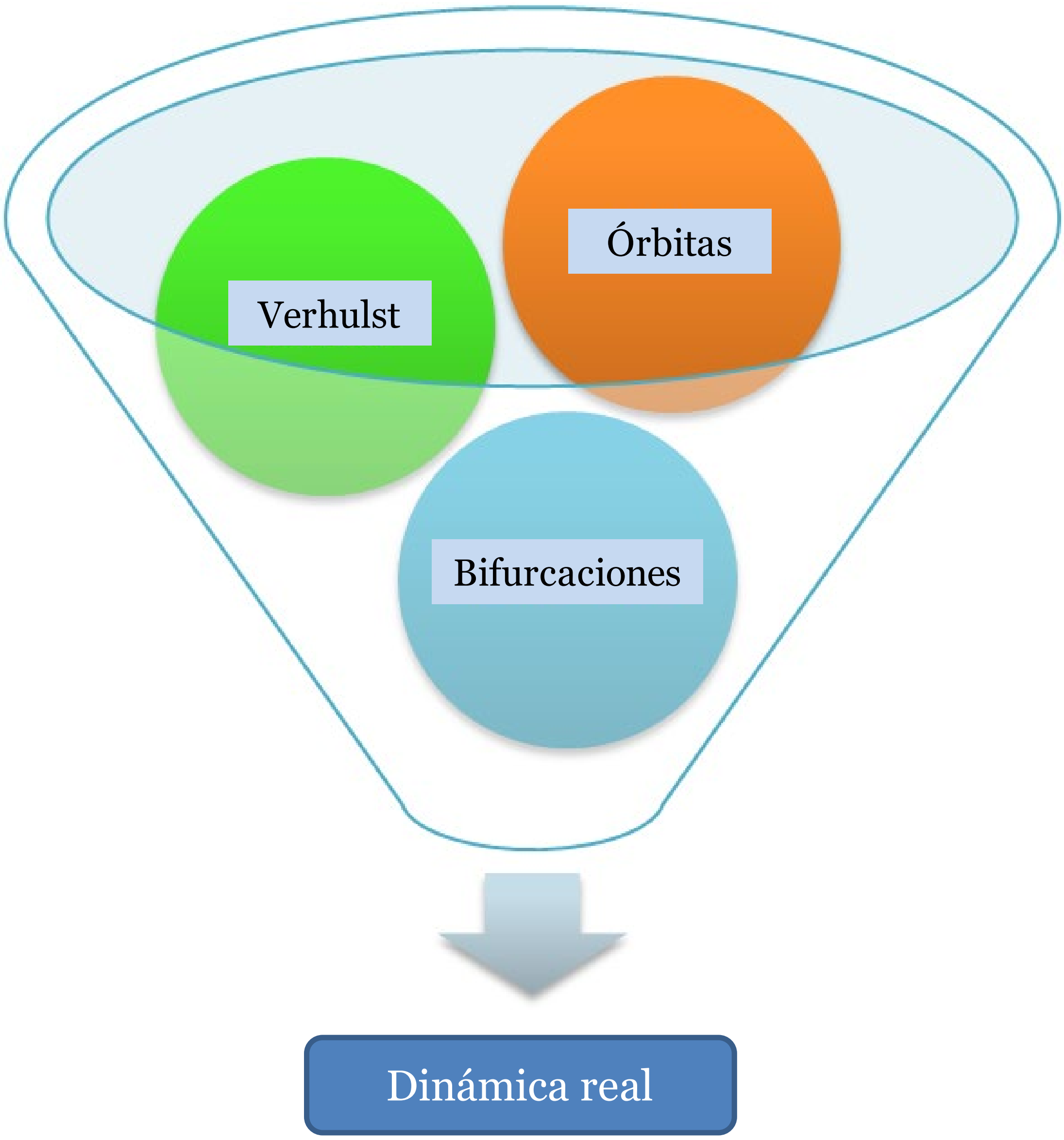
Práctica de SDD I

TEMA 11

Esquema



Ideas clave

11.1. ¿Cómo estudiar este tema?

Para estudiar este tema lee las **Ideas clave** que encontrarás a continuación.

Este tema tiene el formato de un guion de prácticas, por lo que se recomienda realizar los pasos que se van presentando de forma secuencial.

Es imprescindible que se revisen los conceptos de los temas 6 y 7 con anterioridad al seguimiento del presente tema porque así los programas contenidos en este tema favorecerán tu productividad.

