten al usuario guardar el contenido de los lienzos de textura y de sistema dinámico en formato de archivo bmp. Podemos observar en la Figura 8 el resultado de convolucionar la misma textura utilizada para generar la Figura 6, con algunos sistemas dinámicos.

[2]Este sistema, implementado en Delphi para Windows, está disponible para quien lo solicite a los autores, o en `http://www.ingelec.uns.edu.ar/lic`.

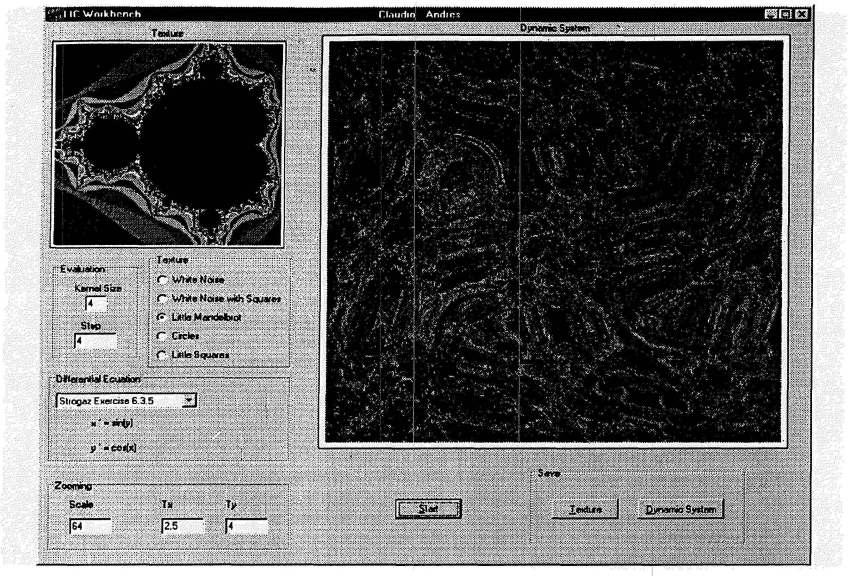

Figura 7: Interfase con el usuario del `LIC Workbench`.

