Generación de números pseudo aleatorios con mapas y osciladores caóticos

Dr. Luis Gerardo de la Fraga

Departamento de Computación
Cinvestav Zacatenco
Correo-e: fraga@cs.cinvestav.mx

5 de abril de 2022

Contenido

Introducción

1. Los números aleatorios solo existen en fuentes físicas:
   - como lanzar dados al aire,
   - en una ruleta,
   - en el ruido blanco de los aparatos eléctricos,
   - en el estado de un flip-flop al encenderlo, etc.
2. Los números pseudo aleatorios son los que se pueden generar dentro de la computadora con un proceso determinista.

Introducción

1. En los GNA sería imposible generar una misma secuencia de números aleatorios.
2. En un GNPA la secuencia depende de un valor inicial o *semilla*.
3. Una semilla debería generarse con un GNA.
4. Los GNPA tienen muchas aplicaciones en:
   - Simulaciones de Monte Carlo
   - Algoritmos evolutivos
   - Video juegos
5. Un GNPA es uno de los objetos básicos de la criptografía (additive stream ciphers) [Kocarev(2001)].

Chaos

- El caos existe en entidades que están definidas como:
- Muy sensibles a sus condiciones iniciales,
- deben tener un exponente de Lyapunov positivo.

Mapas caóticos

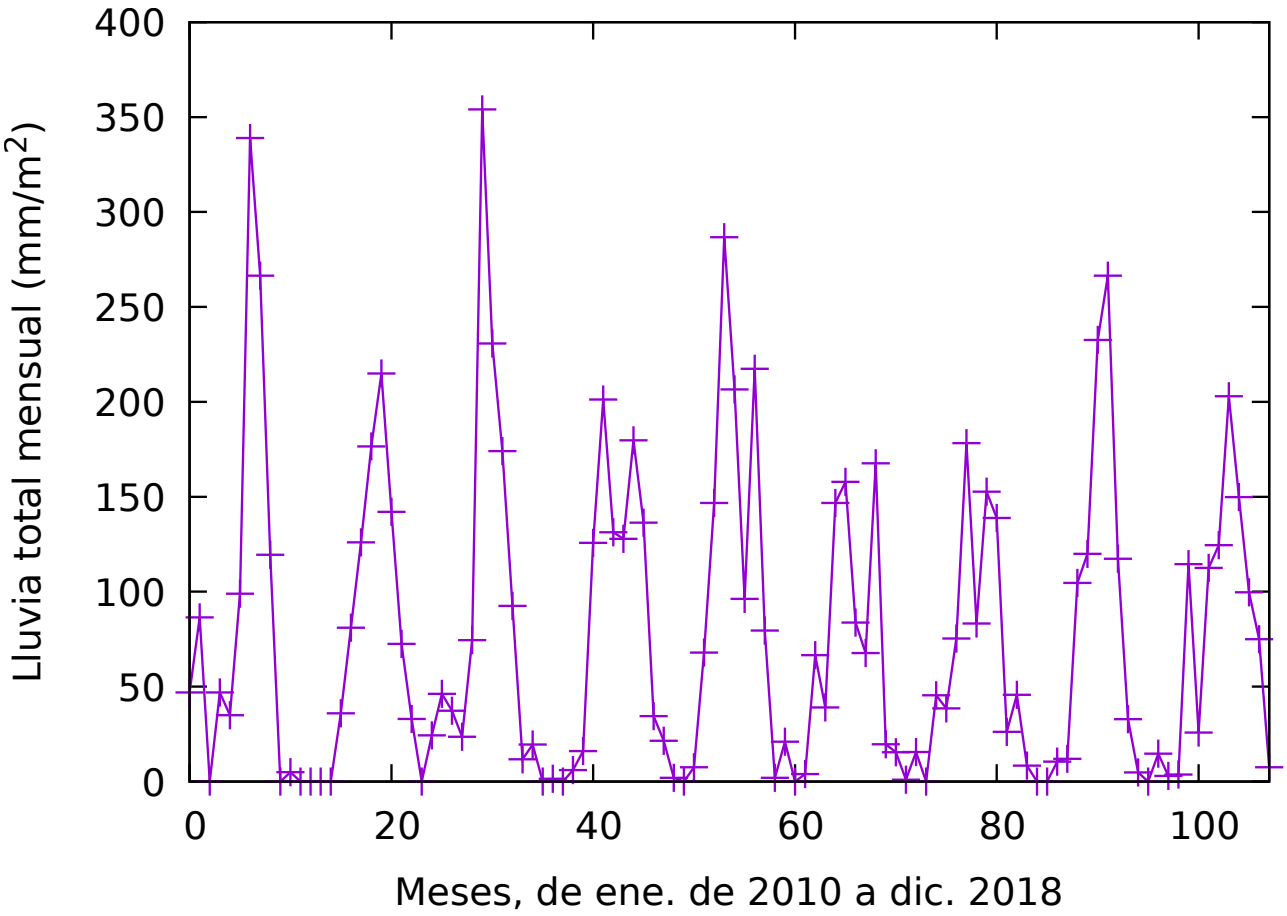
1. Son sistemas discretos
2. Pueden ser de una, dos o más dimensiones
3. Dependen de parámetros y valores iniciales

Osciladores caóticos

1. Son sistemas continuos
2. Están definidos por un sistema de ecuaciones diferenciales
3. Pueden ser de tres o más dimensiones
4. Dependen de parámetros, valores iniciales, también
5. del método de integración de las ecuaciones diferenciales,
6. y del valor del paso de integración.

Exiten señales que se consideren caóticas (se les llama *series de tiempo*) y que no están definicas por una o unas ecuaciones.