A lo largo de este tema se presentan las herramientas estrella asociadas a la dinámica de los sistemas discretos reales: cálculo y representación de órbitas, representación de diagramas de Verhulst y diagramas de bifurcación para familias de sistemas uniparamétricos.

Muchos ejemplos han sido tratados en los temas 6 y 7 de forma que, una vez asimilados los conceptos, desarrollamos el *software* que representa cada una de las figuras.

11.2. Introducción

La verdadera revolución en los sistemas dinámicos discretos se produce cuando las herramientas computacionales son capaces de desarrollar un gran número de operaciones en unas décimas de segundo. Ese es el momento en el que vaticinar qué puede ocurrir en el futuro para un gran conjunto de situaciones iniciales resulta una tarea sencilla.

A lo largo de los dos próximos temas vamos a desarrollar herramientas computacionales relacionadas con los conceptos analíticos que se han visto en los temas 6 a 10. Basados en el programa de acceso libre SciLab, descubriremos una serie de utilidades que nos van a permitir trabajar con los sistemas dinámicos discretos de un modo más sencillo y obtener conclusiones con tan solo un vistazo.

11.3. Dinámica real

En los temas 6 y 7 se trabajó sobre conceptos asociados a la dinámica de sistemas dinámicos discretos reales. A modo de resumen, tenemos que señalar que el sistema dinámico está representado por una expresión analítica cuya órbita queda determinada por dicha expresión y por el punto inicial. Este será el contenido de la primera parte de este apartado.

En la segunda parte profundizaremos sobre determinados sistemas uniparamétricos de forma que estudiaremos el diagrama de bifurcación y cómo generarlo y representarlo con SciLab.

Órbitas

La órbita de un sistema dinámico discreto queda determinada por:

$$\mathcal{O}(x) = \{x, f(x), f^2(x), \dots\}$$

A partir de un punto inicial, el siguiente punto se obtiene a partir del anterior. Puede que conforme avancemos, los puntos se aproximen, se alejen o repitan un patrón. La cuestión es: ¿hasta cuándo calculamos nuevos puntos? Como podríamos seguir hasta el infinito y nunca parar, determinamos una serie de criterios a partir de los cuales consideraremos que se tiene una conclusión al respecto de qué le va a suceder al siguiente punto.

Para poder tomar esas conclusiones definimos $d_{k+1} = |x_{k+1} - x_k|$ como la diferencia entre iterados y t como la tolerancia. En el momento en que la diferencia de iterados sea menor que la tolerancia dejaremos de obtener nuevos puntos considerando que el método ha convergido.

Por otro lado, si los iterados son cada vez mayores es posible que la diferencia entre iterados sea creciente y, por tanto, nunca se alcance un valor por debajo de la tolerancia.

Para ese propósito está el término $maxiter$, que establece el número de iteraciones a partir del cual dejamos de obtener nuevos puntos, considerando que el método no ha convergido. Este criterio puede resultar engañoso, puesto que una órbita periódica caería dentro de este grupo de puntos, de forma que si salimos de las iteraciones por este motivo será necesario realizar un análisis del motivo.

La figura 1 muestra el diagrama de flujo de la generación de las órbitas.