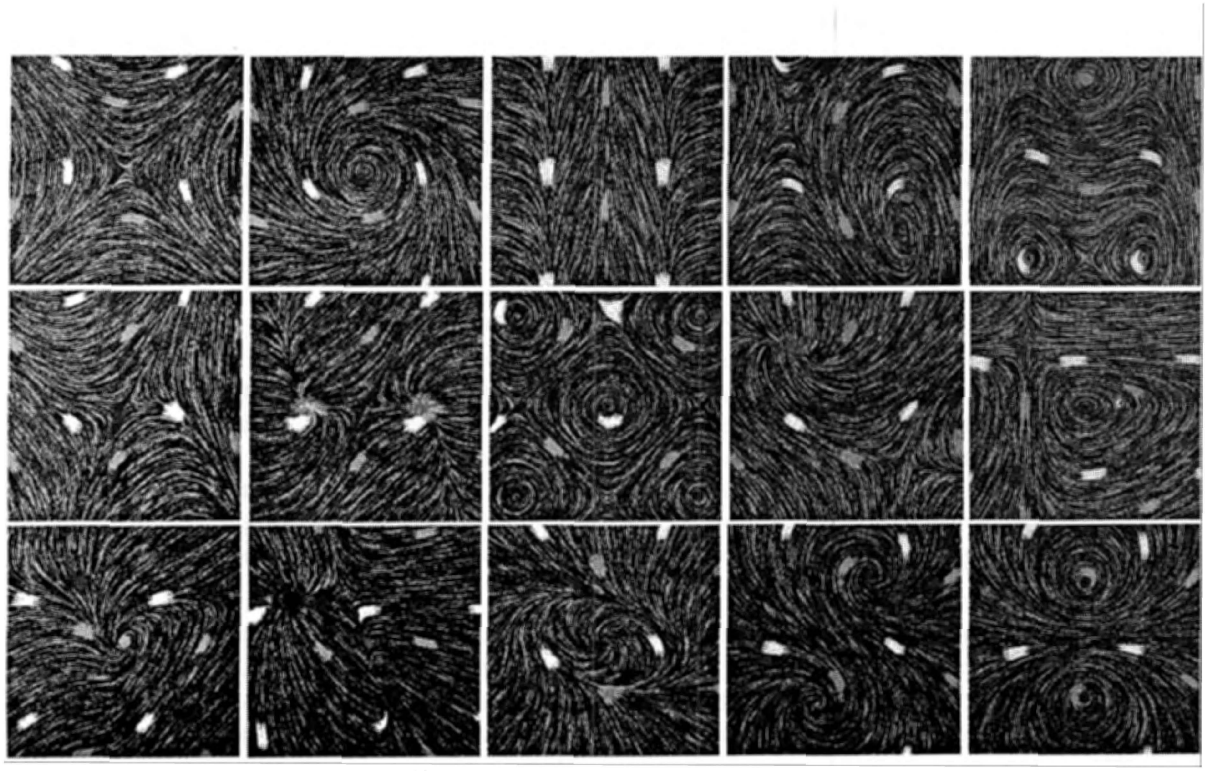

Figura 8: LIC de un conjunto de sistemas dinámicos.

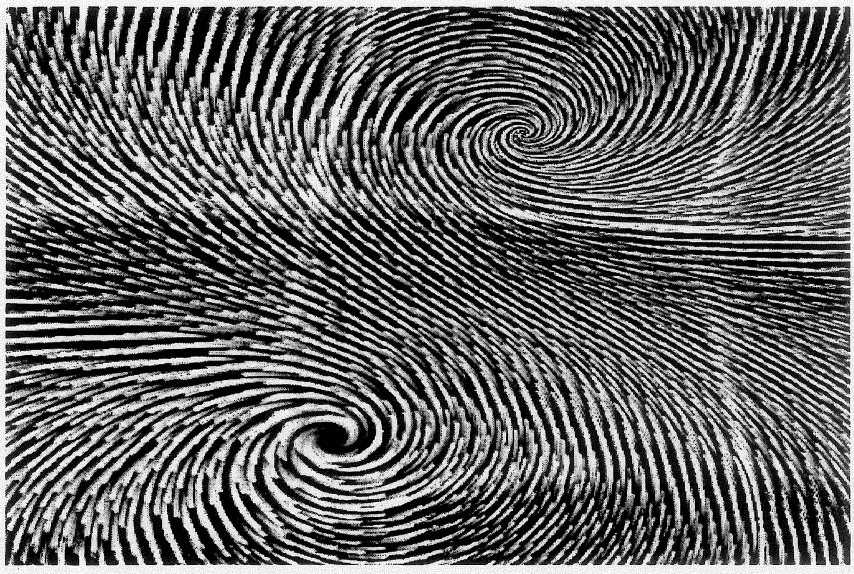

Figura 9: LIC animado.

5 Mejoras, conclusiones y trabajo futuro

En este trabajo se presentaron algunas técnicas derivadas de los sistemas de visualización científica aplicadas al estudio de la dinámica de sistemas no lineales y caóticos. Estas técnicas se basan fundamentalmente en el uso del color, la elección adecuada de las semillas para calcular las trayectorias, y el uso de integrales lineales para representar la topología local del sistema dinámico. Podemos concluir que dichas técnicas constituyen una herramienta adecuada y de gran potencial para el estudio de la dinámica de sistemas no lineales y caóticos.

Una de las ideas más directas que permite mejorar la visualización de la dinámica de los sistemas no lineales, consiste en poder realizar animaciones. Las mismas pueden obtenerse de una manera muy sencilla si se computa la LIC con una ventana que pondere a cada texel en forma asimétrica. Si utilizamos una función ponderadora en forma de rampa creciente, se orientan las trayectorias como si fueran "cometas" cuyas cabezas apuntan hacia la dirección de avance.

El uso de funciones rampa desplazadas, con un desplazamiento diferente en cada cuadro de una animación, produce, en efecto, un movimiento dentro de cada trayectoria que permite visualizar correctamente el sentido y la velocidad de las trayectorias, y por lo tanto ayuda a clasificar los puntos críticos cuando existen ambigüedades en la representación estática. En nuestra implementación, la incorporación de estas ventanas asimétricas es inmediata, pues en cada iteración se computa el valor correspondiente de la función ponderadora a través de la misma variable de iteración sumada a un factor de desplazamiento (o fase) que determina el punto de comienzo de la función rampa. Las posibilidades de este trabajo no permiten mostrar las animaciones obtenidas[3], pero podemos observar un cuadro de la animación de un sistema dinámico en la Figura 9.

[3]También disponibles en la misma URL.

