Implementación de Renyi Maps acoplados y aplicación de pruebas NIST.

Marcos Daniel Calderón Calderón

Abstract

En este reporte, se explica a detalle la implementación de varios mapas caóticos acoplados. En este caso, se utilizará el mapa Renyi.

1. Introducción: Sistemas acoplados.

Un sistema acoplado puede estar representado por medio de ecuaciones diferenciales o discretas. En este tipo de sistemas, las razones de cambio dependen de más variables. Un ejemplo sencillo puede ser el siguiente sistema de ecuaciones diferenciales:

$$\frac{dC}{dt} = \alpha C - \beta CZ \quad \frac{dZ}{dt} = -\gamma Z + \delta CZ \tag{1}$$

Un ejemplo de sistema discreto acoplado es el siguiente:

$$X_{i,j} = (1-\epsilon) f(X_{i,j-1}) + \epsilon H(X_{i,j-1}, ..., X_{N,j-1}) \tag{2}$$

donde:

$$H(X_{i,j-1}, ..., X_{N,j-1}) = \sum_{i=1}^{N} w_i X_{i,j-1}. \tag{3}$$

El indice i representa cada una de los N mapas que se van a acoplar, el índice j representa a las evaluaciones anteriores de cada uno de los mapas.

2. Criterios utilizados para el acoplamiento de mapas.

Se utilizaron diversas maneras para el acoplamiento de los mapas; a continuación, explicamos cada una de ellas.

Email address: marcos.calderon@cimat.mx (Marcos Daniel Calderón Calderón)

2.1. Criterio 1.

El primer criterio utilizado es el siguiente:

$$X_{i,j} = f(X_{i,j-1}) + \epsilon H(X_{i,j-1}, ..., X_{N,j-1}) \tag{4}$$

donde:

$$H(X_{i,j-1}, ..., X_{N,j-1}) = \sum_{i=1}^{N} (X_{i,j-1} \mod 256). \tag{5}$$

además, ϵ es un valor aleatorio del conjunto: $\{-1, 0, 1\}$

2.2. Criterio 2.

El segundo criterio utilizado es el siguiente:

$$X_{i,j} = f(X_{i,j-1}) + \epsilon H(X_{i,j-1}, ..., X_{N,j-1}) \tag{6}$$

donde:

$$H(X_{i,j-1}, ..., X_{N,j-1}) = \bigoplus_{i=1}^{N} (X_{i,j-1}). \tag{7}$$

donde $\bigoplus$ representa la operación XOR, además ϵ es un valor aleatorio del conjunto: $\{-1, 0, 1\}$

2.3. Criterio 3.

El tercer criterio utilizado es el siguiente:

$$X_{i,j} = \hat{\gamma} \bigoplus f(X_{i,j-1}) + \gamma \bigoplus H(X_{i,j-1}, ..., X_{N,j-1}) \tag{8}$$

donde:

$$H(X_{i,j-1}, ..., X_{N,j-1}) = \bigoplus_{i=1}^{N} (X_{i,j-1}). \tag{9}$$

donde $\bigoplus$ representa la operación XOR, además para el cálculo del valor de γ, necesitamos un ϵ que será un valor aleatorio donde $1 \geq \epsilon \geq 32$, el rango es establecido de acuerdo al tipo de dato utilizado cuando se traduce el sistema a un lenguaje de programación elegido. Ahora, podemos calcular γ de la siguiente manera:

$$\gamma = 2^{\epsilon} - 1 \tag{10}$$

$$\hat{\gamma} = M - \gamma \tag{11}$$

donde $M = 2^{32} - 1$.

3. Detalles de implementación.

A continuación, se especifican algunos detalles importantes a tomar en cuenta a la hora de la implementación de los mapas acoplados:

- El lenguaje de programación utilizado será **C**.
- Se trabajará con el tipo de dato **unsigned long**; por lo tanto, se dispone de 32 bits (4 bytes) para la representación de los datos. Por lo tanto, tenemos un rango de $(0 - 4,294,967,295 = 2^{32} - 1)$.
- Se buscaron 80000 valores en la ejecución del programa; además, cada valor está compuesto de 32 bits cuando se guarda la información en un archivo binario. Por las características mencionadas anteriormente, se pueden leer **2,560,000 bits** para la aplicación de las pruebas NIST.
- Para equipos de cómputo de 32 bits, se utilizó el tipo de dato **unsigned long**, que está conformado de 4 bytes. Si el equipo de cómputo utilizado es de 64 bits, se recomienta utilizar el tipo de dato **unsigned int**, todavía no hay un estándar definido para el tamaño de los tipos de datos en los equipos de 64 bits.