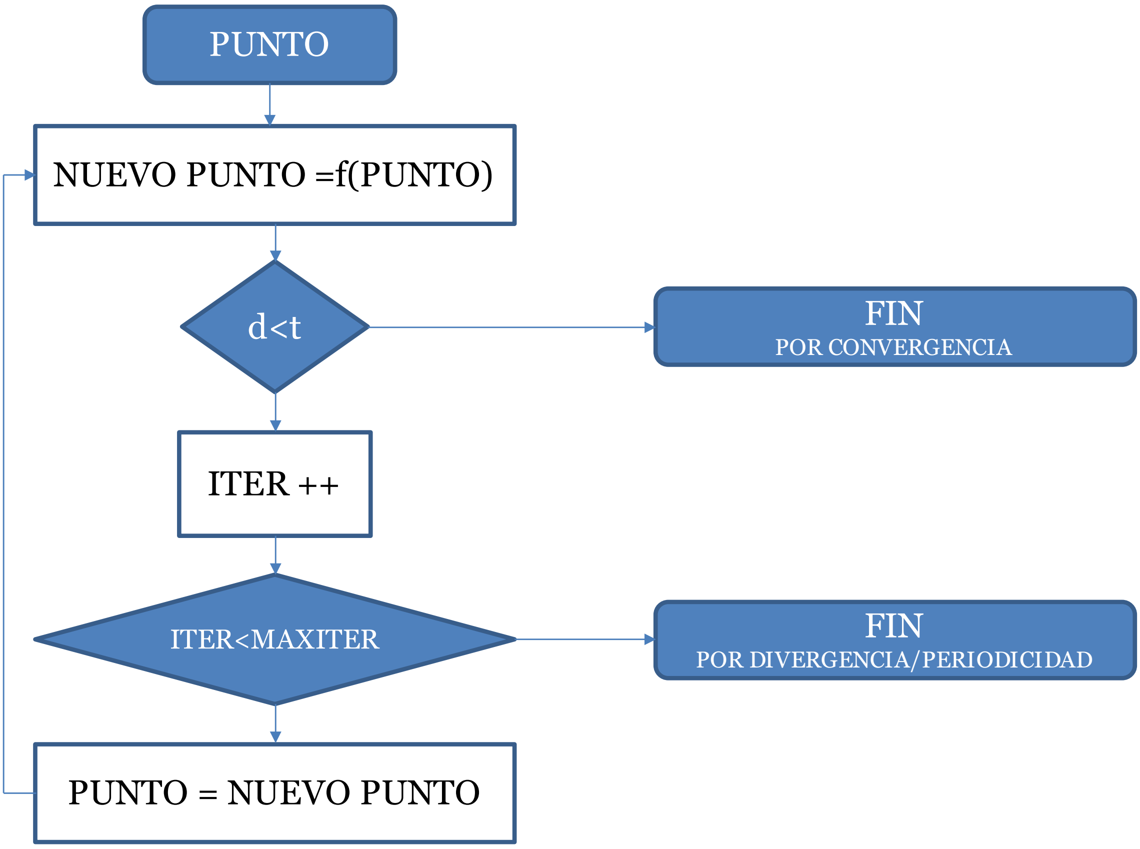


Figura 1. Flujograma de la generación de órbitas

Para ejecutar el código asociado al diagrama de flujo de la figura 1 será necesario definir nuestro sistema dinámico a través de la función f. Para ello, generaremos en SciLab una función con SciNotes cuyo nombre sea f. De cara a mostrar los códigos, tomaremos como ejemplo el sistema dinámico determinado por la función $f(x) = x^2$. La figura 2 muestra el código de SciLab que tenemos que escribir para determinar la función f.

```
function y=f(x)
    y=x.^2;
endfunction
```

Figura 2. Código de la función f

Una vez definida la función que determina el comportamiento del sistema, vamos a generar el código que permite obtener la órbita de un punto inicial. Para ello, las variables de entrada serán la función f, la semilla x_0, el valor de la tolerancia t y el número máximo de iteraciones $maxiter$.

Para realizar un estudio más amplio, las variables de salida serán la órbita de la semilla x, la diferencia entre iterados d y el número de iteraciones requerido $iter$. La figura 3 muestra el código en SciLab para tal propósito.

```
function [iter, d, x]=orbita(f, x0, t, maxiter)
    iter=1;
    d=1;
    x=x0;
    while iter<maxiter&d($)>t
        xk=feval(x($),f);
        d=[d abs(xk-x($))];
        x=[x xk];
        iter=iter+1;
    end
endfunction
```

Figura 3. Código para generar la órbita de una semilla dentro de un sistema dinámico

Probemos el código para una serie de casos. Para ello, iremos a la pantalla en la que se encuentra la consola de SciLab, situaremos el directorio actual donde tengamos ubicados los archivos y clicaremos sobre ellos con el botón secundario para ejecutarlos, como indica la figura 4.

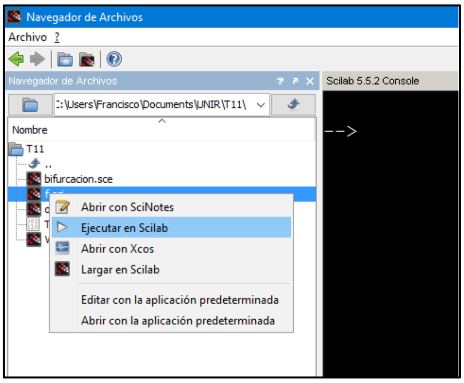


Figura 4. Recuerda ejecutar los programas antes de intentar obtener resultados en la consola

Una vez hayamos procedido de esa manera, vamos a ejecutar el código para obtener diferentes órbitas.