Agradecimientos: Deseamos agradecer a Eduard Gröller por algunas discusiones referidas al contenido de este trabajo. Este trabajo fue parcialmente financiado con un subsidio de la Secretaría de Ciencia y Técnica de la Universidad Nacional del Sur y de la Agencia Nacional de Promoción Científica y Tecnológica.

Referencias

[1] R. S. Avila, l. M. Sobierajski, and A. E. Kaufman. Towards a Comprehensive Volume Visualization System. In *Visualization '92 Proceedings*, pages 13–20, 1992. IEEE Computer Society.

[2] B. Cabral and L. Leedom. Imaging Vector Fields Using Line Integral Convolution. *ACM Computer Graphics (SIGGRAPH Proceedings)*, 25(3):263–270, 1993.

[3] Roger Crawfis, Nelson Max, and Barry Becker. Vector Field Visualization. *IEEE Computer Graphics and Applications*, 14(5):50–56, 1994.

[4] S. Cunningham, J. R. Brown, and M. McGrath. Visualization in Science and Engineering Education. In G. M. Nielson and B. D. Shriver, editors, *Visualization in Scientific Computing*, pages 48–58. IEEE Computer Society Press, 1990.

[5] T. A. Defanti, M. D. Brown, and B. H. McCormick. Visualization: Expanding Scientific and Engineering Research Opportunities. In G. M. Nielson and B. D. Shriver, editors, *Visualization in Scientific Computing*, pages 32–47. IEEE Computer Society Press, 1990.

[6] T. Delmarcelle and L. Hesselink. Visualizing Second-Order Tensor Fields with Hypersrtreamlines. *IEEE Computer Graphics and Applications*, 13(4):25–33, 1993.

[7] C. Delrieux, S. Castro, A. Silvetti, and S. Anchuvidart. Paletas Estáticas para Gráficos de Alta Calidad en PC. *XXII Latin American Conference on Informatics*, 1995.

[8] R. Devaney. *A First Course in Chaotic Dynamical Systems.* Addison Wesley, MA, 1992.

[9] J. Foley, A. Van Dam, S. Feiner, and J. Hughes. *Computer Graphics. Principles and Practice.* Addison-Wesley, Reading, Massachussetts, second edition, 1990.

[10] R. B. Haber and D. A. McNabb. Visualization Idioms: A Conceptual Model for Scientific Visualization Systems. In G. M. Nielson and B. D. Shriver, editors, *Visualization in Scientific Computing*, pages 74–93. IEEE Computer Society Press, 1990.

[11] James Helman and Lambertus Hesselink. Visualizing Vector Field Topology in Fluid Flows. *IEEE Computer Graphics and Applications*, 11(3):36–46, 1991.

[12] P. Keller and M. Keller. *Visual Cues: Practical Data Visualization.* IEEE Computer Society, 1990.

[13] D. Kenwright and G. Mallinson. A 3D Streamline Tracking Algorithm. In *Visualization '92 Conference Proceedings*, pages 62–68, Los Altos, CA, 1992. IEEE.

[14] G. Nielson and B. Shriver. *Visualization in Scientific Computing.* IEEE Computer Society, 1990.

[15] P. K. Robertson and J. O'Callaghan. The Generation of Color Sequences for Univariate and Bivariate Mapping. *IEEE Computer Graphics and Applications*, 6(2), 1986.

[16] L. Rosenblum. Scientific Visualization at Research Laboratories. *IEEE Computer*, 22(8):68–100, 1989.

[17] R. Santaney, D. Silver, N. Sabusky, and J. Cao. Visualizing Features and Tracking Their Evolution. *IEEE Computer*, 27(7):20–27, 1994.

[18] S. Strogatz. *Nonlinear Dynamics and Chaos.* Addison-Wesley, 1994.

[19] J. J. van Wijk. Spot Noise: Texture Synthesis for Data Visualization. *ACM Computer Graphics*, 25(4):309–318, 1991.

[20] J. J. Van Wijk. Rendering Surface Particles. In *Visualization '92 Proceedings*, pages 54–61, 1992, IEEE Computer Society Press.

[21] J. Wright and J. Hsieh. A Voxel-based, Forward-Projection Algorithm for Rendering Surface and Volumetric Data. In *Visualization '92 Proceedings*, pages 340–348, IEEE Computer Society, 1992.

[22] T. S. Yoo, U. Neumann, H. Fuchs, and S. Pizer. Direct Visualization of Volume Data. *IEEE Computer Graphics and Applications*, 12(4):64–71, 1992.