FÍSICA COMPUTACIONAL I SERGIO SANZ HERNÁNDEZ

Curso 2017-2018

Segunda prueba de Evaluación Continua (2ª PEC)

Tema 15
Autómatas Celulares Elementales

Introducción

Los autómatas celulares son idealizaciones matemáticas de sistemas físicos en los que el espacio y el tiempo son magnitudes discretas. Consisten en un "enrejado" formado por celdas idénticas ordenadas uniformemente. Normalmente la extensión de esta rejilla es infinita, pero el estado que puede adquirir cada celda es discreto. El estado de un Autómata Celular (AC) en un determinado instante, viene especificado por los valores de cada celda concreta, y su evolución en el tiempo viene determinado por los valores de las celdas vecinas en el instante de tiempo previo, de acuerdo con reglas específicas.

En general cada celda puede tomar un número arbitrario de valores ($\boldsymbol{k}$), y permitir que el valor de un sitio concreto dependa de los valores de los sitios vecinos hasta una distancia ($\boldsymbol{r}$) dada. La función que describiría esta evolución, para cada celda en un momento ($\boldsymbol{t}$) concreto sería entonces: $\boldsymbol{a_i^{(t+1)} = F(a_{i-r}^{(t)}, \ldots, a_{i+r}^{(t)})}$.

Los AC así construidos generan un número muy alto de patrones diferentes, según la expresión $\boldsymbol{k^{k^{2r+1}}}$, incluso para valores de $\boldsymbol{k}$ y $\boldsymbol{r}$ pequeños, a pesar de lo cual presentar un tipo de comportamiento que pueden ser agrupados en sólo cuatro clases básicas:

- **Clase 1**: La evolución conduce a un estado homogéneo al cabo de pocos pasos, y por lo tanto su predicción es trivial, ya que, independientemente del estado inicial, el sistema evoluciona hacia un único estado homogéneo.
- **Clase 2**: La evolución conduce a un conjunto de estructuras estables o periódicas, que son separadas y sencillas, que se propagan a un número finito de sitios vecinos. En esta clase el cambio en el valor inicial de una sola celda afecta sólo a una región finita de celdas a su alrededor. Este comportamiento implica que la predicción del valor final de un sitio concreto requiere el conocimiento de sólo un conjunto finito de valores iniciales de celdas.
- **Clase 3**: La evolución conduce a un comportamiento caótico. En este caso el cambio en el valor inicial de una celda se propaga indefinidamente, y afecta a cada paso a celdas más alejadas. El valor de una celda concreta, en esta clase, después de muchos pasos depende por tanto de un número creciente de celdas iniciales. Si el estado inicial es aleatorio, esta dependencia puede conducir a una sucesión aparentemente caótica de valores para una determinada celda.
- **Clase 4**: La evolución conduce a estructuras complejas, de duración variable en el tiempo, y estas estructuras destacan por un, incluso, mayor grado de impredecibilidad.

Los AC de Clase 2 pueden ser considerados como "filtros" que seleccionan determinadas características del estado inicial, y se pueden construir para que secuencias iniciales de un determinado tipo sobrevivan y el resto no, por lo que tienen importancia práctica en el **procesado digital de imágenes.**

Los AC de Clase 3 exhiben un comportamiento caótico aperiódico, sin embargo, los patrones generados no son completamente aleatorios ya que presentan propiedades estadísticas con un comportamiento autoorganizativo. Y su comportamiento a largo plazo viene determinado por esas propiedades estadísticas comunes.

En este tema nos vamos a circunscribir a un caso particular en el que cada celda sólo puede tomar dos valores diferentes (0 o 1), y se ve afectada sólo por los valores de las celdas inmediatamente vecinas ($\boldsymbol{r = 1}$), por lo que el valor de cada sitio en un momento dado ($\boldsymbol{t}$)viene representado por la expresión: $\boldsymbol{a_i^{(t+1)} = a_{i-1}^{(t)} + a_{i+1}^{(t)}\ mod\ 2}$, donde por "suma módulo 2" representamos el Booleano XOR.

Este modelo con reglas locales de vecindad unidimensional que viene definido por el valor de sólo tres celdas vecinas y sólo puede tomar los valores 0 y 1 es lo que vamos a llamar **Autómatas Celulares Elementales** (**ACE)**, y es su comportamiento el que vamos a estudiar con más detalle mediante los ejercicios propuestos en el tema.

Puesto que sólo se van a tener en cuenta tres células antecesoras para obtener el valor de la siguiente, eso hace que sólo haya 8 posibilidades de ordenación de los tres dígitos, y puesto que a cada ordenación la podemos asignar un resultado también binario, eso da un resultado de $\boldsymbol{2^8 = 256}$ posibilidades distintas de ACEs en una dimensión con una vecindad de tres celdas. Normalmente se imponen dos restricciones (no esenciales) a estas 256 reglas: La primera es que un ACE será considerado "ilegal" a no ser que un estado inicial nulo compuesto sólo por ceros permanezca inalterado. Esto "prohíbe" las reglas cuya expresión binaria termine por 1. La segunda restricción es que las reglas que sean simétricas (en formato binario) deben dar los mismos resultados (por ejemplo, la 100 y 001, o la 110 y la 011). Estas dos restricciones se requieren en muchas aplicaciones para impedir la propagación instantánea de los sitios con valor 1, y garantizan la isotropía y homogeneidad en la evolución de los ACEs. De esta forma quedan sólo 32 posibles ACEs "legales", dentro de los cuales se encuentran los que vamos a estudiar más en profundidad en los siguientes apartados.

Estos ACEs presentan dos características particulares, una de tipo local, que es **crecimiento de patrones fractales**, fácilmente observables en los ACEs generados a partir de una única celda inicial con valor 1, que se caracteriza por la autosimilitud de las partes con respecto al todo, y otra de tipo global, que es la **autoorganización**, en la que aparecen espontáneamente patrones, incluso a pesar de que el estado inicial no tenga ninguna estructura (o sea aleatorio), y que se estudiarán mediante el cálculo de la "distancia de Hamming" más adelante.

Entre las 32 reglas consideradas "legales" se observan, al menos, tres tipos de comportamientos generales: En el primer caso, el 1 inicial es borrado inmediatamente (reglas 0 y 160), o mantenido inalterado permanentemente (reglas 4 y 36); en una segunda clase (reglas 50 y 122) el valor inicial se copia a ambos lados y genera una estructura que se expande de uno en uno en cada dirección a cada escalón temporal. Estas dos clases son conocidas como "simples". Queda una tercera clase llamada "compleja" (por ejemplo, la regla 90), que producen patrones no triviales, pero que evolucionan hacia un conjunto de configuraciones que sobreviven con probabilidad no nula, considerados "atractores" de la evolución. Hay otras reglas con comportamientos particulares, como la 204, conocida como la "regla identidad", ya que propaga indefinidamente cualquier configuración inicial.

Ejercicio 15.1

A – OBJETIVOS.

Mostar las imágenes generadas por los listados proporcionados, que generan un ACE de 1000 celdas, con una sola celda inicial con valor 1 (la central), llamada "semilla", y observar su

comportamiento durante 500 pasos, de acuerdo a los criterios de clasificación explicados en la introducción, de una manera práctica observable mediante imágenes.

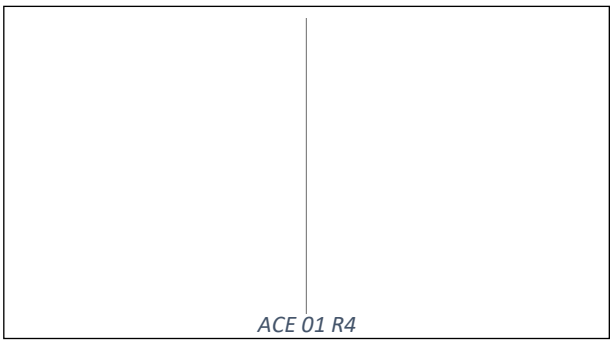
B – METODOLOGÍA.

Para crear las imágenes usamos tres listados en C. El primero simula la evolución de un ACE mediante una Regla elegida, con todas las celdas de la primera línea ceros, excepto la del medio (semilla), un 1. Este primer listado genera un resultado que es guardado en forma de imagen en escala de grises por una función, que en este caso se ha situado en un listado aparte, en forma de librería, para poder ser usado, si fuera necesario por otros programas. Y un tercer listado que sólo contiene las declaraciones de las funciones escritas en el segundo listado, para poder ser incorporado como un archivo de cabecera en el programa principal.

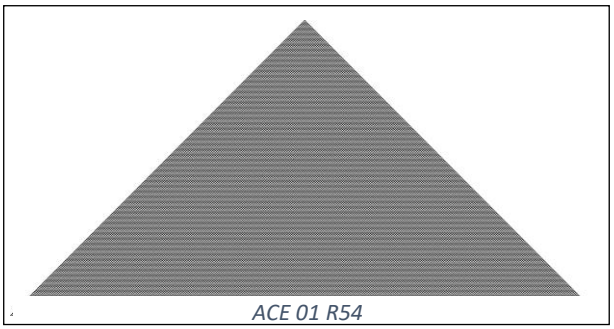
Los programas utilizados para este ejercicio son **EJERCICIO_15_1.c** (programa principal), **libguardaimagen.c** (programa con las funciones de guardado del ACE en forma gráfica) y **libguardaimagen.h** (archivo de cabecera para incluir en el programa principal y poder usar las funciones de la librería).

C – RESULTADOS.

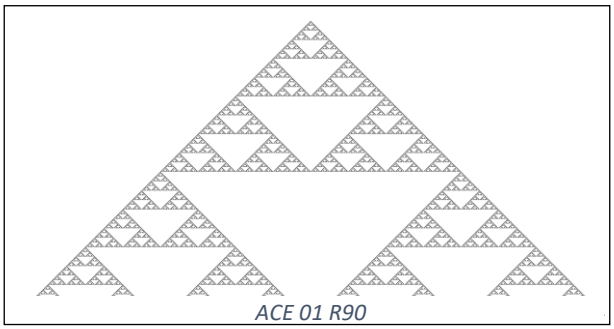
Obtenemos las siguientes imágenes, correspondientes a las reglas que figuran al pie de cada una, que son las pedidas en el ejercicio.



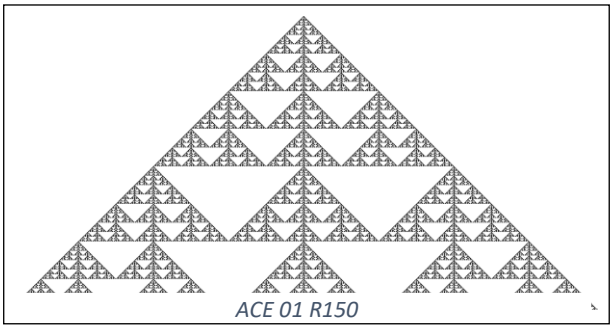
ACE 01 R4



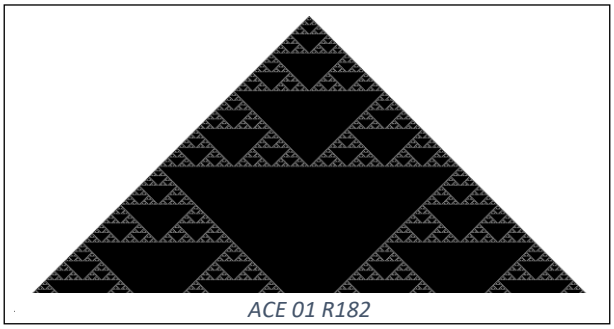
ACE 01 R54



ACE 01 R90



ACE 01 R150



ACE 01 R182

D – DISCUSIÓN.