Ejemplo 1 | Determinar la órbita del sistema dado por $f(x) = x^2$ para las semillas $x_0 = \{0.1, 0.25, 0.33, 0.5, 0.87, 1, 2\}$. Se considerará que el método ha convergido cuando la diferencia entre iterados sea de 1e-6. El máximo número de iteraciones permitido es de 50. Representar, en cada caso, el número de iteración en el eje de abscisas y el valor del sistema en el de ordenadas.

Para ejecutar el programa en el primer caso se escribe sobre la línea de comandos:

```
[iter,d,x]=orbita(f,.1,1e-6,50);
```

La representación se realiza, en cada caso, con el comando:

```
plot(1:iter,x);
```

De esta manera, la tabla 1 representa los resultados obtenidos en la que se añade la fila n que determina el número de iteraciones a partir del cual se ha dejado de ejecutar el programa.

x_0	0.1	0.25	0.33	0.5	0.87	1	2
x_n	0	0	0	0	0	1	∞
n	5	6	6	7	9	2	12

Tabla 1. Punto final de la órbita de $f = x^2$ para diferentes semillas

La representación gráfica de todas las semillas da lugar a la figura 4.

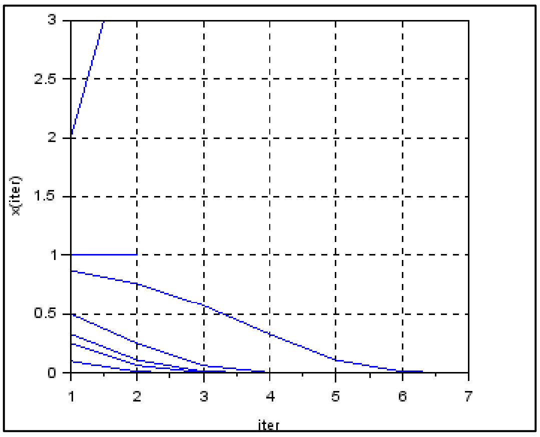


Figura 4. Evolución del sistema $f = x^2$ para diferentes semillas

Diagrama de Verhulst

El diagrama de Verhulst o diagrama cobweb (telaraña) se describió en el punto 6.4.2, correspondiente a la caracterización gráfica de puntos fijos y periódicos. Este diagrama representa la función $y = x$ e $y = f(x)$ de forma que, en la intersección entre ambas funciones están los puntos fijos. La representación de las órbitas da lugar a esa telaraña que determina el comportamiento de cada semilla.

La figura 5 muestra el código SciLab para la representación del diagrama de Verhulst.

```
function [iter, xk]=Verhulst(f, x0, t, maxiter, rangox)
// Representación y1=x, y2=f(x)
    y1=rangox; y2=feval(rangox,f);
    plot(rangox,[y1;y2]);
    h=gca();
    line1=h.children.children(2); line2=h.children.children(1);
    line1.thickness=3; line1.foreground=0;
    line2.thickness=3; line2.foreground=17;
    xgrid;
// Generación órbita
    [iter,d,xk]=orbita(f,x0,t,maxiter);
// Representación órbita
    xplot=repmat(xk,2,1); xplot=xplot(:);
    yplot=xplot(2:$); xplot($)=[];
    plot(xplot,yplot);
    point=h.children(1).children;
    point.foreground=5;
    point.polyline_style=4; point.arrow_size=3;
// Ajuste del gráfico
    h.data_bounds=[min(rangox) max(rangox) ...
        min(rangox) max(rangox)];
    h.isoview="on";
endfunction
```

Figura 5. Código para la representación del diagrama de Verhulst

Podemos observar en las líneas de código los comandos *gca, children, thickness, foreground, polyline_style, arrow_size, data_bounds* o *isoview*. Estos comandos se refieren a las características de la representación. Se invita al lector a modificarlas para darle su propio estilo al gráfico.