- datos financieros,
- datos del clima,
- señales biológicas del organismo (nivel de glucosa durante el día).



https://smn.conagua.gob.mx/tools/RESOURCES/Mensuales/pue/00021096.TXT

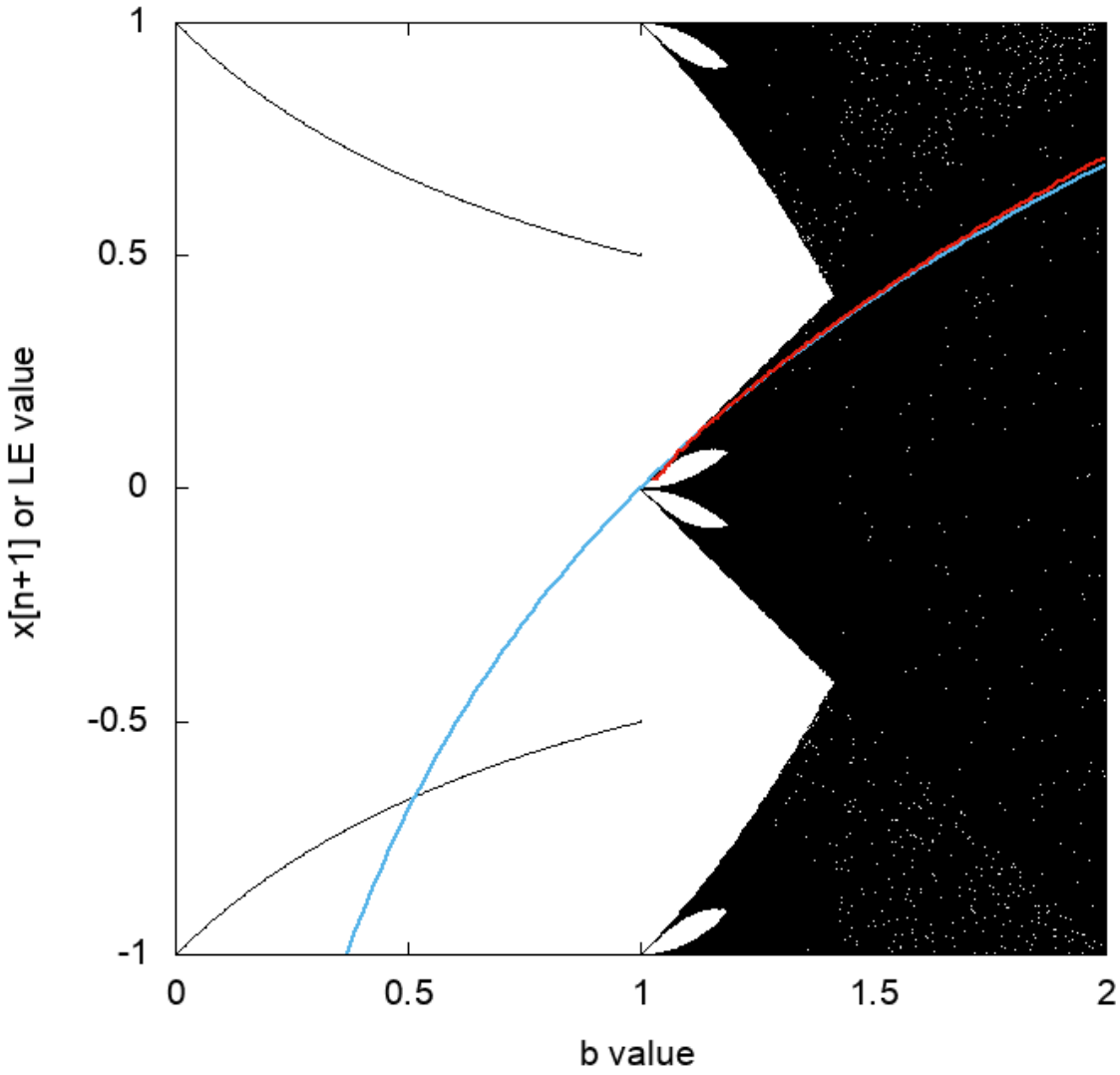
SANTA RITA TLAHUAPAN, TL

Mapa de corrimiento de Bernouli

$$x_{n+1} = \begin{cases} bx_n - a, & \text{if } x_n \geq 0, \\ bx_n + a, & \text{if } x_n < 0. \end{cases}$$

El parámetro a es un factor de escala para que la salida esté en el intervalo $[-a, a]$.

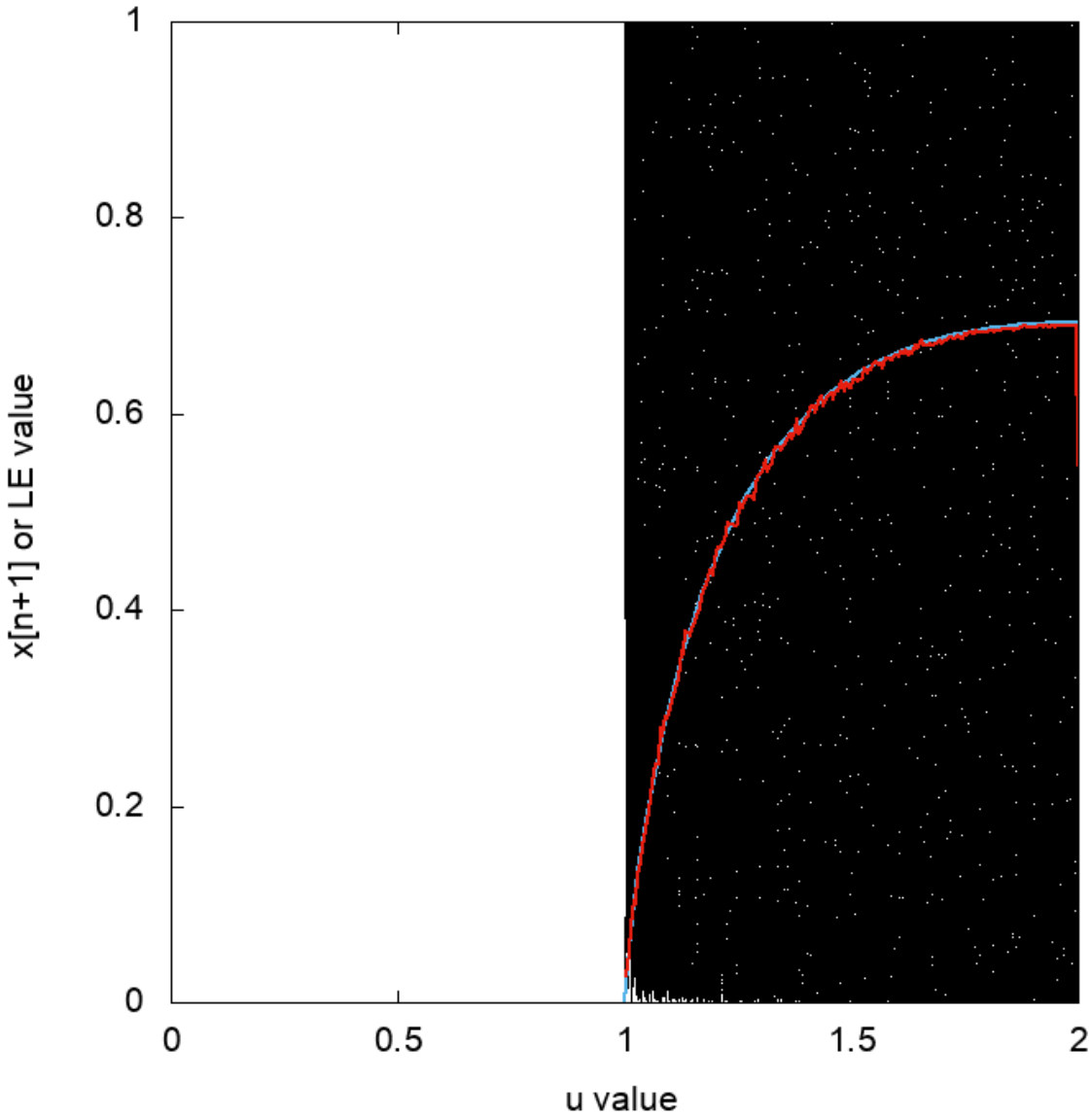
Diagrama de bifurcación para el mapa de Bernouli