Lo primero que se observa es que los cinco ACEs se encuentran dentro de la clase que denominamos "***legales***", ya que corresponden a los binarios 00000100, 00110110, 01011010, 10010110 y 10110110, respectivamente 4, 54, 90, 150 y 182, que terminan todos en 0, y que tienen las propiedades de isotropía y homogeneidad, como se observa, presentando todas ellas (excepto la 4 que es del grupo de las ***simples***, por lo que tiene un comportamiento trivial, como se puede ver, alcanzando una configuración estacionaria en el tiempo), un comportamiento no trivial, con las propiedades de aparición de patrones fractales, de autosimilaridad, como corresponde al grupo de reglas ***complejas*** al que se observa que pertenecen.

Ejercicio 15.2

A – OBJETIVOS.

Simulación de la evolución de ACEs compuestos por 500 celdas a partir de una primera fila de ceros y unos repartidos aleatoriamente y observar su comportamiento durante los primeros 500 pasos de evolución, conforme a la clasificación proporcionada en la introducción.

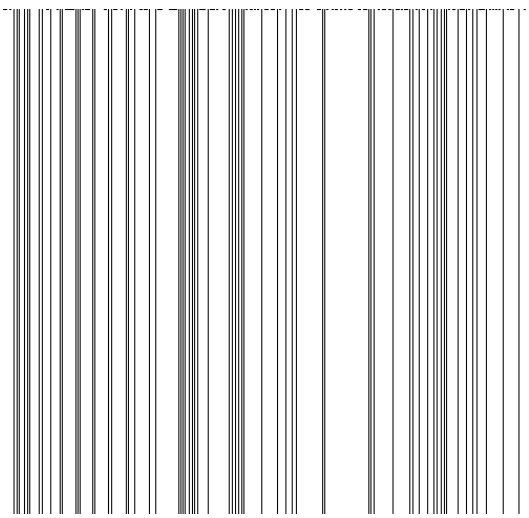
B – METODOLOGÍA.

Es análoga a la usada para el ejercicio anterior, sólo que el primer paso del ACE no está formado por una única celda central con valor 1, sino por una sucesión de ceros y unos distribuídos aleatoriamente a lo ancho del tamaño del ACE, que en este caso será de 500 celdas, y el código, usando este primer paso simulará la evolución durante 500 pasos, guardando el resultado nuevamente como una imagen, mediante los mismos programas que hemos usado en el ejercicio anterior para esta tarea.

Los programas utilizados para este ejercicio son **EJERCICIO_15_2.c** (programa principal), **libguardaimagen.c** (programa con las funciones de guardado del ACE en forma gráfica) y **libguardaimagen.h** (archivo de cabecera para incluir en el programa principal y poder usar las funciones de la librería).

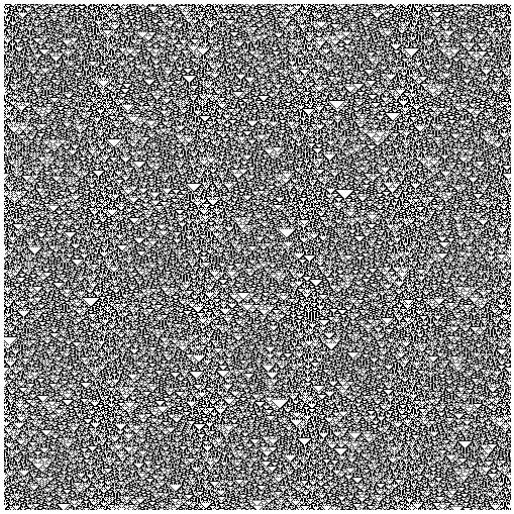
C – RESULTADOS.

Obtenemos las siguientes imágenes, correspondientes a las reglas que figuran al pie de cada una, que son las pedidas en el ejercicio.

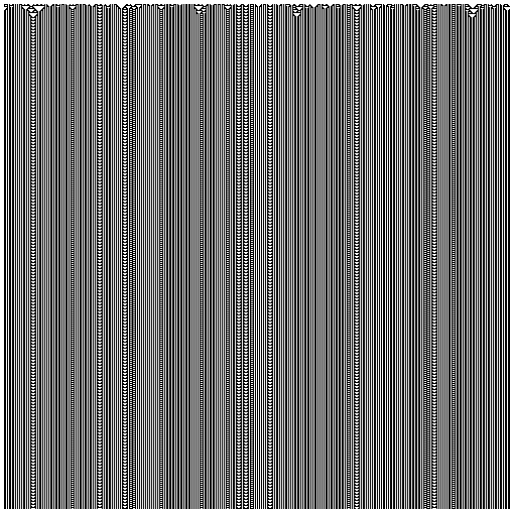


ACE 02 R4

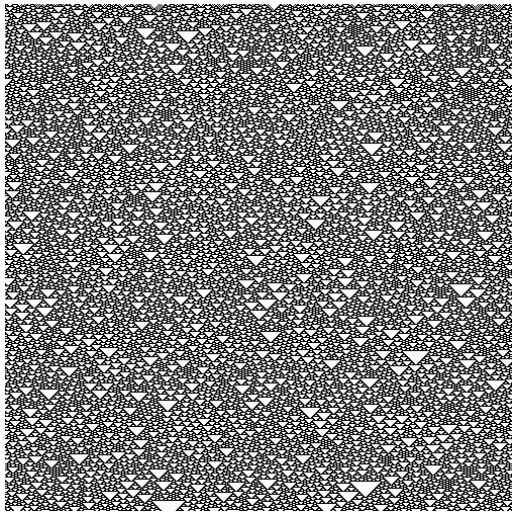
ACE 02 R32



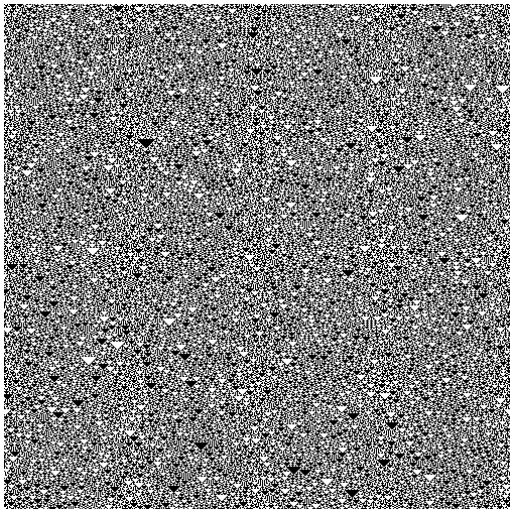
ACE 02 R90



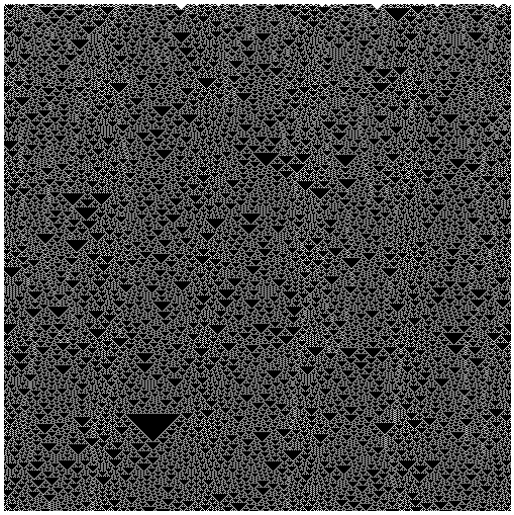
ACE 02 R94



ACE 02 R122



ACE 02 R150



ACE 02 R182

D – DISCUSIÓN.

Como consecuencia de la representación gráfica de los ACEs pedidos podemos reconocer diferentes tipos de comportamientos, correspondientes a los tipos de reglas mencionados en la introducción. Las reglas 4, 00000100; 32, 00100000; 90, 01011010; 94, 01011110; 122, 01111010; 150, 10010110 y 182, 10110110, usadas para estas simulaciones vuelven a ser de las consideradas "***legales***", por lo que muestran las características que corresponden a este tipo de reglas, que son la formación de patrones de crecimiento, y la autoorganización, como se ve por la formación espontánea de patrones triangulares, a pesar de partir de un inicio desorganizado aleatorio.

Además, se pueden clasificar dentro de éstas entre ***simples***, que serían las reglas 4, 32 y 94, ya que muestran un comportamiento trivial, pues las dos primeras alcanzan en los primeros pasos un estado final que no varía con el tiempo, la 4 autorepetitivo y la 32 se extingue, mientras que la 94 llega también rápidamente a un comportamiento periódico o ciclo límite, en el que los estados del ACE se repiten periódicamente. Por el contrario, el resto de las reglas representadas, 90, 122, 150 y 182, son reglas ***complejas***, que dan lugar a un comportamiento no trivial, con un comportamiento caótico, pero autoorganizativo, con la aparición de triángulos de todas las escalas. En este caso de las reglas complejas las figuras ilustran el hecho destacable de que su evolución temporal destruye la independencia de las celdas iniciales, y genera correlaciones entre los valores de sitios separados. De hecho, las bases de los triángulos que se observan son fluctuaciones en las que una secuencia de muchas celdas adyacentes tienen los mismos valores. El comportamiento de estos ACEs complejos puede ser caracterizado en analogía al comportamiento de los ***sistemas dinámicos***.

Ejercicio 15.3

A – OBJETIVO.

Se trata de mostrar el funcionamiento de la generación de números aleatorios en programación mediante lenguaje C.

B – METODOLOGÍA.

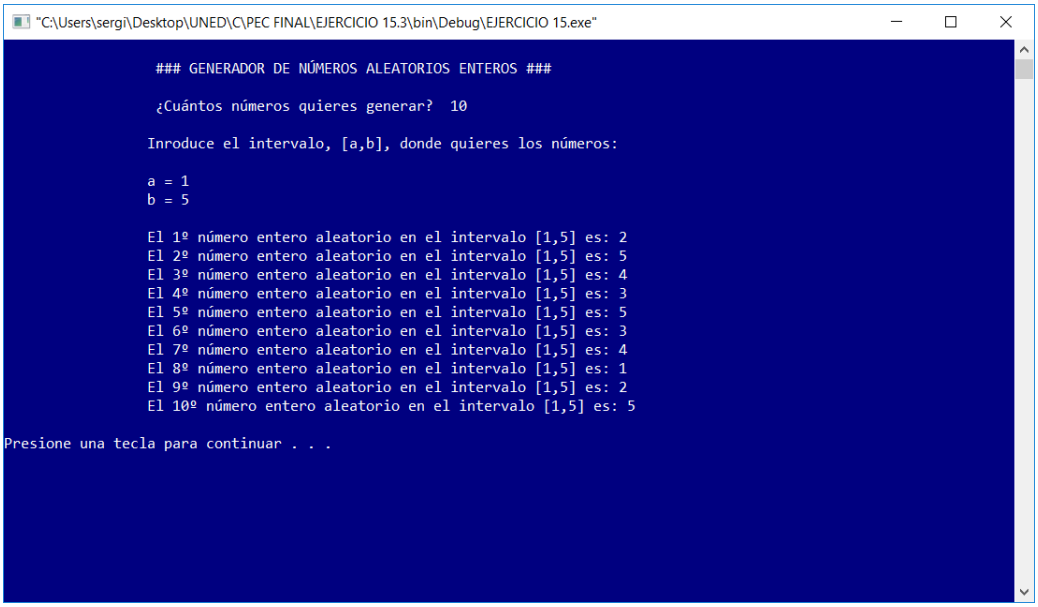
Se trata de la construcción de un código que genere un número entero pseudo aleatorio, y que lo sitúe en un intervalo dado.