4. Resultados.

4.1. Aplicación de las pruebas NIST a los mapas acoplados del Criterio 1.

Recordemos que el criterio 1 era el siguiente:

$$X_{i,j} = f(X_{i,j-1}) + \epsilon H(X_{i,j-1}, ..., X_{N,j-1}) \tag{12}$$

donde:

$$H(X_{i,j-1}, ..., X_{N,j-1}) = \sum_{i=1}^{N} (X_{i,j-1} \mod 256). \tag{13}$$

además, ϵ es un valor aleatorio del conjunto: $\{-1, 0, 1\}$

Se ejecutó el siguiente código:

- **./assess 2560000**
- User Prescribed Input File: **binarioSUMA.dat**
- Enter 0 if you DO NOT want to apply all of the statistical tests to each sequence and 1 if you DO. Enter chice: **1**
- How many bitstreams? **1**
- Input File Format: [0] ASCII - A sequence of ASCII 0's and 1's [1] Binary - Each byte in data file contains 8 bits of data

  Select input mode: **1**

Se obtuvieron los siguientes resultados:

Cuadro 1: Resultados de las pruebas de aleatoriedad NIST a los datos binarioSUMA.dat .

Prueba Aplicada	P-Valor	Exito?
Aproximate Entropy	0.241029	√
Block Frecuency	0.464753	√
Cumulative Sums	Forward test: 0.628651, Reverse test: 0.848525	√
FFT	0.149985	√
Frecuency	0.625019	√
Linear Complexity	0.595287	√
Longest Run	0.511949	√
Non Overlapping Template	P-valores aceptados: 145 de 148	√
Overlapping Template	0.037821	√
Random Excursions	P-valores aceptados: 8 de 8	√
Random Excursions Variant	P-valores aceptados: 18 de 18	√
Rank	0.111388	√
Runs	0.957015	√
Serial	P-valores aceptados: 2 de 2	√
Universal	0.136601	√

4.2. Aplicación de las pruebas NIST a los mapas acoplados del Criterio 2.

El criterio 2 era el siguiente:

$$X_{i,j} = f(X_{i,j-1}) + \epsilon H(X_{i,j-1}, ..., X_{N,j-1}) \tag{14}$$

donde:

$$H(X_{i,j-1}, ..., X_{N,j-1}) = \bigoplus_{i=1}^{N} (X_{i,j-1}). \tag{15}$$

donde $\bigoplus$ representa la operación XOR, además ϵ es un valor aleatorio del conjunto: $\{-1, 0, 1\}$

Para el criterio 2, se ejecutó el siguiente código:

- **./assess 2560000**
- User Prescribed Input File: **binarioXOR.dat**
- Enter 0 if you DO NOT want to apply all of the statistical tests to each sequence and 1 if you DO. Enter chice: **1**
- How many bitstreams? **1**
- Input File Format: [0] ASCII - A sequence of ASCII 0's and 1's [1] Binary - Each byte in data file contains 8 bits of data

  Select input mode: **1**

Se obtuvieron los siguientes resultados:

Cuadro 2: Resultados de las pruebas de aleatoriedad NIST a los datos binarioXOR.dat .