Mapa tienda de campaña

$$x_{n+1} = \begin{cases} u\, x_n, & \text{for } x_n \in [0, \frac{1}{u}], \\ \frac{u}{u-1}(1 - x_n), & \text{for } x_n \in (\frac{1}{u}, 1]. \end{cases}$$

Diagrama de bifurcación para el mapa tienda de campaña

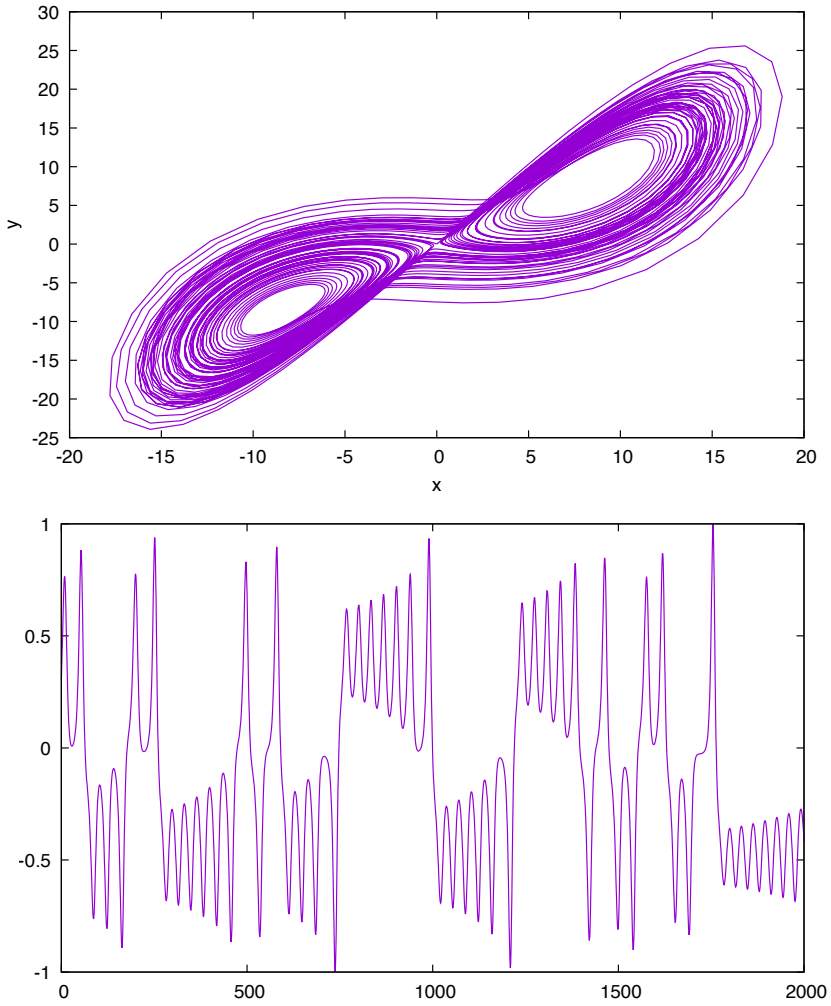


El sistema de Lorenz

$$\dot{x} = \sigma(y - x),$$
$$\dot{y} = x(\rho - z/k) - y,$$
$$\dot{z} = xy/k - \beta z.$$

Para los valores , $(x_0, y_0, z_0) = (5.0, 10.0, 15.0)$, $\sigma = 10$, $\rho = 28$, $\beta = 2.6667$.
El sistema se debe de integrar, usando el método de Euler:

$$x_{t+1} = h[\sigma(y_t - x_t)],$$
$$y_{t+1} = h[x_t(\rho - z_t/k) - y_t],$$
$$z_{t+1} = h[x_t y_t/k - \beta z_t].$$



Con $h = 0.001$, y cada muestra se tomó cada 20 pasos de integración.