Para ello he construido un pequeño programa que puede generar varios números aleatorios enteros (cuya cantidad tiene que ser introducida cuando se ejecuta el programa), en un intervalo cerrado [a, b], cuyos extremos también nos los pide el programa al ejecutarse, y nos imprime los números generados por pantalla a continuación.

El programa que efectúa esta tarea es el archivo **EJERCICIO_15_3.c** , y en el propio listado están explicadas en forma de comentarios las órdenes para realizar este cometido.

C – RESULTADOS.

Probando con diferentes cantidades de números y diferentes intervalos se comprueba que el código genera números sin ningún patrón (es decir, aleatorios), que todos ellos se sitúan dentro del intervalo deseado, incluyendo ambos extremos del intervalo, como se podría comprobar fácilmente forzando a que el programa sitúe por ejemplo cinco números en un intervalo formado por dos extremos que sean números enteros consecutivos.



D – DISCUSIÓN.

Mediante este sencillo programa se ponen de manifiesto las características de generación de números aleatorios en C, que, mediante el comando srand(time(0)); hace que el programa use como semilla para el generador de números el reloj de la CPU, y mediante la fórmula (double)rand()/(RAND_MAX+1.0); nos genera un número pseudoaleatorio en el intervalo [0, 1), al dividir cualquier número generado por el máximo que puede generar, que es $2^n - 1$, con $n = nº\ de\ bits$ usados para los enteros.

Ejercicio 15.4

A – OBJETIVO.

Calcular y representar, mediante un programa que simule la evolución durante 500 pasos de dos ACEs de 1000 celdas, que difieren en el estado inicial aleatorio en el estado de una sola celda central, simultáneamente, las diferencias entre los estados de los dos ACEs a cada paso, guardando estos datos en un archivo para poder representarlo con posterioridad con un programa que nos permita hacerlo, como Gnuplot.

Además de realizar las gráficas de las distancias de Hamming para cada Regla pedida, teniendo en cuenta que la evolución temporal de las distancias es proporcional a una ley de potencias ($H_t \sim t^{\alpha}$), vamos a calcular el "exponente de Hamming" (α), para las reglas que sea posible, que son las **reglas complejas.** tomando logaritmos y ajustando la recta resultante $log\ H_t = a + \alpha\ log\ t$, ya que la pendiente de la misma corresponderá con dicho exponente, mediante el comando "fit" de Gnuplot que realiza el ajuste por mínimos cuadrados a partir de los datos guardados en el archivo "haming.dat" con la instrucción:

fit a+b*x "haming.dat" u (log($1)): (log($2)) via a, b ,

dibujando posteriormente el ajuste y los puntos en la misma gráfica mediante el comando:

plot a+b*x, "haming.dat" u (log($1)): (log($2)) pt 7 ps 0.5 ,

para comprobar la bondad de cada ajuste, además de comprobar que el valor del exponente obtenido se ajusta con los valores esperados de 0,59 o 1, que son los valores que deben resultar para reglas complejas (según la regla), ya que en el caso de las reglas simples, como las configuraciones evolucionan hacia estados triviales, la distancia se mantiene en valores bajos, o constante según la regla de que se trate.

B – METODOLOGÍA.

Este estudio se va a llevar a cabo mediante un programa que, a diferencia de los ejercicios anteriores, tiene que generar simultáneamente dos ACEs de 1000 celdas, a partir de un estado inicial aleatorio igual para ambos, en el que se introduce una orden para que la celda del medio de la primera fila se haga distinta entre los dos antes de iniciar la construcción del resto de los dos ACEs.

A continuación, se simulará el resto de los dos ACEs al mismo tiempo, comprobando mediante el operador booleano XOR cada celda que se va generando, y sumando el número de estados diferentes en cada paso de tiempo, para irlos guardando cada vez que se termine de generar un escalón temporal en un archivo, que llamamos "haming.dat", y que almacenará el número de diferencias (parejas de datos $(t, H_t(s_1, s_2))$ en cada paso de evolución de los ACEs.

El listado que realiza estas tareas corresponde al archivo **EJERCICIO_15_4.c**

Una vez terminada la simulación de los 500 pasos quedarán los datos de las distancias de haming guardadas en ese archivo, que se podrá representar mediante Gnuplot, con el comando

plot "haming.dat"with lines ,

para observar el comportamiento de la diferencia de la evolución de las dos simulaciones.

Para calcular el exponente de Hamming de las reglas complejas, tomamos logaritmos de la ley de potencias que aproxima su comportamiento y ajustando la recta resultante

$$\boldsymbol{log\ H_t = a + \alpha\ log\ t}\ ,$$

ya que la pendiente de la misma corresponderá con dicho exponente, mediante el comando "fit" de Gnuplot que realiza el ajuste por mínimos cuadrados a partir de los datos guardados en el archivo "haming.dat" con la instrucción:

fit a+b*x "haming.dat" u (log($1)): (log($2)) via a, b ,

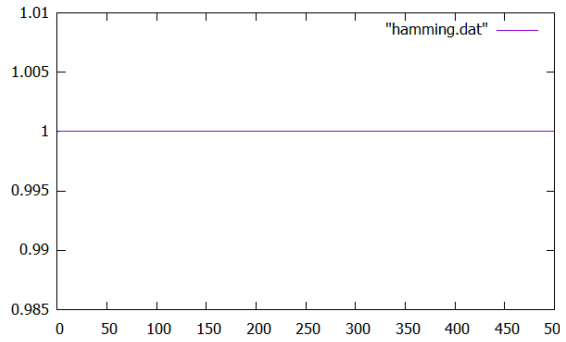
dibujando posteriormente el ajuste y los puntos en la misma gráfica mediante el comando:

plot a+b*x, "haming.dat" u (log($1)): (log($2)) pt 7 ps 0.5 ,

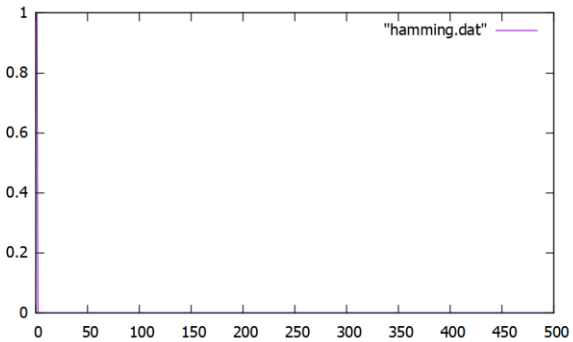
para comprobar la bondad de cada ajuste, además de comprobar que el valor del exponente obtenido se ajusta con los valores esperados de 0,59 o 1, que son los valores que deben resultar para reglas complejas (según la regla), ya que, en el caso de las reglas simples, como las configuraciones evolucionan hacia estados triviales, la distancia se mantiene en valores bajos, o constante según la regla de que se trate.

C – RESULTADOS.

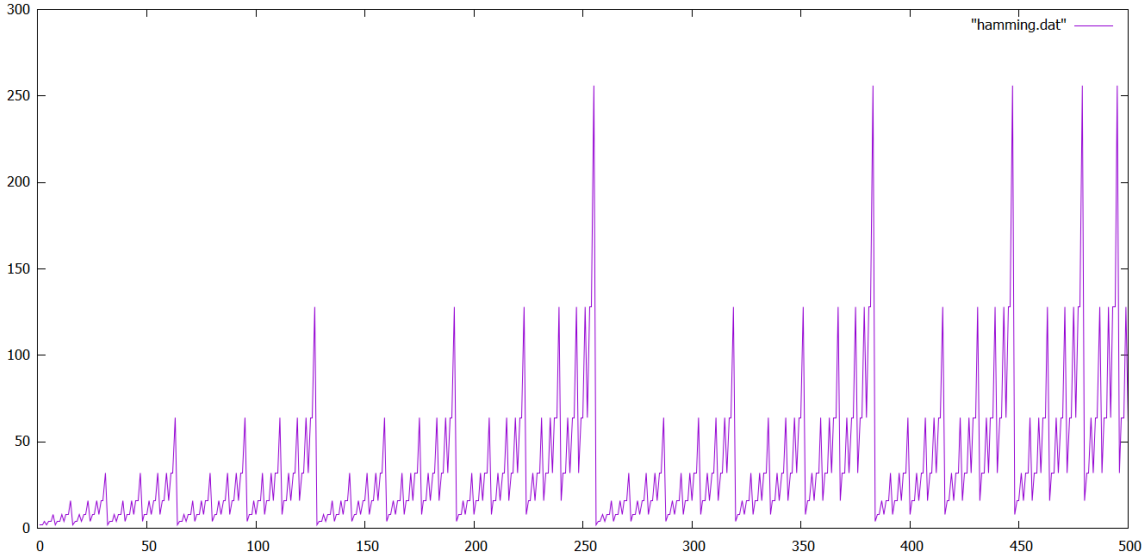
El primer grupo de resultados que obtenemos son las gráficas de las ***distancias de Hamming*** para las reglas 4, 32, 90, 94, 122, 150 y 182, que son las que se nos pide:



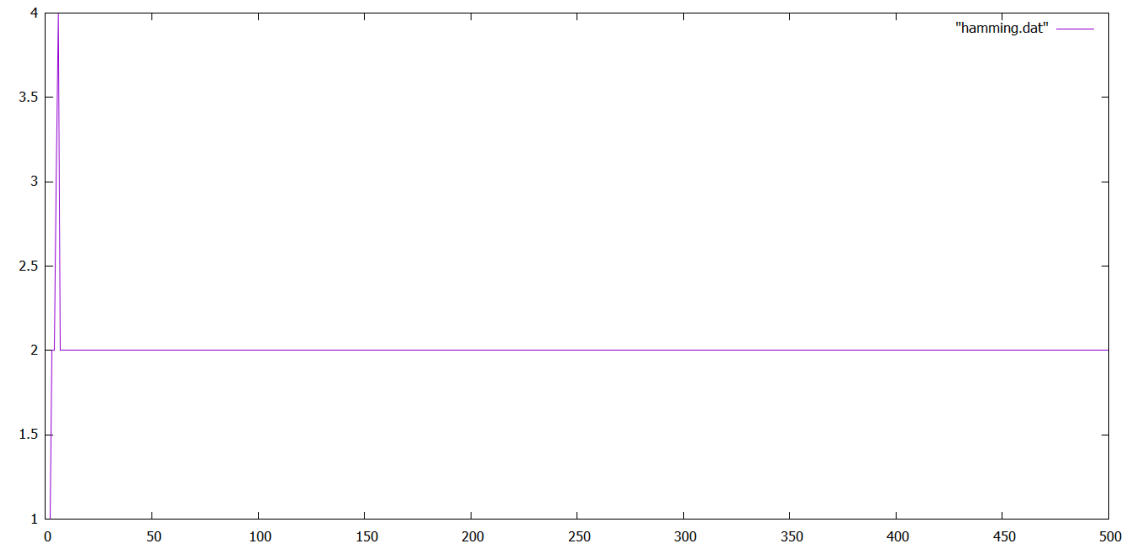
Dist. Hamming R4



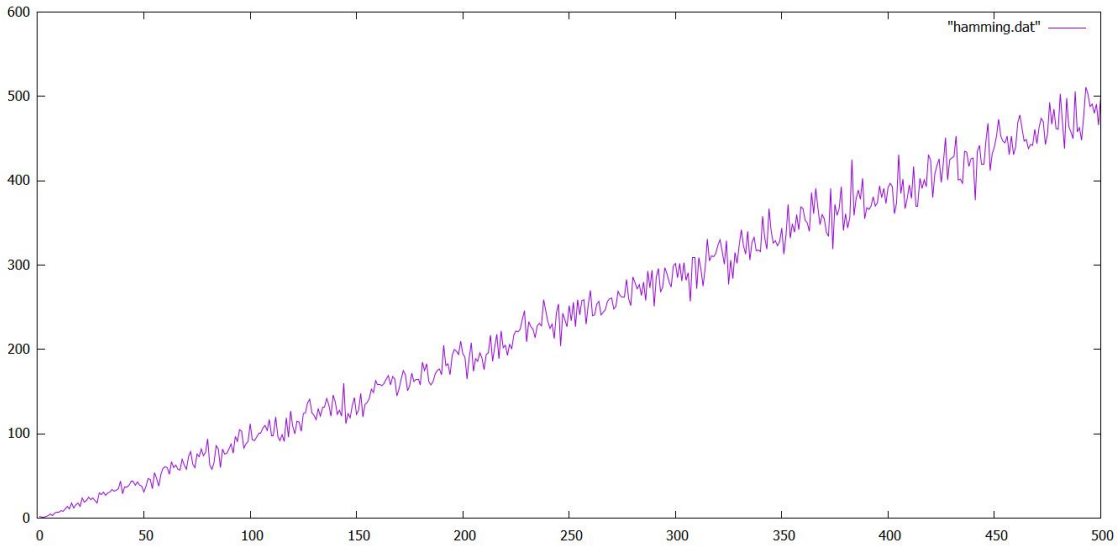
Dist. Hamming R32



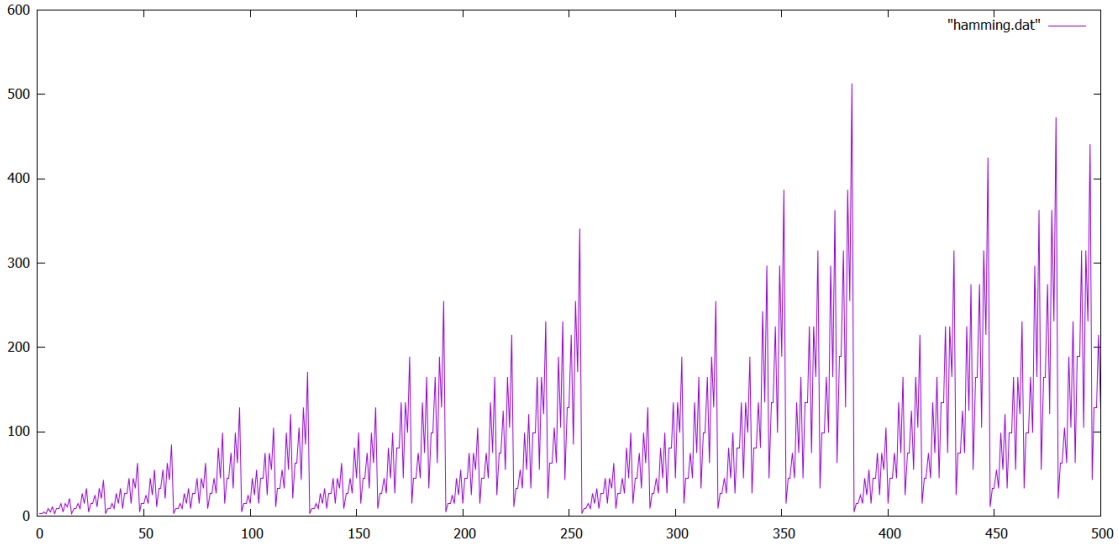
Dist. Hamming R90



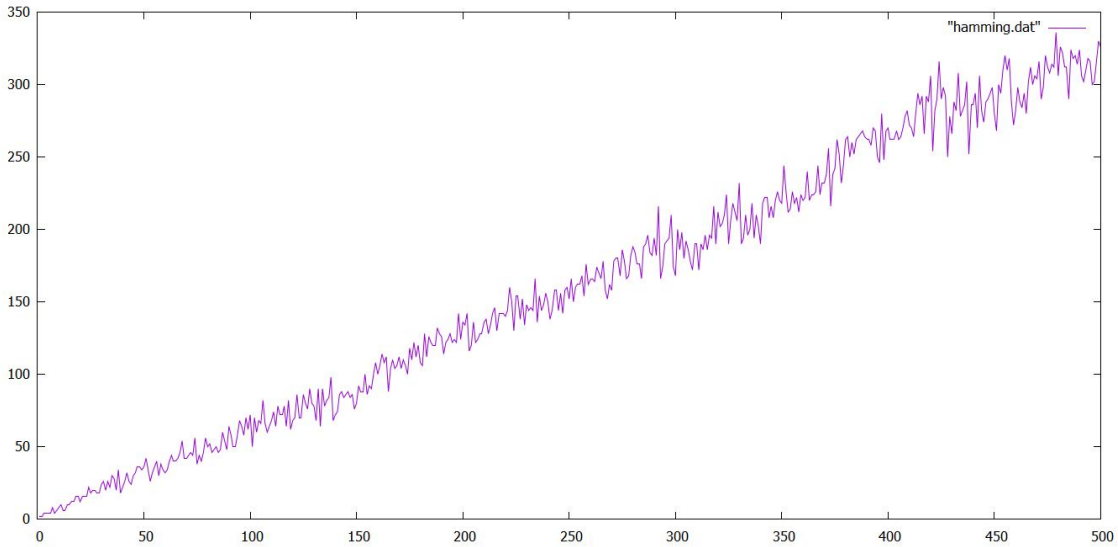
Dist. Hamming R94



Dist. Hamming R122

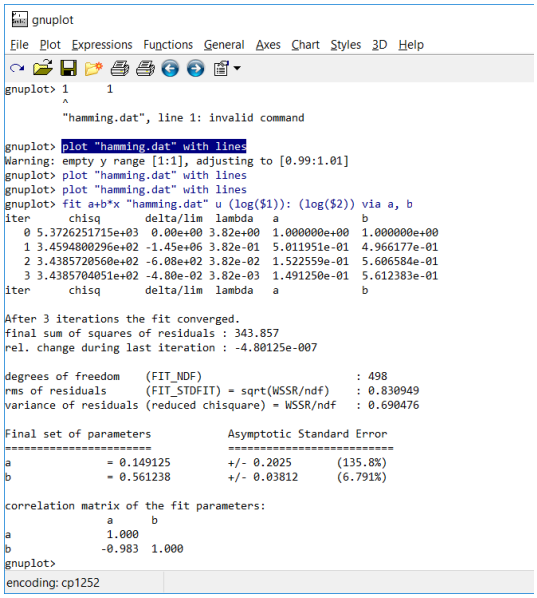


Dist. Hamming R150



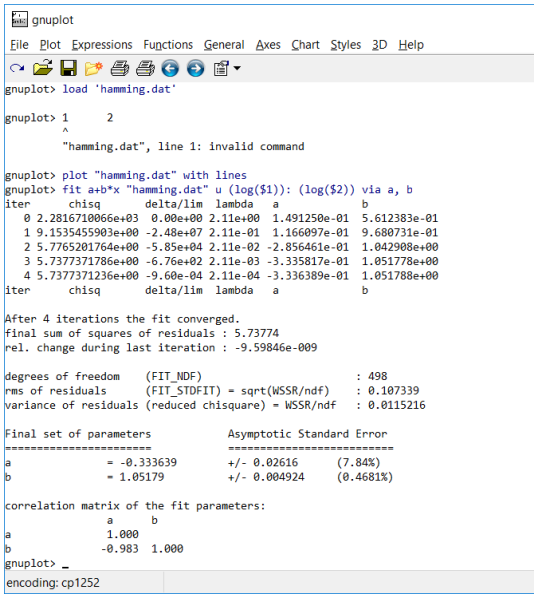
Dist. Hamming R182

El siguiente grupo de resultados son los ajustes obtenidos por Gnuplot de las reglas complejas, de donde el valor de ***b*** (pendiente de la recta) es el valor del ***exponente de Hamming***:



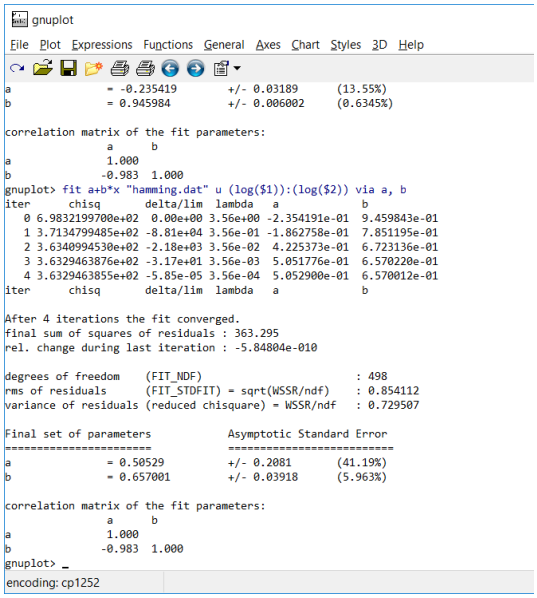
```
gnuplot
File Plot Expressions Functions General Axes Chart Styles 3D Help

gnuplot> 1      1
         ^
        "hamming.dat", line 1: invalid command

gnuplot> plot "hamming.dat" with lines
Warning: empty y range [1:1], adjusting to [0.99:1.01]
gnuplot> plot "hamming.dat" with lines
gnuplot> plot "hamming.dat" with lines
gnuplot> fit a+b*x "hamming.dat" u (log($1)): (log($2)) via a, b
iter      chisq       delta/lim  lambda   a             b
   0 5.3726251715e+03  0.00e+00 3.82e+00  1.000000e+00  1.000000e+00
   1 3.4594800296e+02 -1.45e+06 3.82e-01  5.011951e-01  4.966177e-01
   2 3.4385720560e+02 -6.08e+02 3.82e-02  1.522559e-01  5.606584e-01
   3 3.4385704051e+02 -4.80e-02 3.82e-03  1.491250e-01  5.612383e-01
iter      chisq       delta/lim  lambda   a             b

After 3 iterations the fit converged.
final sum of squares of residuals : 343.857
rel. change during last iteration : -4.80125e-007

degrees of freedom    (FIT_NDF)                        : 498
rms of residuals      (FIT_STDFIT) = sqrt(WSSR/ndf)    : 0.830949
variance of residuals (reduced chisquare) = WSSR/ndf   : 0.690476

Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a               = 0.149125         +/- 0.2025       (135.8%)
b               = 0.561238         +/- 0.03812      (6.791%)

correlation matrix of the fit parameters:
                a      b
a               1.000
b              -0.983  1.000
gnuplot>
encoding: cp1252
```