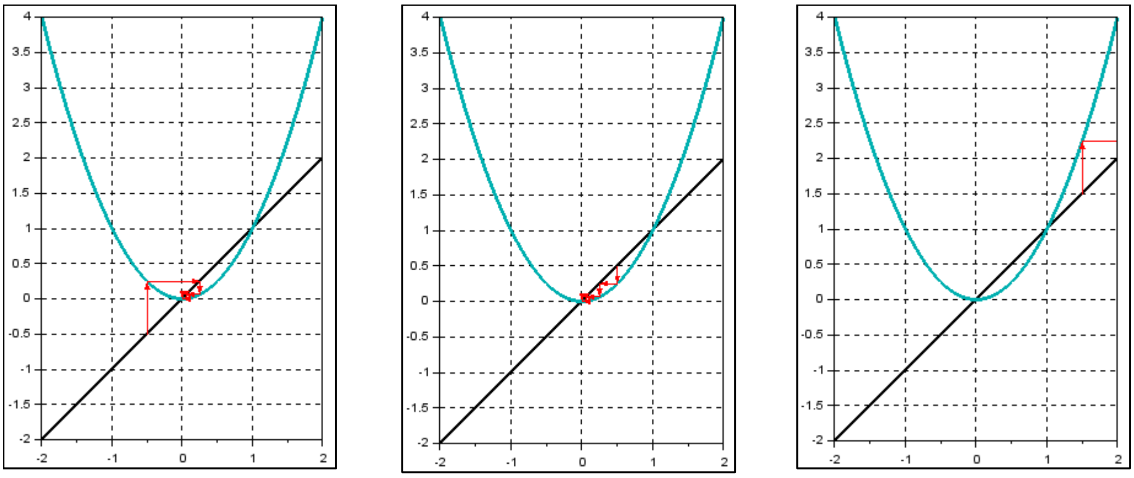
Las líneas 2-9 representan las funciones $y = x$ e $y = f(x)$ dentro del rango de valores de x dados por la variable de entrada *rangox*. Las líneas 10-11 llaman a la función orbita que hemos descrito en el apartado 11.3.1. Las líneas 12-18 generan los puntos que van a generar las flechas que representan la órbita, siendo representadas en las líneas 19-22.

Ejemplo 2 | Representación del diagrama de Verhulst para el sistema determinado por $f(x) = x^2$ con semillas $x_0 = \{-0.5, 0.5, 1.5\}$. Toma como criterios de parada los establecidos en el ejemplo 1.

Realizamos la llamada a la función Verhulst con:

```
[iter,xk]=Verhulst(f,-.5,1e-6,50,linspace(-2,2));
```

Modificando el segundo parámetro de entrada, correspondiente a la semilla, obtenemos los siguientes diagramas de Verhulst, representados en la figura 6.



a) $x_0 = -0.5$ b) $x_0 = 0.5$ c) $x_0 = 1.5$

Figura 6. Diagramas de Verhulst del sistema $f = x^2$ para diferentes semillas

En la figura 6 se pueden observar dos comportamientos diferentes. Por un lado, cuando la semilla es -0.5 ó 0.5, la órbita tiende al punto fijo atractor del sistema $x^* = 0$. Por otro lado, cuando la semilla es 1.5, la órbita diverge hacia el ∞.

Diagrama de bifurcación

Retomemos el ejemplo de la familia logística correspondiente al apartado 7.4.1. Esta familia tiene la expresión:

$$f_\lambda(x) = \lambda x(1-x)$$

Con $\lambda > 0$ y $0 < x < 1$. La tabla 2 recoge las características dinámicas de los puntos críticos del sistema.

Puntos fijos	0	$\frac{\lambda-1}{\lambda}$		
Puntos fijos periodo 2	0	$\frac{\lambda-1}{\lambda}$	$\frac{\lambda+1+\lambda^2\sqrt{(\lambda+1)(\lambda-3)}}{2\lambda}$	$\frac{\lambda+1-\lambda^2\sqrt{(\lambda+1)(\lambda-3)}}{2\lambda}$

Tabla 2. Puntos fijos de la familia logística

Los puntos en los que se producen las bifurcaciones cumplen:

$$\begin{cases} f_\lambda(x^B) = x^B \\ |f'_\lambda(x^B)| = 1 \end{cases}$$

La primera característica la cumplen los puntos fijos $x^* = \left\{0, \frac{\lambda-1}{\lambda}\right\}$. La función derivada tiene la expresión $f'_\lambda(x) = \lambda(1-2x)$, de forma que:

$$\begin{cases} f'_\lambda(0) = \lambda \\ f'_\lambda\left(\frac{\lambda-1}{\lambda}\right) = 2-\lambda \end{cases}$$

El punto fijo $x^* = 0$ será punto de bifurcación cuando $\lambda = 1$, mientras que el punto fijo $x^* = \frac{\lambda-1}{\lambda}$ será punto de bifurcación cuando $\lambda = \{1,3\}$.