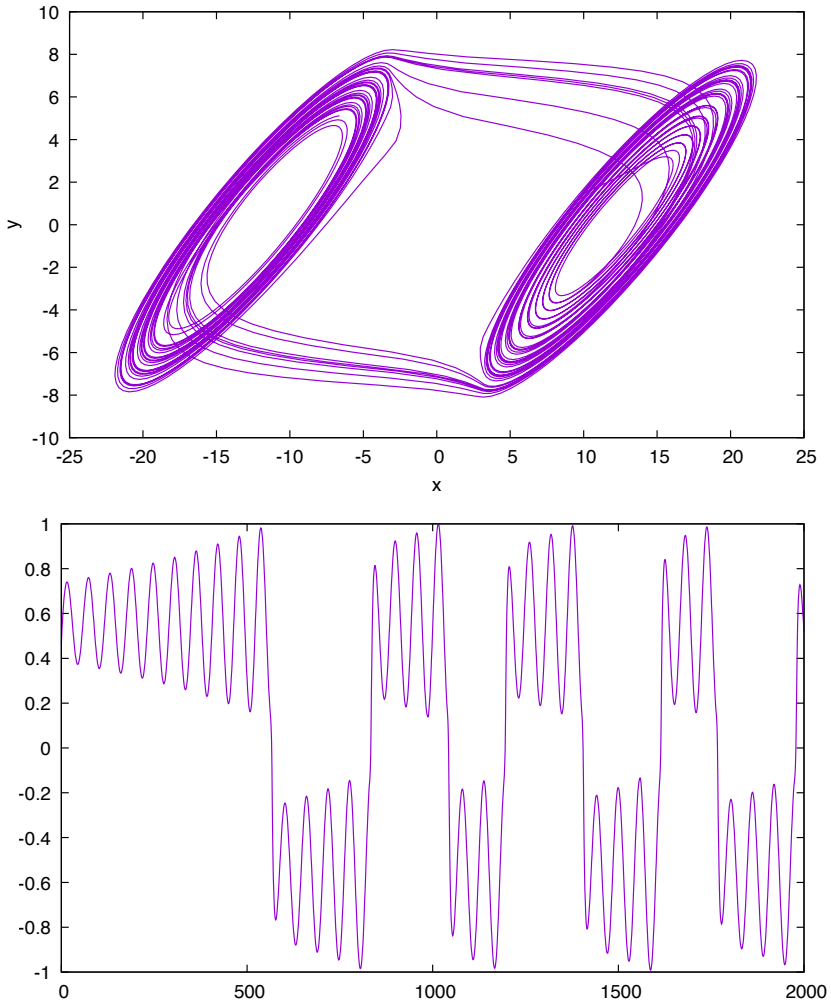
El oscilador de Chua

$$\dot{x} = \alpha[y - x - kg(x/k)],$$
$$\dot{y} = x - y + z,$$
$$\dot{z} = -\beta y,$$

Para los valores , $(x_0, y_0, z_0) = (10.0, 1.5, 0.0)$, $\alpha = 10$, $\beta = 15$, $k = 32$.

$$g(x) = p_2 x + 0.5(p_1 - p_2)(|x + p_3| - |x - p_3|),$$

para $p_1 = -3.036$, $p_2 = -0.276$, y $p_3 = 0.1$.



Con $h = 0.001$, y cada muestra se tomó cada 20 pasos de integración.

Formas de generar números aleatorios

1. Con mapas, aplicando un umbral y corrigiendo la secuencia generada.
2. Con mapas, aplicando la operación mod 255
3. GNPA existentes
4. ¿Con osciladores?

Generación de secuencias binarias aleatorias

Una secuencia de bits puede realizarse como

$$b_{n+1} = \begin{cases} 0 & \text{si } x_{n+1} < q, \\ 1 & \text{si } x_{n+1} \geq q, \end{cases}$$

para un x_{n+1} calculado por cualquier mapa y para un valor de umbral q.
Esta secuencia es pseudo aletoria porque se puede generar exactamente igual a partir de las condiciones iniciales del mapa.

L.G. de la Fraga, E. Torres-Pérez, E. Tlelo-Cuautle, and C. Mancillas-López, Hardware Implementation of pseudo-random number generators based on chaotic maps, Nonlinear Dynamics (2017), Volume 90, Issue 3, pp. 1661–1670, DOI: 10.1007/s11071-017-3755-z

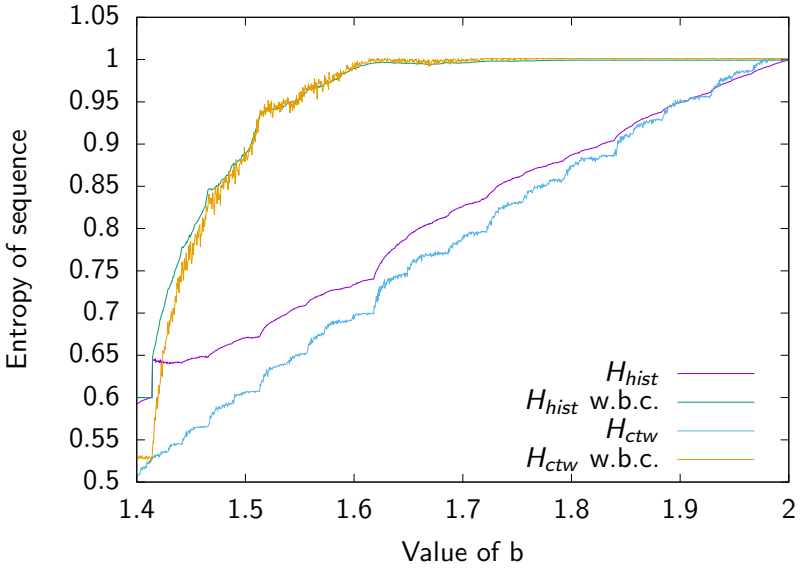
Entropía

La entropía es una medida de que tan impredecible es una secuencia binaria.