Ajuste R90, b=0.56



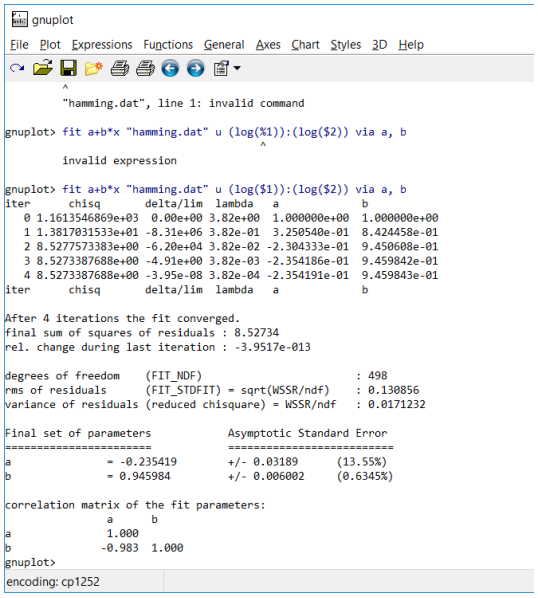
```
gnuplot
File Plot Expressions Functions General Axes Chart Styles 3D Help

gnuplot> load 'hamming.dat'

gnuplot> 1      2
         ^
        "hamming.dat", line 1: invalid command

gnuplot> plot "hamming.dat" with lines
gnuplot> fit a+b*x "hamming.dat" u (log($1)): (log($2)) via a, b
iter      chisq       delta/lim  lambda   a             b
   0 2.2816710066e+03  0.00e+00 2.11e+00  1.491250e-01  5.612383e-01
   1 9.1535455903e+00 -2.48e+07 2.11e-01  1.166097e-01  9.680731e-01
   2 5.7765201764e+00 -5.85e+04 2.11e-02 -2.856461e-01  1.042908e+00
   3 5.7377371786e+00 -6.76e+02 2.11e-03 -3.335817e-01  1.051778e+00
   4 5.7377371236e+00 -9.60e-04 2.11e-04 -3.336389e-01  1.051788e+00
iter      chisq       delta/lim  lambda   a             b

After 4 iterations the fit converged.
final sum of squares of residuals : 5.73774
rel. change during last iteration : -9.59846e-009

degrees of freedom    (FIT_NDF)                        : 498
rms of residuals      (FIT_STDFIT) = sqrt(WSSR/ndf)    : 0.107339
variance of residuals (reduced chisquare) = WSSR/ndf   : 0.0115216

Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a               = -0.333639        +/- 0.02616      (7.84%)
b               = 1.05179          +/- 0.004924     (0.4681%)

correlation matrix of the fit parameters:
                a      b
a               1.000
b              -0.983  1.000
gnuplot> _
encoding: cp1252
```

Ajuste R122, b=1.05



```
gnuplot
File Plot Expressions Functions General Axes Chart Styles 3D Help

a               = -0.235419        +/- 0.03189      (13.55%)
b               = 0.945984         +/- 0.006002     (0.6345%)

correlation matrix of the fit parameters:
                a      b
a               1.000
b              -0.983  1.000
gnuplot> fit a+b*x "hamming.dat" u (log($1)):(log($2)) via a, b
iter      chisq       delta/lim  lambda   a             b
   0 6.9832199700e+02  0.00e+00 3.56e+00 -2.354191e-01  9.459843e-01
   1 3.7134799485e+02 -8.81e+04 3.56e-01 -1.862758e-01  7.851195e-01
   2 3.6340994530e+02 -2.18e+03 3.56e-02  4.225373e-01  6.723136e-01
   3 3.6329463876e+02 -3.17e+01 3.56e-03  5.051776e-01  6.570220e-01
   4 3.6329463855e+02 -5.85e-05 3.56e-04  5.052900e-01  6.570012e-01
iter      chisq       delta/lim  lambda   a             b

After 4 iterations the fit converged.
final sum of squares of residuals : 363.295
rel. change during last iteration : -5.84804e-010

degrees of freedom    (FIT_NDF)                        : 498
rms of residuals      (FIT_STDFIT) = sqrt(WSSR/ndf)    : 0.854112
variance of residuals (reduced chisquare) = WSSR/ndf   : 0.729507

Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a               = 0.50529          +/- 0.2081       (41.19%)
b               = 0.657001         +/- 0.03918      (5.963%)

correlation matrix of the fit parameters:
                a      b
a               1.000
b              -0.983  1.000
gnuplot> _
encoding: cp1252
```

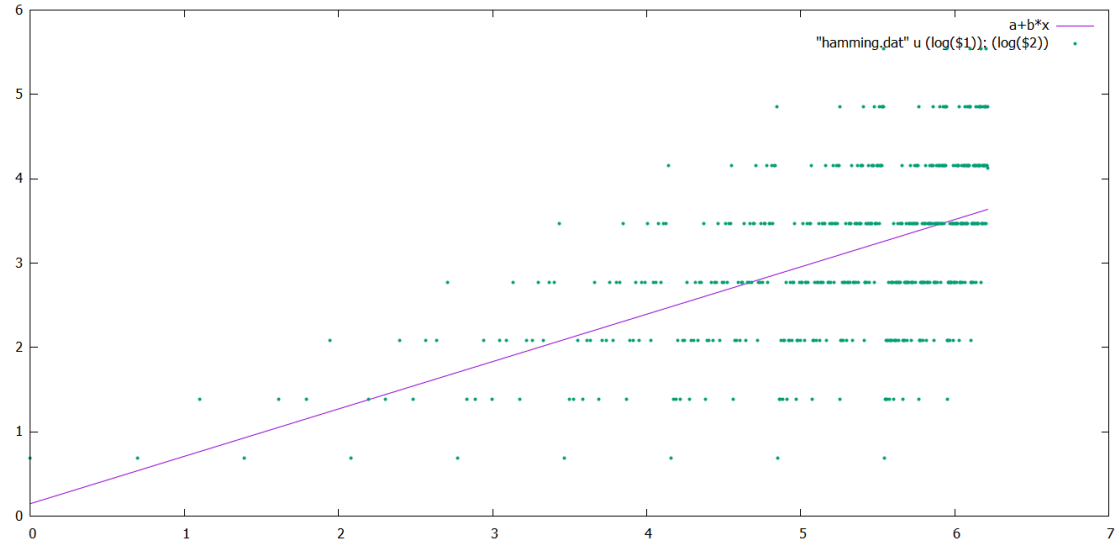
Ajuste R150, b=0.66



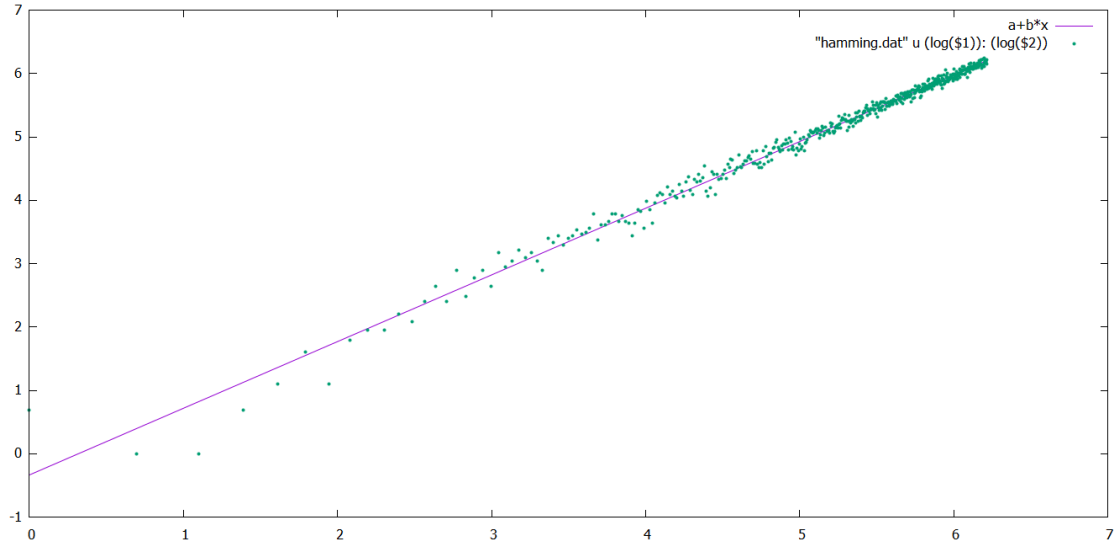
```
gnuplot
File Plot Expressions Functions General Axes Chart Styles 3D Help

         ^
        "hamming.dat", line 1: invalid command

gnuplot> fit a+b*x "hamming.dat" u (log(%1)):(log($2)) via a, b
                                    ^
        invalid expression

gnuplot> fit a+b*x "hamming.dat" u (log($1)):(log($2)) via a, b
iter      chisq       delta/lim  lambda   a             b
   0 1.1613546869e+03  0.00e+00 3.82e+00  1.000000e+00  1.000000e+00
   1 1.3817031533e+01 -8.31e+06 3.82e-01  3.250540e-01  8.424458e-01
   2 8.5277573383e+00 -6.20e+04 3.82e-02 -2.304333e-01  9.450608e-01
   3 8.5273387688e+00 -4.91e+00 3.82e-03 -2.354186e-01  9.459842e-01
   4 8.5273387688e+00 -3.95e-08 3.82e-04 -2.354191e-01  9.459843e-01
iter      chisq       delta/lim  lambda   a             b

After 4 iterations the fit converged.
final sum of squares of residuals : 8.52734
rel. change during last iteration : -3.9517e-013

degrees of freedom    (FIT_NDF)                        : 498
rms of residuals      (FIT_STDFIT) = sqrt(WSSR/ndf)    : 0.130856
variance of residuals (reduced chisquare) = WSSR/ndf   : 0.0171232

Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a               = -0.235419        +/- 0.03189      (13.55%)
b               = 0.945984         +/- 0.006002     (0.6345%)

correlation matrix of the fit parameters:
                a      b
a               1.000
b              -0.983  1.000
gnuplot>
encoding: cp1252
```