Prueba Aplicada	P-Valor	Exito?
Aproximate Entropy	0.858228	√
Block Frecuency	0.118270	√
Cumulative Sums	Forward test: 0.021856, Reverse test: 0.035757	√
FFT	0.787496	√
Frecuency	0.022608	√
Linear Complexity	0.791809	√
Longest Run	0.378018	√
Non Overlapping Template	P-valores aceptados: 144 de 148	√
Overlapping Template	0.457920	√
Random Excursions	NOT APPLICABLE.	
Random Excursions Variant	NOT APPLICABLE.	
Rank	0.278871	√
Runs	0.019569	√
Serial	P-valores aceptados: 2 de 2	√
Universal	0.249147	√

4.3. Aplicación de las pruebas NIST a los mapas acoplados del Criterio 3.

EL criterio 3 era el siguiente:

$$X_{i,j} = \hat{\gamma} \bigoplus f(X_{i,j-1}) + \gamma \bigoplus H(X_{i,j-1}, ..., X_{N,j-1}) \tag{16}$$

donde:

$$H(X_{i,j-1}, ..., X_{N,j-1}) = \bigoplus_{i=1}^{N} (X_{i,j-1}). \tag{17}$$

Para el criterio 3, se ejecutó el siguiente código:

- **./assess 2560000**
- User Prescribed Input File: **binarioXORcomp.dat**
- Enter 0 if you DO NOT want to apply all of the statistical tests to each sequence and 1 if you DO. Enter chice: **1**
- How many bitstreams? **1**
- Input File Format: [0] ASCII - A sequence of ASCII 0's and 1's [1] Binary - Each byte in data file contains 8 bits of data

  Select input mode: **1**

Se obtuvieron los siguientes resultados:

Cuadro 3: Resultados de las pruebas de aleatoriedad NIST a los datos binarioXORcomp.dat .

Prueba Aplicada	P-Valor	Exito?
Aproximate Entropy	0.979174	√
Block Frecuency	0.283892	√
Cumulative Sums	Forward test: 0.320736, Reverse test: 0.476676	√
FFT	0.990848	√
Frecuency	0.286876	√
Linear Complexity	0.142002	√
Longest Run	0.106050	√
Non Overlapping Template	P-valores aceptados: 145 de 148	√
Overlapping Template	0.879647	√
Random Excursions	P-valores aceptados: 8 de 8	√
Random Excursions Variant	P-valores aceptados: 18 de 18	√
Rank	0.398102	√
Runs	0.964539	√
Serial	P-valores aceptados: 2 de 2	√
Universal	0.603939	√

5. Conclusiones.

En general, los criterios 1 y 3 arrojan mejores resultados que el criterio 2. Si se tuviera que elegir entre el criterio 1 y el 3. Se puede concluir que el criterio 3 obtuvo un mejor desempeño de acuerdo a los p-valores obtenidos.

6. Anexos.

Se anexan los códigos obtenidos para cada cirterio utilizado.

6.1. Criterio 1.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>


#define RENYI_MAP(var, parametro, j) ((var)*(parametro)+((var)>>(j)))

#define  MAX 4294967295 /*El valor de 2^32-1.*/
#define TAMANIOCICLO 4294967296 /*EL valor de 2^32.*/
#define noMapas 4 /*NUmero de mapas a utilizar.*/
#define ITtotales 80000 /*Iteraciones totales para NIST.*/
```

```
/*
 File:   main.c
 Author: daniel
 El siguiente programa es un ejemplo de cuatro mapas caoticos RENYI acoplados.
 Se utiliza la suma para lograr el acoplado.
 Para declarar nuestras variables, utilizamos los siguientes componentes:
 -Un arreglo que guarda el valor de los parametros para cada mapa.
 -Un arreglo que guarda el valor de los valores calculados para cada mapa.
*/


int main(){


   /*Declaramos los arreglos que vamos a utilizar para guardar esto.*/
   unsigned long Xn[noMapas];
   unsigned long Xtotal[ITtotales];
   unsigned long parametros[noMapas];
   unsigned int i;
   int epsilon;
   FILE *  archivobin;


   unsigned long H=0;
   unsigned int j=9; /*No mas de 16.*/
   unsigned long iteraciones=0;
   unsigned long IT = 80000;


  /* Apertura del fichero de destino, para escritura en binario.*/
   archivobin = fopen ("binarioSUMA.dat", "wb");
   if (archivobin==NULL)
   {
   perror("No se puede abrir binarioSUMA.dat");
   return -1;
   }



   /*Inicializamos nuestros parametros, en este punto se aplica el uso
   de una llave, en este ejemplo todavia no elegimos una. Tambin,
```