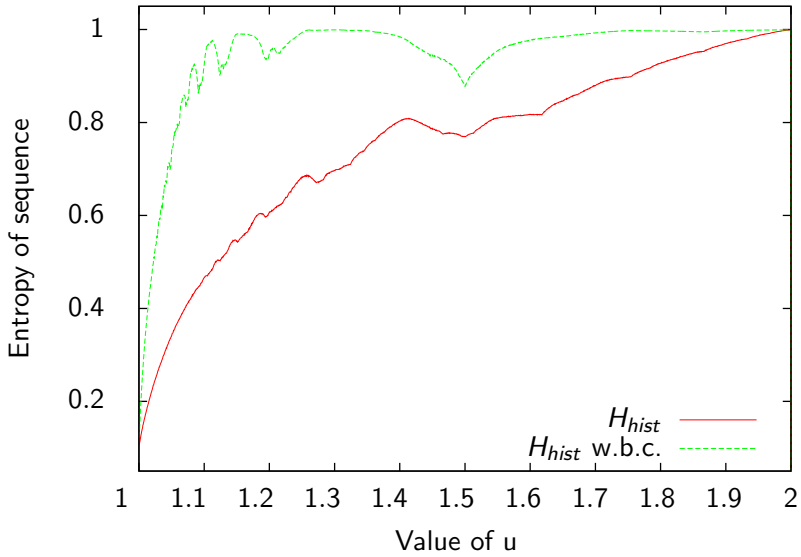
$$H(b) = -\sum_{i=1}^{n} p_i \log_2 p_i,$$

- donde b es la secuencia construida con los símbolos $\{0, 1\}$,
- p_i distribución de probabilidades $\{p_1, p_2, \ldots, p_n\}$,
- La entropía es igual a 1 para una secuencia totalmente impredecible
- La entropía es igual a 0 para una secuencia totalmente predecible

- Las secuencias se tuvieron que corregir usando la técnica llamada *reducción de conteo de bits*,
- Cada 5 bits se realiza la operación xor entre ellos.
- La entropía se midió con secuencias de 1 millón de bits e histogramas de subsecuencias de tamaño 10 símbolos
- También se aplicó la *poderación con el árbol de contexto*, que requiere secuencias de 5 mil símbolos pero es mucho más tardado que con histogramas. Este árbol se usa también para realizar los códigos de Huffman.



(a) Bernoulli shift map



(b) Tent map

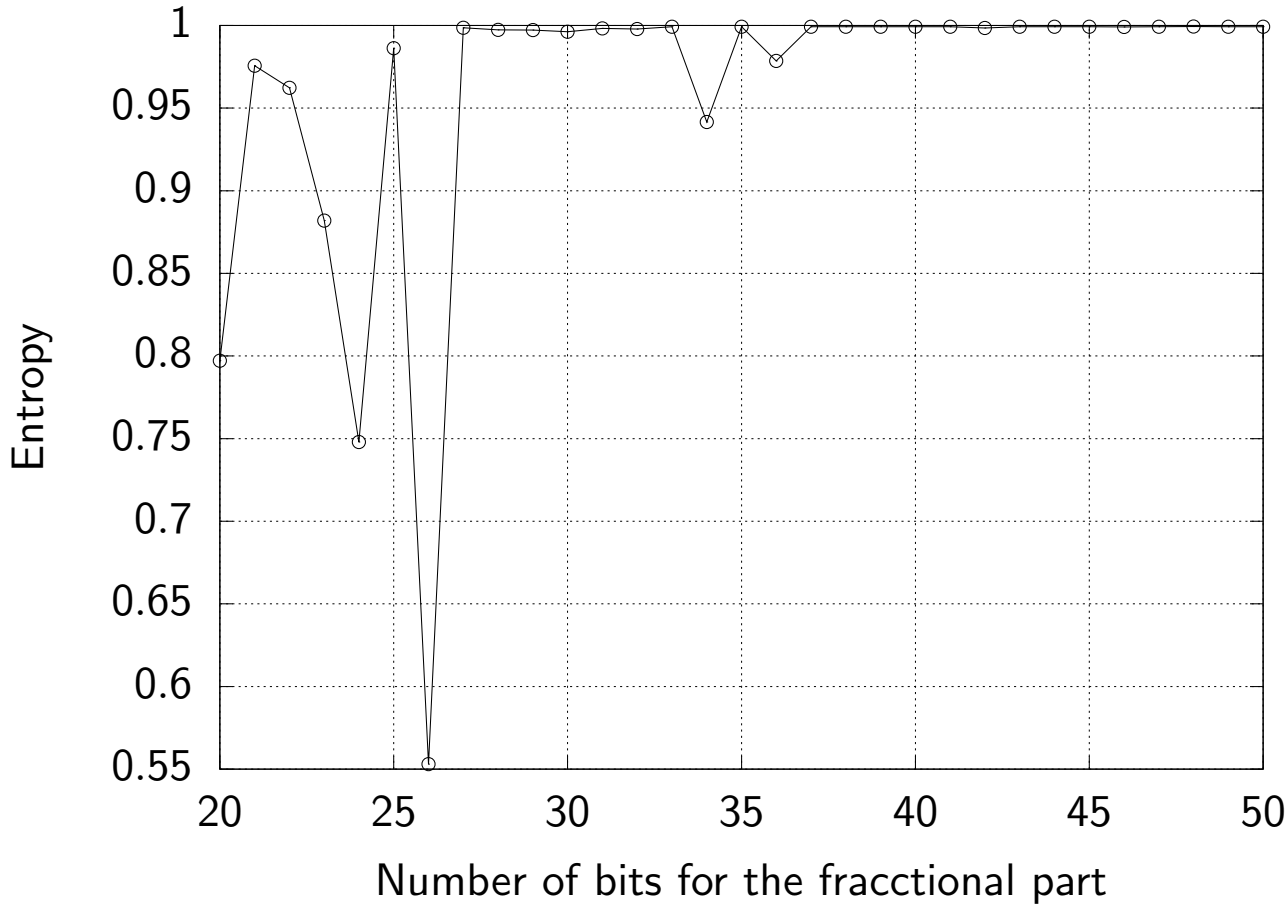
El National Institute of Standards and Technology (NIST) de los Estados Unidos de América ha creado un software libre para probar genenerадores de números aleatorios y pseudo aleatorios.

Results of applying NIST test to our PRNG implemented in floating and fixed point arithmetic.