Ajuste R182, b=0.95

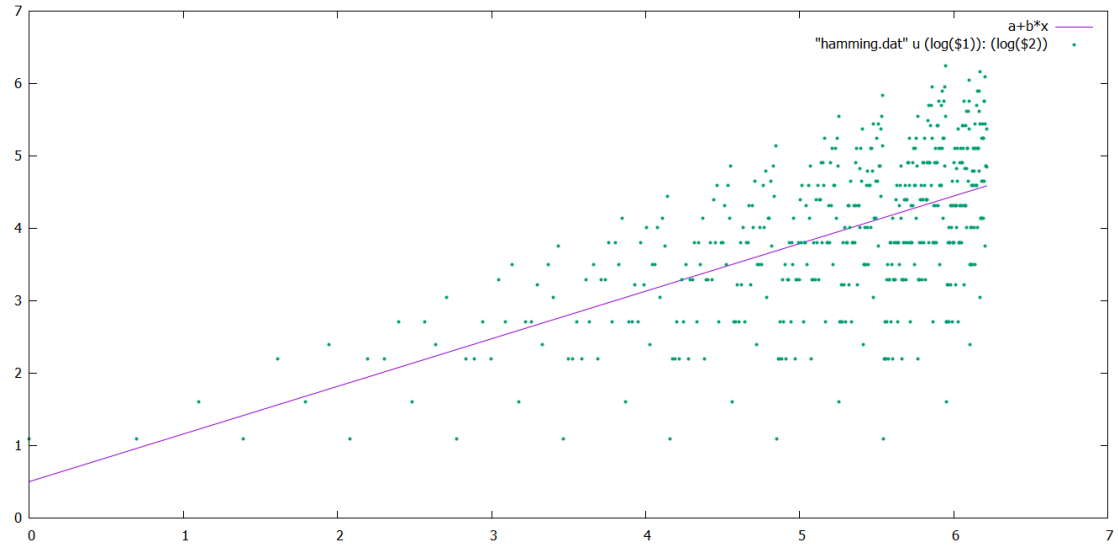
El último grupo de resultados de este ejercicio son las gráficas en las que se representa el ***ajuste de las rectas*** obtenidas junto con las distancias de Hamming a las que corresponden, para observar el grado de ajuste calculado:



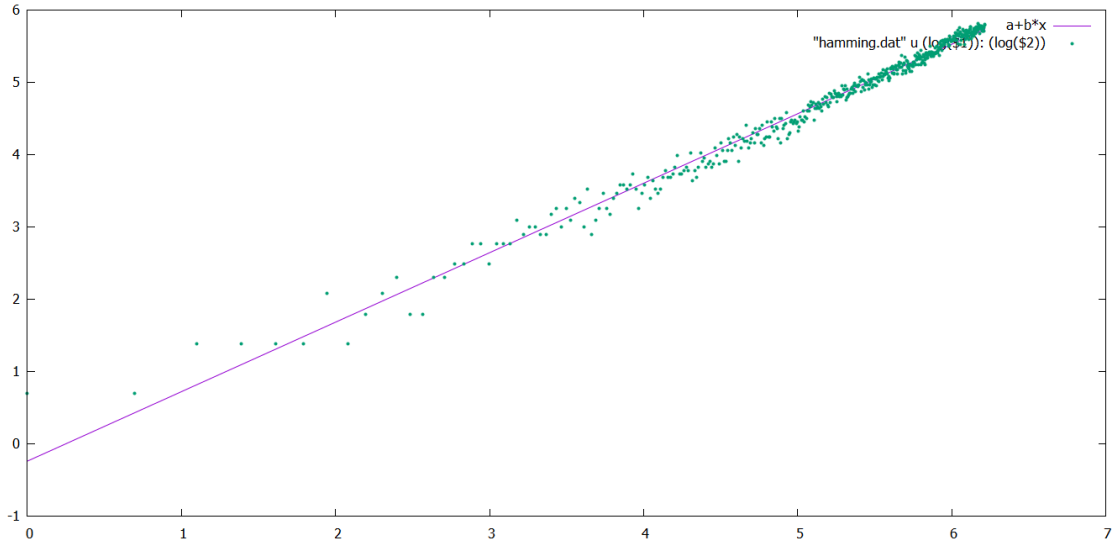
Recta de Ajuste R90



Recta de Ajuste R122



Recta de Ajuste R150



Recta de Ajuste R182

D – DISCUSIÓN.

En primer lugar, podemos analizar los resultados obtenidos de representar las **distancias de Hamming** para cada regla. Las gráficas se pueden agrupar en tres clases, que corresponden a los tipos de reglas que las generan. Las reglas, 4, 32 y 94 producen gráficas muy similares, ya que las tres llegan a una distancia constante tras un corto periodo inicial (o ni siquiera como en la regla 4 que siempre es 1, como corresponde con reglas simples, bien sea de ***punto límite***, como la 4 y la 32, o de ***ciclo límite***, como la 94, ya que al llegar todos los ACEs de esas reglas a una situación estacionaria a los pocos pasos, es lógico que las diferencias entre distintos ACEs de la misma regla se estabilicen. Las gráficas de las reglas 90 y 150, a pesar de que la distancia de Hamming crece, no lo hace constantemente, sino que muestran unos patrones de cierta periodicidad, en comparación con el crecimiento más uniforme de las reglas 122 y 182. Ambos tipos acaban divergiendo en el tiempo, como corresponde a un comportamiento de reglas complejas, pero se diferencian en que la reglas 90 y 150 corresponden a un subtipo de reglas complejas llamado ***reglas aditivas***, cuyo comportamiento se caracteriza por el hecho de que los patrones finales obtenidos al cabo de un tiempo, son simplemente la superposición de los patrones que se hubieran generado por cada celda inicial no nula aislada, y su regla en notación binaria es de la forma $\boldsymbol{\alpha_1\alpha_2\ 0\ \alpha_3\alpha_2\alpha_1\alpha_3\ 0}$, con $\boldsymbol{\alpha_3 = \alpha_1 \oplus \alpha_2}$, mientras el resto de las reglas complejas "legales", no cumplen esta condición, y se consideran ***reglas no aditivas***, como la 122 y la 182, y de ahí la diferencia con las gráficas de la distancia de Hamming, de las reglas 90 y 150.

Algo similar ocurre con el cálculo del ***exponente de Hamming***, mediante el ajuste con Gnuplot, aunque una diferencia es que en el caso de reglas simples (4, 32 y 94) no se puede hacer ningún ajuste. En cuanto a las reglas complejas también se aprecia diferencia entre los valores de los exponentes, ya que mientras en las reglas aditivas (90 y 150) está próximo al valor teórico de 0,59 (exactamente 0,56 y 0,66, respectivamente), en el caso de las no aditivas (122 y 182) están próximos al valor teórico de 1 (exactamente 1,05 y 0,95, respectivamente).

Por último, en los gráficos en los que se representa la recta de ajuste junto a los puntos ajustados se aprecia que, efectivamente, la pendiente de la recta de las reglas no aditivas (122 y 182) es superior, y las rectas se ajustan mucho mejor a la nube de puntos, ya que la evolución de la distancia de Hamming es más uniforme, que en el caso de las reglas 90 y 150.

Ejercicio 15.5

A – OBJETIVO.

Realizar el ajuste de la recta

$$log\ H_t = a + \alpha\ log\ t$$

para calcular el Coeficiente de Hamming **(α)** mediante un ajuste por mínimos cuadrados mediante un código en C, y comparar los resultados obtenidos por este método, con los obtenidos en el ejercicio anterior correspondiente al ajuste de dicha recta mediante Gnuplot.

B – METODOLOGÍA.

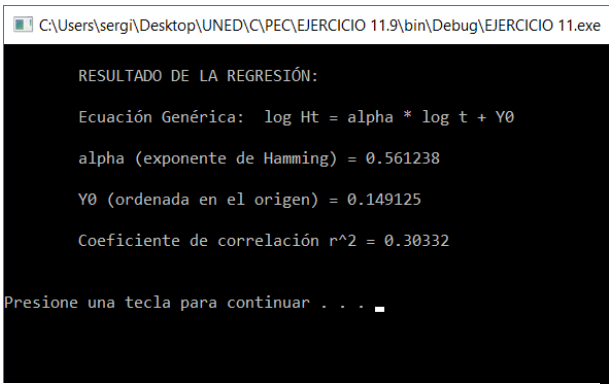
El procedimiento por el que se va a realizar esto, es mediante un código en C que realiza un ajuste por mínimos cuadrados de unos datos leídos de un archivo que contiene pares de coordenadas para ajustar.

En este caso particular el código va a ir integrado al final del programa del ejercicio anterior como **EJERCICIO_15_5.c**, y leerá los valores de la Distancia de Hamming directamente después de que el archivo haya sido creado y cerrado, y devolverá los resultados del ajuste como resultado del programa.

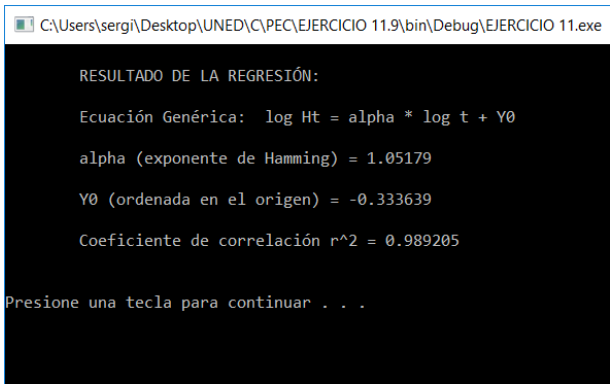
Como dato adicional hay que mencionar que los datos que lee el código de ajuste tienen que ser pasados a logaritmos, antes de ser introducidos en la fórmula de ajuste, para poder emplearlos como ajuste de una recta, según la ecuación mostrada más arriba.

C – RESULTADOS.

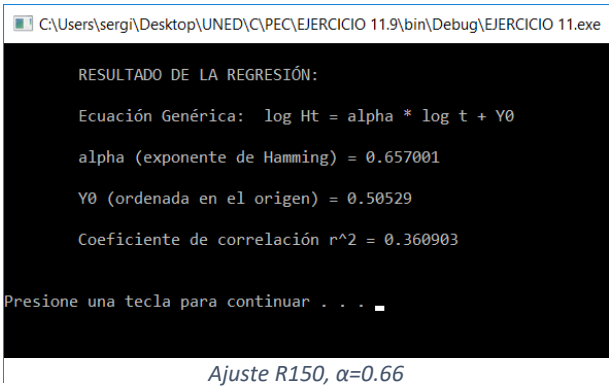
Mostramos las salidas del ajuste para las mismas reglas que lo admitían en el ejercicio anterior, que son, como habíamos mencionado, las reglas complejas 90, 122, 150 y 182.