El diagrama de bifurcación representa el final de la órbita de todas las posibles semillas para un rango de valores de λ. Para determinar ese rango de valores de λ es necesario realizar el estudio previo que hemos visto previamente.

Como se indica en el enunciado de la familia logística, los valores de la semilla estarán en el rango $0 < x_0 < 1$. Los valores de λ que tomaremos serán $1 < \lambda < 4$.

La figura 7 ilustra el código SciLab para la generación del diagrama de bifurcación.

```
01 function X=bifurcacion(x0, lambda)
02 // Creación malla de puntos
03     [X,L]=meshgrid(x0,lambda);
04 // Estructura iterativa
05     iter=1;
06     while iter<500
07         X=L.*X.*(1-X);
08         iter=iter+1;
09     end
10 // Representación gráfica
11     plot(lambda,X,'b.');
12     h=gca();
13     for j=1:length(x0)
14         h.children.children(j).mark_size=3;
15     end
16 endfunction
```

Figura 7. Código para la generación del diagrama de bifurcación de la familia logística

Cabe destacar que en el caso del diagrama de bifurcación de la función logística es necesario escribir de forma explícita la función de iteración dentro del código (recordar que en los programas anteriores realizábamos la llamada a la función f que habíamos definido en otro lugar). Esto es debido a que el tamaño de la pila de memoria es limitado.

De este modo podemos ver esa expresión explícita en la línea 7. La ejecución de este código, escribiendo en la línea de comandos:

```
X=bifurcacion(linspace(0,1,501),linspace(1,4,501));
```

Vemos que da lugar al diagrama de bifurcación de la figura 8. Cabe destacar que las variables de entrada son los valores que toman la semilla y el parámetro. Es posible que de nuevo la pila de memoria no permita utilizar la precisión que consideremos, de forma que habría que variar el tercer parámetro del comando *linspace* y tener la imagen con una peor resolución o utilizar alguna técnica alternativa para representar la figura por trozos.

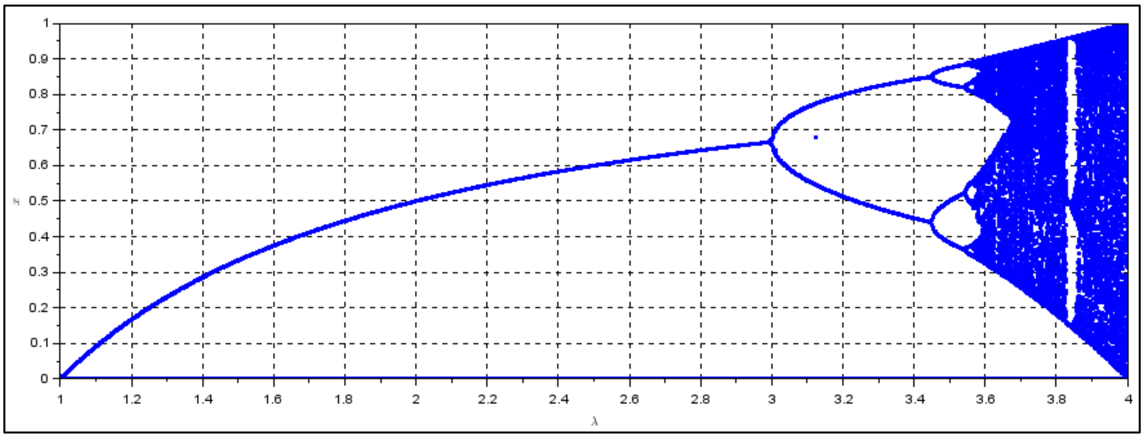


Figura 8. Diagrama de bifurcación de la función logística

Para una mejor interpretación del diagrama de bifurcación vamos a ejecutar de nuevo el programa de Verhulst para las diferentes regiones que aparecen en el diagrama.

Tomaremos los valores $\lambda = \{2.5, 3.2, 3.5, 3.7\}$. Representaremos solo la zona de interés que en este caso serán los valores de $x \in [0.2,1]$, tomando como semilla $x_0 = 0.5$. La figura 9 recoge los diagramas de Verhulst asociados.