	Test name	Bernoulli shift map				Tent map		Zigzag map	
		Floating point		Fixed arithmetic		Fixed arithmetic		Fixed arithmetic	
		p-value	prop.	*p*-value	prop.	*p*-value	prop.	*p*-value	prop.
1	Frequency	0.419021	0.98	0.699313	0.98	0.911413	1.00	0.514124	0.99
2	Block frequency	0.678686	1.00	0.678686	0.99	0.883171	0.98	0.075719	0.99
3	Cumulative sums +	0.534648	0.98	0.710364	0.98	0.118131	1.00	0.265410	0.97
4	Runs	0.366918	1.00	0.719747	0.99	0.304126	0.99	0.162606	1.00
5	Longest run of ones	0.455937	1.00	0.236810	1.00	0.213309	1.00	0.122325	0.99
6	Rank *	0.001030	0.99	0.935716	0.99	0.911413	1.00	0.971699	0.99
7	Spectral DFT	0.003712	1.00	0.494392	0.99	0.010988	0.99	0.026948	0.99
8	Nonperiodic templates *+	0.474987	0.99	0.488481	0.99	0.441509	0.99	0.479177	0.99
9	Overlapping templates *	0.191687	0.97	0.924076	0.98	0.401199	1.00	0.048716	1.00
10	Universal statistical *	0.455937	0.99	0.964295	0.98	0.437274	1.00	0.534146	0.99
11	Approximate entropy	0.013569	1.00	0.779188	0.99	0.304126	1.00	0.366918	0.99
12	Random excursion *	0.580777	0.99	0.225431	0.98	0.285487	1.00	0.355641	0.99
13	Random excursion variant *	0.525278	0.99	0.445670	0.99	0.309401	1.00	0.224650	0.99
14	Serial +	0.301860	1.00	0.456707	1.00	0.465962	0.99	0.331048	0.99
15	Linear complexity	0.816537	0.97	0.129620	0.98	0.759756	0.99	0.719747	0.98

Aritmética de punto fijo

- Es bueno conceptualizar el GNPA como un hardware embebido en la computadora.
- Se realizan prototipos en FPGAs
- Se usa punto fijo en vez de punto flotante
- Se tiene un número $A(a.b)$, con a bits en la parte entera, b bits en la parte fraccional.
- La suma de dos números $A(a.b)$ resulta en un número $A((a+1).b)$ [Tlelo-Cuautle et al.(2016)Tlelo-Cuautle, Rangel-Magdaleno, and de
- La multiplicación de dos números $A(a.b)$ resulta en un número $A((2a+1).2b)$.
- a se escoge lo sufientemente grande para mantener los resultados de las sumas y multiplicaciones.
- El número $A(a, 2b)$ se debe regresar a la representación $A(a, b)$ por corrimiento de b bits a la derecha.



Entropía vs. el número de bits usado en la parte fraccional para implementar el GNPA con aritmética de punto fijo. Esta gráfica es discreta, las líneas se adicionaron solo con el propósito de claridad.

Existen tres GNPA que se pueden utilizar en aplicaciones de Monte Carlo:

1. RANLUX
2. RANLUX++
3. MIXMAX

F. James and L. Moneta, Review of High-Quality Random Number Generators, Computing and Software for Big Science (2020) 4:2, DOI: 10.1007/s41781-019-0034-3

Estos GNPA están basados en

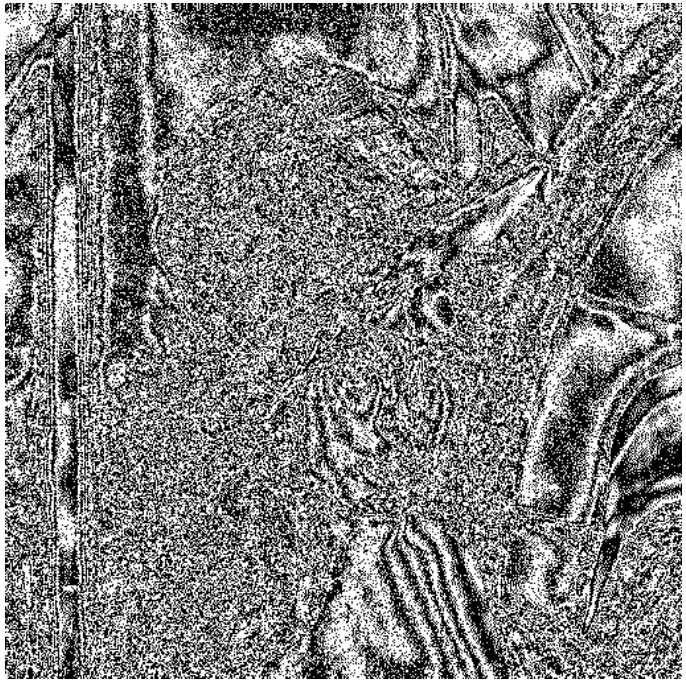
$$x(i+1) = A\, x(i) \bmod 1,$$

- donde la matriz A es de tamaño 8 hasta unos cientos,
- x es un vector de números reales,
- todos los elementos de A son enteros,
- $\det(A) = 1$,
- todos los eigenvalores de A deben de estar tan lejos como se pueda del círculo unitario.
- Se deben decargar un número de vectores para asegurar la independencia de vectores sucesivos.
- MIXMAX usa una matriz A de tamaño 256×256.

En el paper

L.G. de la Fraga, C. Mancillas-López, E. Tlelo-Cuautle, Designing an authenticated Hash function with a 2D chaotic map. Nonlinear Dynamics, Vol. 104, pp. 4569-4580 (2021). DOI: 10.1007/s11071-021-06491-3

Se usa el mapa 2D y los bits menos significativos son los que se toman como aleatorios