```
los parametros son fijos en este ejemplo.*/
parametros[0]=131071;
Xn[0]=653;
parametros[1]=104729;
Xn[1]=769;
parametros[2]=524287;
Xn[2]=227;
parametros[3]=65537;
Xn[3]=823;

/*Primero, hacemos un ciclo inicial para calcular un nuevo valor para cada
uno de los mapas y calcular, por primera vez, el resultado de la operacion
XOR.*/
for( i =0; i<noMapas; i++){
    Xn[i]= RENYI_MAP(Xn[i],parametros[i],j);
    H+=Xn[i]%256;
}


/*Elegimos un epsilon aleatorio entre 1 y 16 (para operaciones de 32 bits).
En este caso, elegimos uno que este entre 0 y 15.*/
epsilon = 1;


unsigned int k;
unsigned long newH;

do {

     newH = 0;
     for(k=0;k<noMapas; k++){
         Xn[k]= RENYI_MAP(Xn[k],parametros[k],j) + epsilon*H;
         Xtotal[iteraciones++] = Xn[k];

         newH+=(Xn[k]%256);
     }
     H = newH;
     /*Ahora, elegimos un valor para epsilon entre 1 y 8 bits.*/
     epsilon = (H % 3) - 1;



} while (iteraciones < IT);
```

```
   /*Escribimos la informacion.*/
   fwrite(Xtotal,4,80000,archivobin);

   if(!fclose(archivobin)){
      printf( "\nArchivo binario cerrado\n" );
   }
   else{
      printf( "\nError: Archivo binario no cerrado \n" );
      return 1;
   }



return 0;
}
```

6.2. Criterio 2.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>


#define RENYI_MAP(var, parametro, j) ((var)*(parametro)+((var)>>(j)))

#define  MAX 4294967295 /*El valor de 2^32-1.*/
#define TAMANIOCICLO 4294967296 /*EL valor de 2^32.*/
#define noMapas 4 /*NUmero de mapas a utilizar.*/
#define ITtotales 80000 /*Iteraciones totales para NIST.*/



/*
 File:   main.c
 Author: daniel
 El siguiente programa es un ejemplo de cuatro mapas caoticos RENYI acoplados.
 Se utiliza la suma para lograr el acoplado.
 Para declarar nuestras variables, utilizamos los siguientes componentes:
 -Un arreglo que guarda el valor de los parametros para cada mapa.
 -Un arreglo que guarda el valor de los valores calculados para cada mapa.
*/
```

```
int main(){


   /*Declaramos los arreglos que vamos a utilizar para guardar esto.*/
   unsigned long Xn[noMapas];
   unsigned long Xtotal[ITtotales];
   unsigned long parametros[noMapas];
   unsigned int i;
   int epsilon;
   FILE *  archivobin;


   unsigned long H=0;
   unsigned int j=9; /*No mas de 16.*/
   unsigned long iteraciones=0;
   unsigned long IT = 80000;

   /*Apertura del fichero de destino, para escritura en binario.*/
   archivobin = fopen ("binarioXOR.dat", "wb");
   if (archivobin==NULL)
   {
   perror("No se puede abrir binarioXOR.dat");
   return -1;
   }


   /*Inicializamos nuestros parametros, en este punto se aplica el uso
   de una llave, en este ejemplo todavia no elegimos una. Tambin,
   los parametros son fijos en este ejemplo.*/
   parametros[0]=131071;
   Xn[0]=653;
   parametros[1]=104729;
   Xn[1]=769;
   parametros[2]=524287;
   Xn[2]=227;
   parametros[3]=65537;
   Xn[3]=823;

   /*Primero, hacemos un ciclo inicial para calcular un nuevo valor para cada
   uno de los mapas y calcular, por primera vez, el resultado de la operacion
   XOR.*/
   for( i =0; i<noMapas; i++){
```

```
    Xn[i]= RENYI_MAP(Xn[i],parametros[i],j);
    H^=Xn[i];
}


/*Elegimos un epsilon aleatorio entre 1 y 16 (para operaciones de 32 bits).
En este caso, elegimos uno que este entre 0 y 15.*/
epsilon = 1;


unsigned int k;
unsigned long newH;

do {

     newH = 0;
     for(k=0;k<noMapas; k++){
         Xn[k]= RENYI_MAP(Xn[k],parametros[k],j) + epsilon*H;
         Xtotal[iteraciones++] = Xn[k];

         newH^=Xn[k];
     }
     H = newH;
     /*Ahora, elegimos un valor para epsilon entre 1 y 8 bits.*/
     epsilon = (H % 3) - 1;



} while (iteraciones < IT);

/*Escribimos la informacion.*/
fwrite(Xtotal,4,80000,archivobin);

if(!fclose(archivobin)){
   printf( "\nArchivo binario cerrado\n" );
}
else{
   printf( "\nError: Archivo binario no cerrado \n" );
   return 1;
}
```

```
return 0;
}
```