11.4. Referencias bibliográficas

Devaney, R. (1989). *An introduction to chaotic dynamical systems*. Addison-Wesley.

Devaney, R. L. (1992). *A first course in chaotic dynamical systems*. Universidad de Boston.

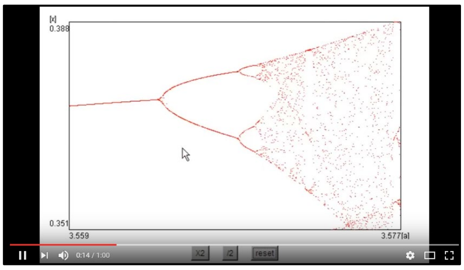
Giraldo, A. S. (2002). *Sistemas dinámicos discretos y caos. Teoría, ejemplos y algoritmos*. Universidad Politécnica de Madrid.

Lo + recomendado

No dejes de ver…

Bifurcation logistic map

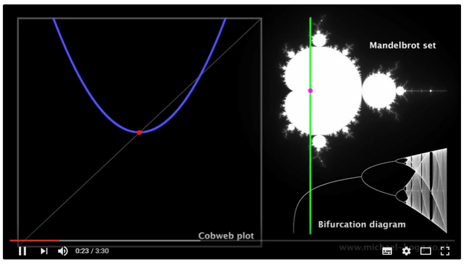
Este vídeo muestra el comportamiento en diferentes aumentos de cada una de las regiones para visualizar con mayor detalle el diagrama de bifurcación de la familia logística.



Accede al vídeo desde el aula virtual o a través de la siguiente dirección web:
https://www.youtube.com/watch?v=be4d8bTKjdM

Mandelbrot set – from order to chaos

En el enlace se muestra un vídeo interesante que muestra a la vez el diagrama de Verhulst, el punto del conjunto de Mandelbrot y el diagrama de bifurcación. En él se pueden identificar de un vistazo los diferentes comportamientos en función de la región a la que pertenece el parámetro.



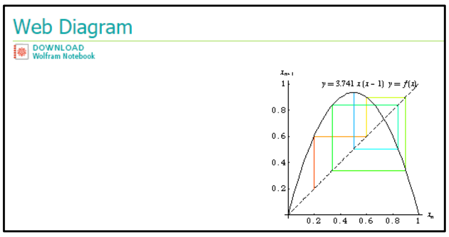
Accede al artículo desde el aula virtual o a través de la siguiente dirección web:
https://www.youtube.com/watch?v=VjzkW1IlVI4

+ Información

A fondo

Web diagram

En esta página de Wolfram Mathworld se describe cómo se genera el diagrama de Verhulst, además de ilustrar una animación de un miembro de la familia logística.



Accede al artículo desde el aula virtual o a través de la siguiente dirección web:
http://mathworld.wolfram.com/

Enlaces relacionados

Introducción a los sistemas dinámicos

De nuevo, el curso OCW de la Universidad Politécnica de Madrid supone una buena referencia para reutilizar y afianzar los conceptos vistos a lo largo del tema. Se presentan diferentes ejemplos de sistemas dinámicos de variable real.

Accede a la página desde el aula virtual o a través de la siguiente dirección web:
http://ocw.upm.es/matematica-aplicada/introduccion-a-los-sistemas-dinamicos

Modelización matemática de sistemas dinámicos

El curso OCW de la Universidad Politécnica de Cartagena es una buena ampliación sobre la modelización de sistemas dinámicos. Constituye un recurso útil para ampliar los contenidos de todo el curso y para visualizar ejemplos de aplicaciones a sistemas mecánicos y eléctricos.

OCW UPCT

Accede a la página a través de la siguiente dirección web:
https://ocw.bib.upct.es/course/view.php?id=109&topic=2

Bibliografía

Cordero, A., Torregrosa, J. R. y Vindel, P. (2013). *Dynamics of chebyshev-helley type methods. Applied mathematics and computation*. Elsevier.

Franco-Medrano, F. y Solís, F. J. (2015). *Stability of real parametric polynomial discrete dynamica systems*. Zhan Zhou.