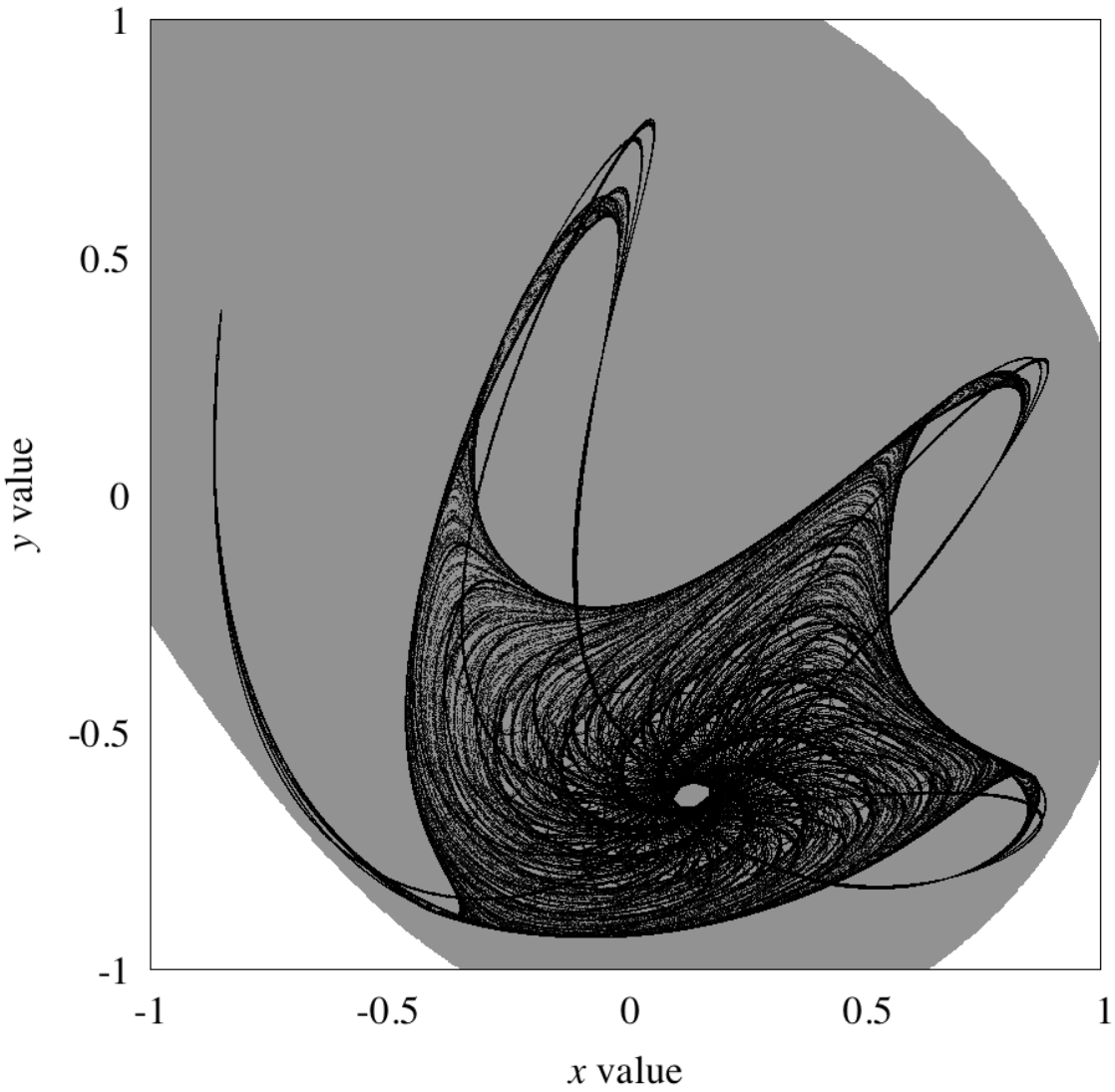


Mapa 2D

$$x_{n+1} = a_1 + a_2 x_n + a_3 x_n^2 + a_4 x_n y_n + a_5 y_n + a_6 y_n^2,$$
$$y_{n+1} = a_7 + a_8 x_n + a_9 x_n^2 + a_{10} x_n y_n + a_{11} y_n + a_{12} y_n^2.$$

Los valores de los coeficientes se buscaron exhaustivamente en una computadora [Sprott(1993)].

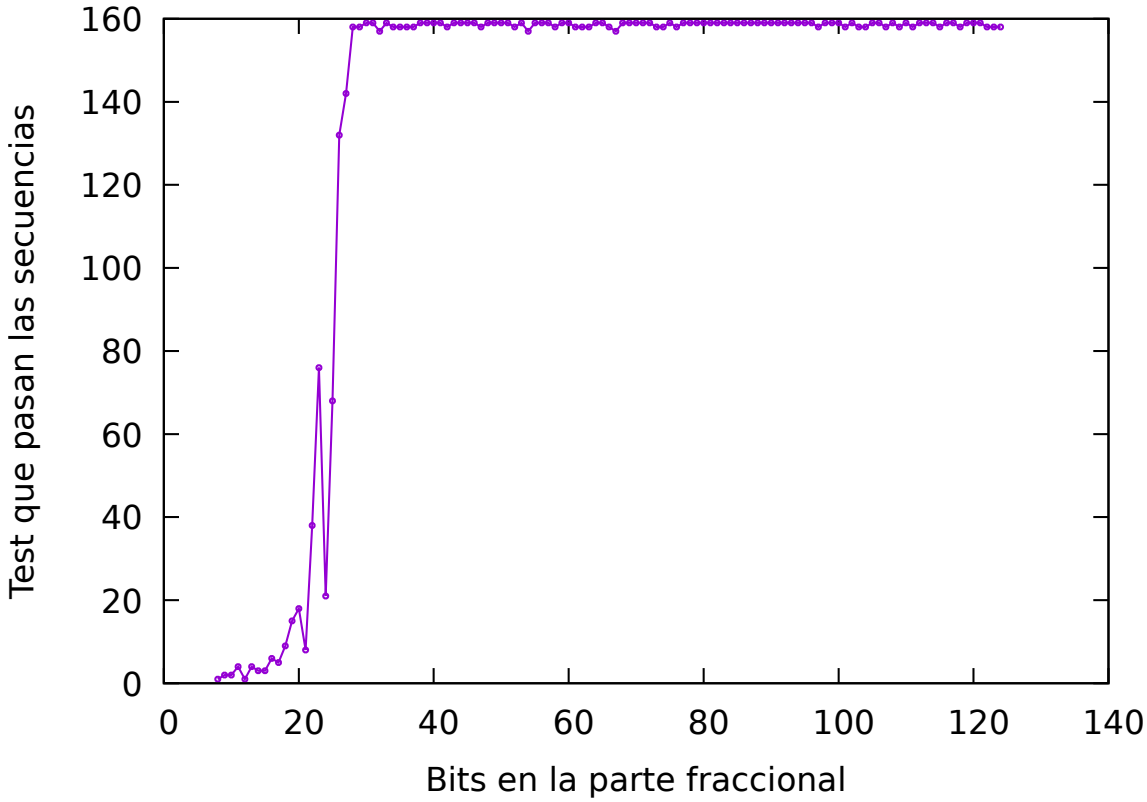
Para los valores
$\{-0.6, -0.1, 1.1, 0.2, -0.8, 0.6, -0.7, 0.7, 0.7, 0.3, 0.6, 0.9\}$
El dibujo de la fase (en negro) y su dominio de atracción (en gris)