6.3. Criterio 3.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>


#define RENYI_MAP(var, parametro, j) ((var)*(parametro)+((var)>>(j)))

#define  MAX 4294967295 /*El valor de 2^32-1.*/
#define TAMANIOCICLO 4294967296 /*EL valor de 2^32.*/
#define noMapas 4 /*NUmero de mapas a utilizar.*/
#define ITtotales 80000 /*Iteraciones totales para NIST.*/



/*
 File:   main.c
 Author: daniel
 El siguiente programa es un ejemplo de cuatro mapas caoticos RENYI acoplados.
 Para declarar nuestras variables, utilizamos los siguientes componentes:
 -Un arreglo que guarda el valor de los parametros para cada mapa.
 -Un arreglo que guarda el valor de los valores calculados para cada mapa.
*/


int main(){


   /*Declaramos los arreglos que vamos a utilizar para guardar esto.*/
   unsigned long Xn[noMapas];
   unsigned long Xtotal[ITtotales];
   unsigned long parametros[noMapas];
   unsigned int i;
   unsigned int epsilon;
   unsigned long gamma;
   unsigned long gammaComp;
   FILE *  archivobin;

   unsigned long H=0;
   unsigned int j=9; /*No mas de 16.*/
```

```
 unsigned long iteraciones=0;
 unsigned long IT = 80000;

/* Apertura del fichero de destino, para escritura en binario.*/
 archivobin = fopen ("binarioXORcomp.dat", "wb");
 if (archivobin==NULL)
 {
 perror("No se puede abrir binarioXORcomp.dat");
 return -1;
 }


 /*Inicializamos nuestros parametros, en este punto se aplica el uso
  *de una llave, en este ejemplo todavia no elegimos una. Tambien,
  *los parametros son fijos en este ejemplo.*/
 parametros[0]=131071;
 Xn[0]=653;
 parametros[1]=104729;
 Xn[1]=769;
 parametros[2]=524287;
 Xn[2]=227;
 parametros[3]=65537;
 Xn[3]=823;

 /*Primero, hacemos un ciclo inicial para calcular un nuevo valor para cada
 uno de los mapas y calcular, por primera vez, el resultado de la operacion
 XOR.*/
 for( i =0; i<noMapas; i++){
     Xn[i]= RENYI_MAP(Xn[i],parametros[i],j);
     H^=Xn[i];
 }


  /*Elegimos un epsilon aleatorio entre 1 y 16 (para operaciones de 32 bits).
    En este caso, elegimos uno que este entre 0 y 15.*/
 epsilon = 5;

 gamma= pow(2,epsilon);
 gamma-=1;
 gammaComp= MAX-gamma;

 unsigned int k;
 unsigned long newH;
```

```
   do {

        newH = 0;
        for(k=0;k<noMapas; k++){
            Xn[k]= gammaComp^RENYI_MAP(Xn[k],parametros[k],j)+ gamma^H;
            Xtotal[iteraciones++] = Xn[k];

            newH^=Xn[k];
        }
        H = newH;
        /*Ahora, elegimos un valor para epsilon entre 1 y 8 bits.*/
        epsilon = (H % 8) + 1;

   } while (iteraciones < IT);


   /*Escribimos la informacion.*/
   fwrite(Xtotal,4,80000,archivobin);

   if(!fclose(archivobin)){
      printf( "\nArchivo binario cerrado\n" );
   }
   else{
      printf( "\nError: Archivo binario no cerrado \n" );
      return 1;
   }


return 0;
}
```