Test

1. La diferencia entre iterados sirve:

A. Para dejar de iterar cuando el valor está por debajo de un determinado umbral.

B. Para dejar de iterar a partir de un número máximo de iteraciones.

C. Para dejar de iterar cuando el punto deja de moverse.

D. Ninguna de las anteriores es correcta.

2. El número máximo de iteraciones queda determinado:

A. Por las características del sistema.

B. Por el usuario.

C. Por el grado del polinomio.

D. Ninguna de las anteriores es correcta.

3. ¿Por qué motivo se sale del proceso iterativo que obtiene la órbita de una semilla en un sistema dinámico?

A. Porque se ha alcanzado un punto fijo.

B. Porque se ha agotado la memoria del sistema.

C. Porque se ha alcanzado una órbita periódica.

D. Porque el método ha divergido.

4. El diagrama de Verhulst representa además de la función que determina el comportamiento del sistema:

A. La función $y = -x$.

B. La función $y = x$.

C. La función $y = 2x$.

D. La función $y = -2x$.

5. El diagrama de Verhulst permite conocer:

A. La órbita de todas las posibles semillas en una sola representación.

B. La órbita de una semilla en cada representación.

C. La órbita de toda una familia de sistemas.

D. Ninguna de las anteriores es correcta.

6. Si se toma como semilla un punto fijo atractor del sistema, el diagrama de Verhulst:

A. No muestra la función $y = x$.

B. No muestra la función $y = f(x)$.

C. No muestra las flechas de la órbita.

D. Ninguna de las anteriores es correcta.

7. El diagrama de bifurcación representa el valor final de la iteración:

A. De todas las semillas para un valor de λ.

B. De una semilla para todos los valores λ.

C. De una semilla para un valor de λ.

D. De todas las semillas para todos los valores de λ.

8. Los puntos de bifurcación:

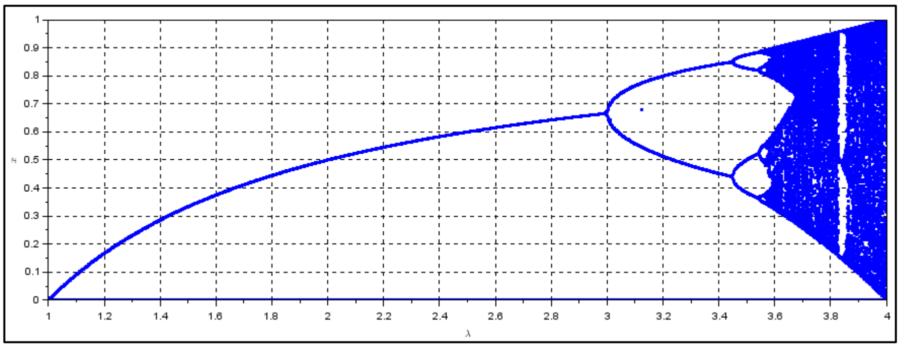
A. Son necesariamente puntos fijos.

B. No son ni atractores ni repulsores.

C. Se determinan a partir de la segunda derivada de la función del sistema.

D. Ninguna de las anteriores es correcta.

9. Dado el siguiente diagrama de bifurcación, ¿de qué periodo será la órbita del sistema cuando $\lambda = 3.2$:



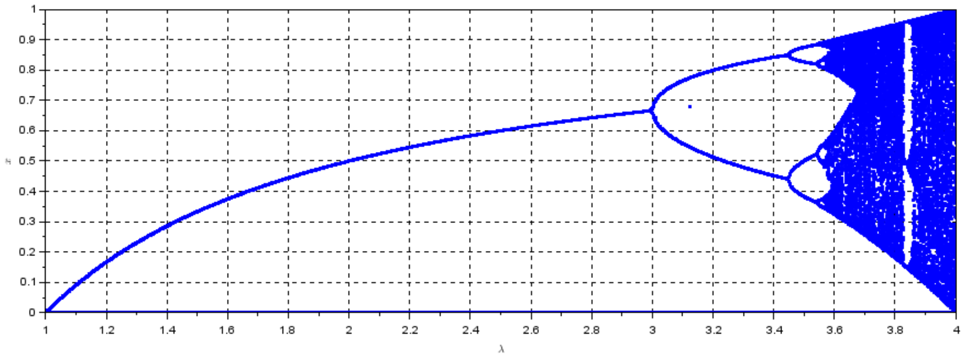
A. 1.

B. 2.

C. 4.

D. 8.

10. Dado el siguiente diagrama de bifurcación, ¿de qué periodo será la órbita del sistema cuando $\lambda = 3.5$?



A. 1.

B. 2.

C. 4.

D. 8.