La secuencia aleatoria se construye como:

$$s_{n+1} = \{x_{n+1} \bmod 256,\ y_{n+1} \bmod 256\}.$$

Las pruebas TestU01 para 100 secuencias de 1 millón de bits:



Una aplicación: la evolución diferencial en aritmética de punto fijo

Luis Gerardo de la Fraga, Differential Evolution under Fixed Point Arithmetic and FP16 Numbers, Mathematical and Computational Applications, Number 13, Vol. 26, 2021, DOI: 10.3390/mca26010013

Require: The search space and the value v for the stop condition. The values for population size, μ; maximum number of generations, g; difference and recombination constants, F and R respectively.
Ensure: A solution of the minimization problem

initialize($P = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_\mu\}$)
evaluate(P)
$k = 0$
repeat
  for $i = 1$ to μ **do**
    Let r_1, r_2 and r_3 be three random integers in $[1, \mu]$, such that $r_1 \neq r_2 \neq r_3$
    Let j_{rand} be a random integer in $[1, n]$
    for $j = 1$ to n **do**
      $$x'_j = \begin{cases} x_{r_3,j} + F(x_{r_1,j} - x_{r_2,j}) & \text{if } U(0,1) < R \text{ or } j = j_{\text{rand}} \\ x_{i,j} & \text{otherwise} \end{cases}$$
      if $x'_j < l_i$ or $x'_j > u_i$ **then** ▷ Check bounds
        $x'_j = U(0,1)(u_i - l_i) + l_i$
      end if
    end for
    if $f(\mathbf{x}') < f(\mathbf{x}_i)$ **then**
      $\mathbf{x}_j = \mathbf{x}'$
    end if
  end for
  min $= f(\mathbf{x}_1)$, max $= f(\mathbf{x}_1)$
  for $i = 2$ to μ **do**
    if $f(\mathbf{x}_i) <$ min **then**
      min $= f(\mathbf{x}_i)$
    end if
    if $f(\mathbf{x}_i) >$ max **then**
      max $= f(\mathbf{x}_i)$
    end if
  end for
  $k \leftarrow k + 1$
until $(\max - \min) < v$ or $k > g$

1: initialize($P = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_\mu\}$)
2: evaluate(P)
3: $k = 0$
4: **repeat**
5: **for** $i = 1$ to μ **do**
6: Let r_1, r_2 and r_3 be three random integers in $[1, \mu]$, such that $r_1 \neq r_2 \neq r_3$
7: Let j_{rand} be a random integer in $[1, n]$
8: **for** $j = 1$ to n **do**
9: $$x'_j = \begin{cases} x_{r_3,j} + F(x_{r_1,j} - x_{r_2,j}) & \text{if U(0,1)} < \text{R or } j = j_{\text{rand}} \\ x_{i,j} & \text{otherwise} \end{cases}$$
10: **if** $x'_j < l_i$ or $x'_j > u_i$ **then** ▷ Check bounds
11: $x'_j = U(0,1)(u_i - l_i) + l_i$
12: **end if**
13: **end for**
14: **if** $f(\mathbf{x}') < f(\mathbf{x}_i)$ **then**
15: $\mathbf{x}_j = \mathbf{x}'$
16: **end if**
17: **end for**
18: **for** $i = 2$ to μ **do**
19: min = best individual
20: max = worst individual
21: **end for**
22: **until** $(\max - \min) < s$ or $k > g$

Para la evolución diferencial necesitamos una función que nos genere números aleatorios en un intervalo $[0, a]$ como en

> *D. Lemire, Fast Random Integer Generation in an Interval, ACM Transactions on Modeling and Computer Simulation, 29(1), 2019, DOI: 10 . 1145/ 3230636 .*

Conclusiones

1. Vimos que es un generador de números aleatorios
2. Vimos que es un generador de números pseudo aleatorios
3. Vimos mapas y osciladores caóticos
4. Vimos como realizar un GNPA con mapas
5. Vimos su realización con aritmética en punto fijo
6. Vimos una aplicación de un GNPA en el algoritmo de la evolución difencial

Esta charla y el software

1. El PDF de esta charla y el software en python de las simulaciones está disponible en `https://cs.cinvestav.mx/~fraga/Charlas`
2. La biblioteca para realizar números alearios con una distribución uniforme en el intervalo $[0, 2^{16} - 1]$ junto con otra función para generar números aleatorios, también con una ditribución uniforme en el intervalo $[0, s]$ con $s < 2^{16} - 1$, está disponible en: `https://cs.cinvestav.mx/~fraga/Softwarelibre/Myrand.tar.gz`
3. El procedimiento en Cython para llamar a las funciones en C también está incluido en la URL del punto anterior.

L. Kocarev.

Chaos-based cryptography: a brief overview.

IEEE Circuits Syst. Mag., 1(3):6–21, 2001.

E. Tlelo-Cuautle, J.J. Rangel-Magdaleno, and L.G. de la Fraga.

Engineering Applications of FPGAs.

Springer, 2016.

doi: 10.1007/978-3-319-34115-6.

J.C. Sprott.

Automatic generation of strange attractors.

Computers & Graphics, 17(3):325 – 332, 1993.

ISSN 0097-8493.

doi: 10.1016/0097-8493(93)90082-K.