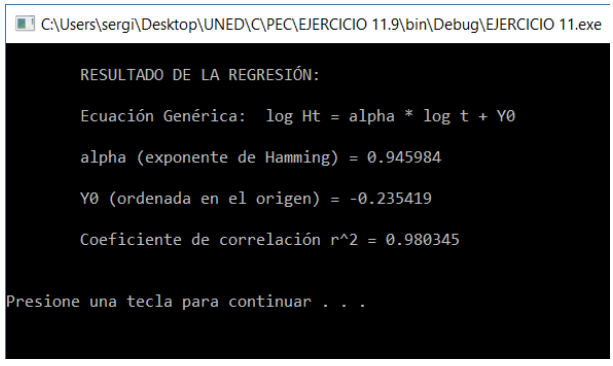
Ajuste R90, α=0.56



Ajuste R122, α=1.05



Ajuste R150, α=0.66



Ajuste R182, α=0.95

D – DISCUSIÓN.

No vamos a comentar aquí la causa de que los coeficientes de Hamming tengan los valores que salen, ya que se hizo con detalle en el ejercicio anterior.

Lo único que cabe destacar en este ejercicio es que, como cabía esperar, los valores obtenidos por este método son iguales a los que se obtienen mediante Gnuplot, no sólo redondeados a dos decimales, como yo los he representado en los pies de imagen, sino que ***coinciden con una precisión de 6 decimales*.**

Otro hecho que se puede observar en el ajuste por este sistema, a diferencia del realizado con Gnuplot, que no proporciona el dato, es que los ***coeficientes de correlación (r^2),*** de las reglas 122 y 182 están mucho más próximos a 1, como se corresponde con un ajuste mucho más preciso por el hecho de estar los puntos de las Distancias de Hamming más próximos a la recta que los de las reglas 90 y 150, en que las distancias de Hamming tienen una dispersión muy grande, ya que son reglas con un patrón de comportamiento particular, como se comentó en el ejercicio anterior.

Ejercicio 15.6

A – OBJETIVO.

Mediante el programa correspondiente, en este ejercicio, vamos a generar un archivo con el nombre “atractor_REGLA.dat” para cada una de las reglas 4, 90, 94, 126, 150 y 182, en el que se van a guardar, para poder representarlo con posterioridad con Gnuplot, los estados visitados por cada ACE en sucesivos pasos de tiempo de evolución, hasta un total de 20, para poder discutir gráficamente los resultados observados.

B – METODOLOGÍA.

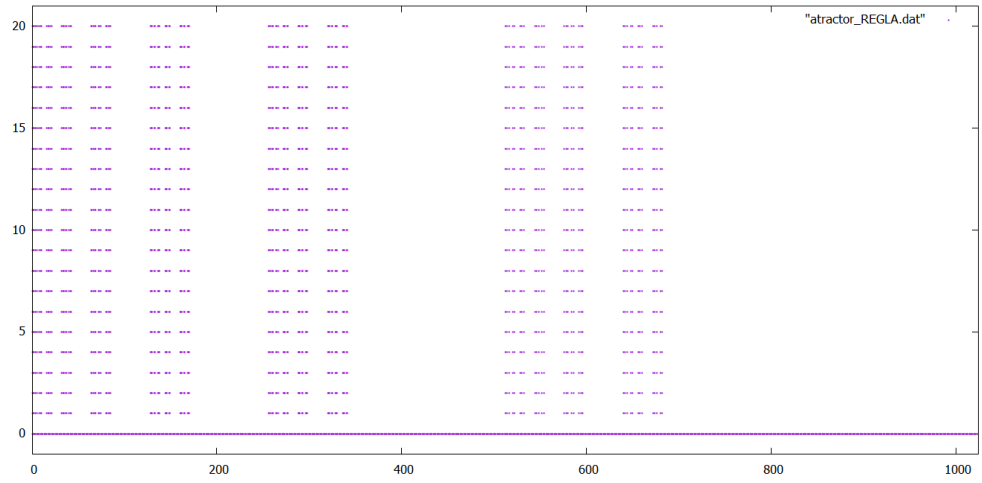
El programa que realizamos que se llamará **EJERCICIO_15_6.c ,** lo que realizará es simular los 20 primeros pasos de evolución de el ACE correspondiente, al mismo tiempo que almacena en una matriz que guarda, en cada paso del ACE, los estados que son visitados.

Posteriormente los datos almacenados en esa matriz son escritos en un archivo con el nombre “atractor_REGLA.dat”, para poder representarlos con Gnuplot, mediante el comando

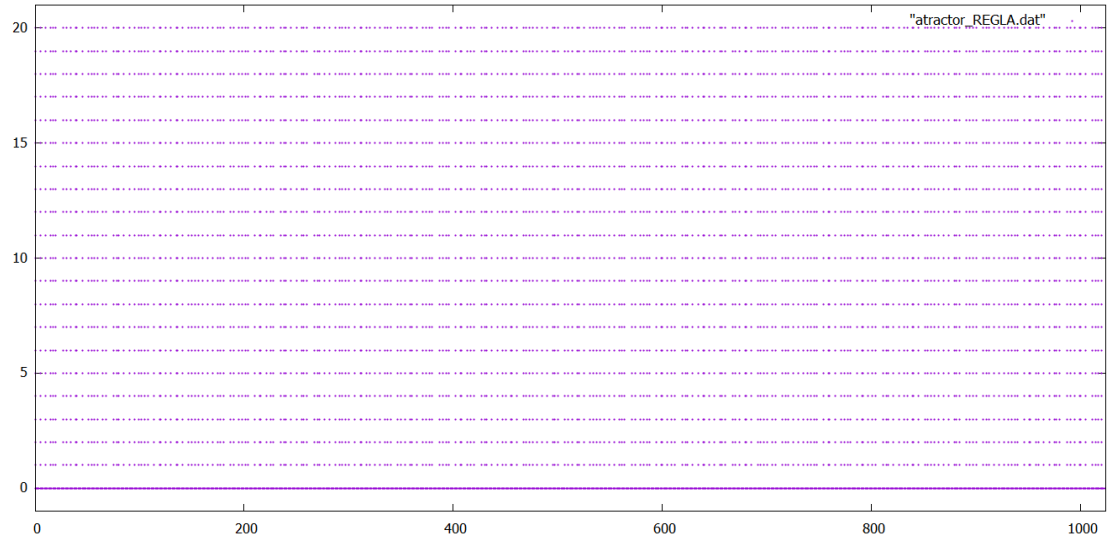
plot [0:1024] [-1:21] “atractor_REGLA.dat” with points pt 7 ps 0.2

C – RESULTADOS.

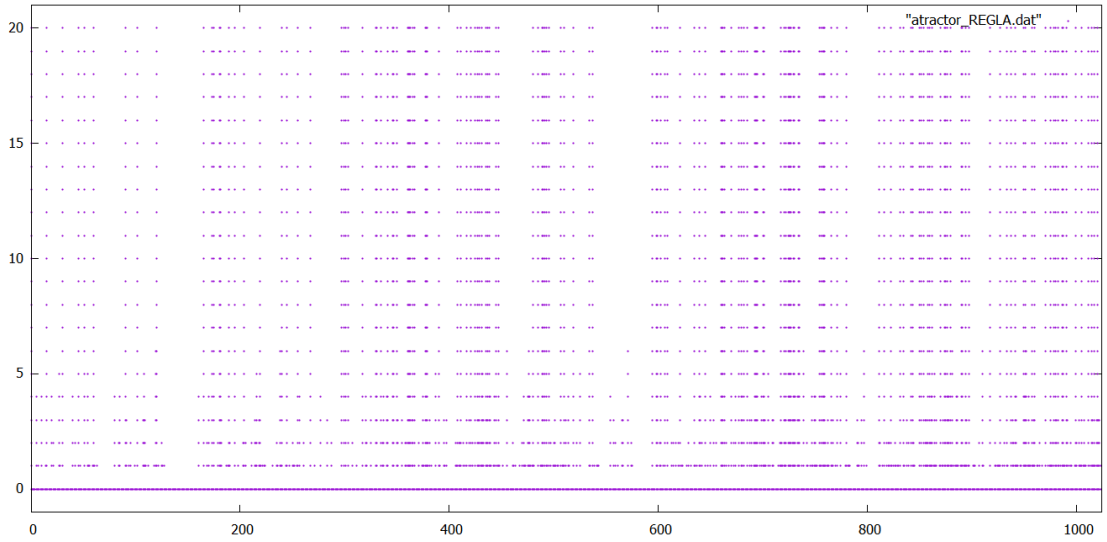
Presentamos las gráficas obtenidas, para cada una de las reglas pedidas, mediante Gnuplot:



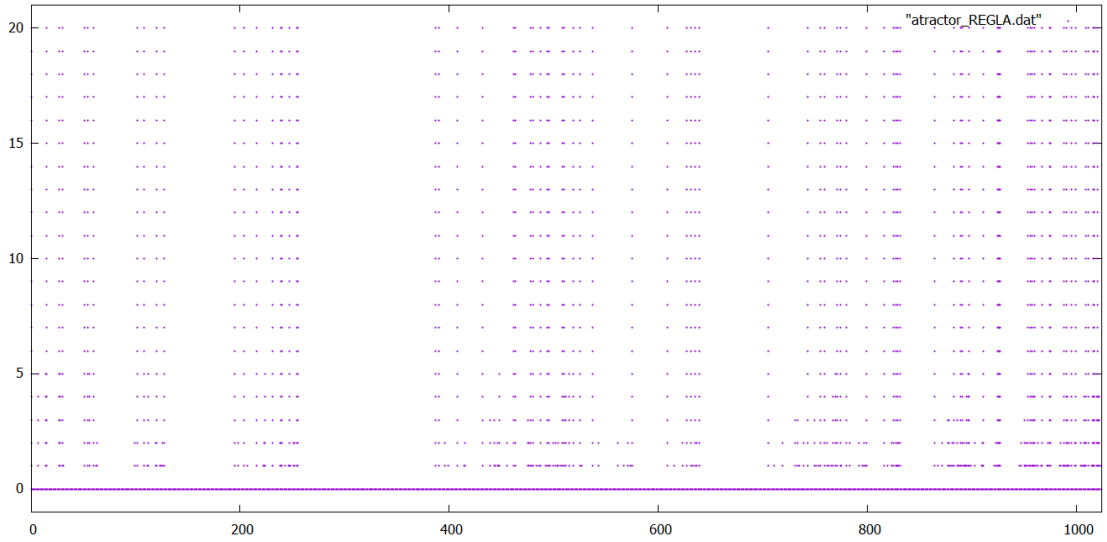
Atractor_REGLA4



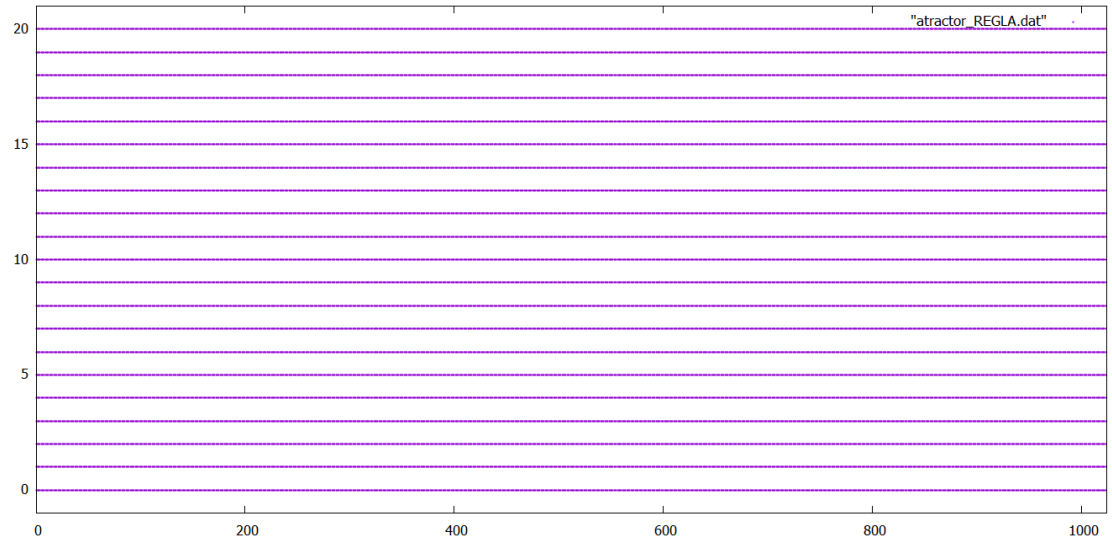
Atractor_REGLA90



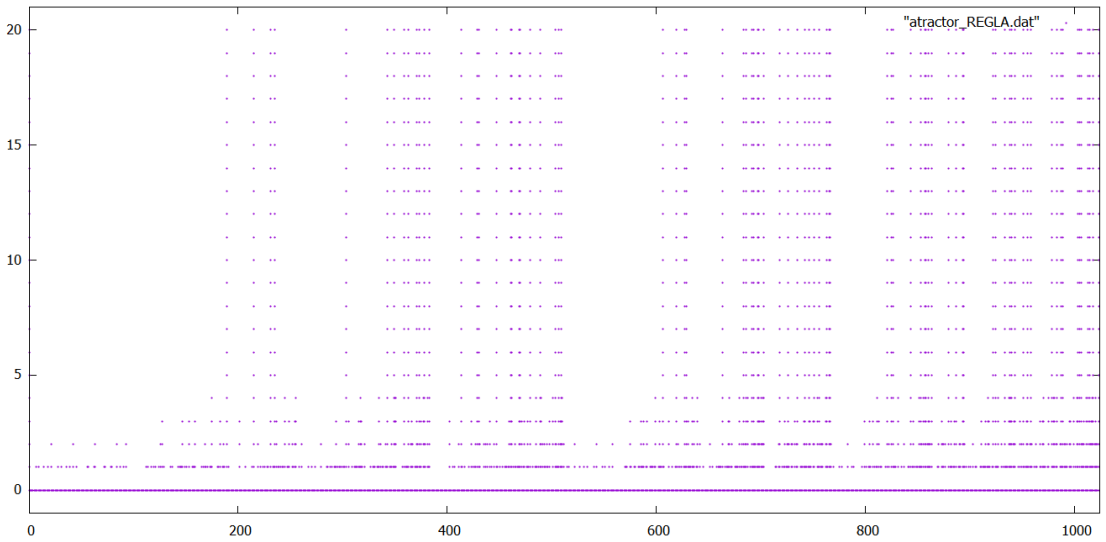
Atractor_REGLA94



Atractor_REGLA126



Atractor_REGLA150



Atractor_REGLA182

D – DISCUSIÓN.

Estos gráficos muestran el comportamiento de las probabilidades para los ACEs de cada regla en función del tiempo, comenzando de un conjunto inicial aleatorio. Como se esperaba de la irreversibilidad de la evolución de los ACEs, diferentes configuraciones alcanzan diferentes probabilidades a medida que evoluciona, y las probabilidades para algunas configuraciones disminuyen hasta cero. El conjunto de configuraciones que sobreviven con una probabilidad no nula tras muchos pasos de evolución del ACE es lo que se conoce como ***atractores*** de la evolución. Este conjunto de atractores constituye un conjunto de Cantor de una dimensión específica para cada regla, siendo menor dicha dimensión, cuanto mayor es la irreversibilidad de la evolución. Si el conjunto de atractores tiene una dimensión de 1, eso significa, por tanto, que se pueden dar todas las configuraciones del ACE en largos periodos, como sucede para la regla 150.

En estos gráficos inicialmente (t=0) todas las configuraciones ocurren con igual probabilidad, por lo que la primera línea tiene marcados todos los puntos. La probabilidad para

una configuración viene dada en las sucesivas líneas en columnas verticales, apareciendo un punto en un particular escalón cuando la probabilidad es distinta de cero. La evolución de ACE modifica la probabilidad con que cada configuración puede ser visitada, produciéndose vacíos en los lugares correspondientes a configuraciones con probabilidad cero. Cuanto más regulares son los patrones de evolución que generan los ACEs, menor es la dimensión del conjunto de atractores, y esto es lo que muestran los gráficos para las diferentes reglas.

Una vez más se comprueba el diferente comportamiento entre reglas aditivas (90 y 150), y las no aditivas (el resto), ya que las primeras tienden a visitar más estados con igual probabilidad (en el caso de la 150 todos), mientras las no aditivas forman rápidamente atractores, dejando muchos estados con una probabilidad de cero.

Ejercicio 15.7

A – OBJETIVO.

Ahora lo que queremos representar es la fracción de estados visitados en función del tiempo, lo cual da información sobre las probabilidades de los atractores, ya que ello influirá en las propiedades estadísticas del conjunto, como la capacidad de formar patrones triangulares, por ejemplo.

B – METODOLOGÍA.

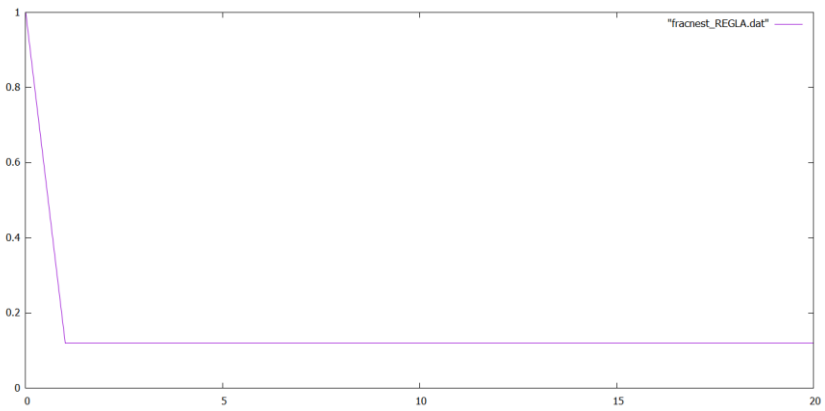
El programa que va a efectuar la tarea de calcular la fracción de estados visitados (nº de estados visitados dividido por el nº de estados posibles) es el archivo **EJERCICIO_15_7.c ,** y el método es, basándose en el programa del ejercicio anterior, cuenta el número de estados visitados en cada paso de tiempo y lo divide entre $\mathbf{2^N}$, que es el número total de estados posibles. En el programa hemos llamado ***nest*** *(nº de estados)* a la variable que guarda el número de estados visitados y ***fracc (****fracción)* a la variable resultante de dividir la anterior por $\mathbf{2^N}$ **.**

Esta variable junto con el contador de pasos se almacenan en el archivo "fracnest_REGLA.dat" para cada regla pedida (las mismas que en el ejercicio anterior).

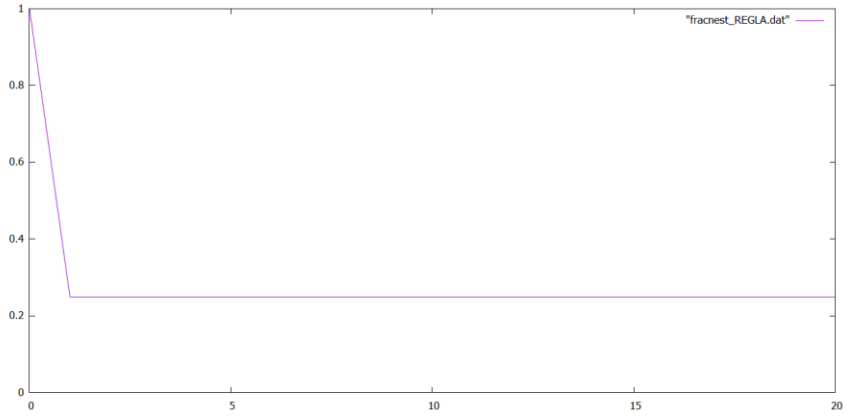
Por último se representan estos archivos para dar un gráfico con Gnuplot.

C – RESULTADOS.

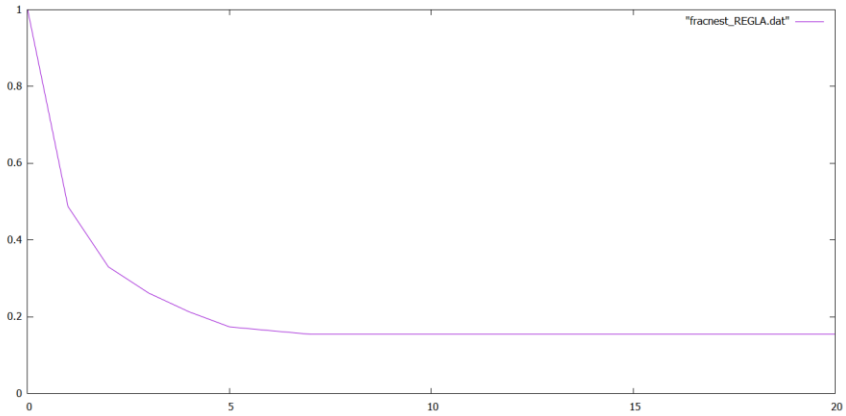
Los gráficos obtenidos mediante el procedimiento anterior son los siguientes:



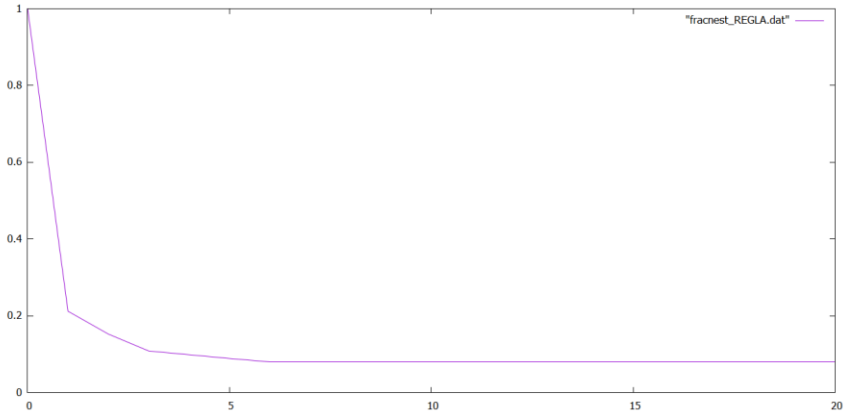
fracnest_REGLA4



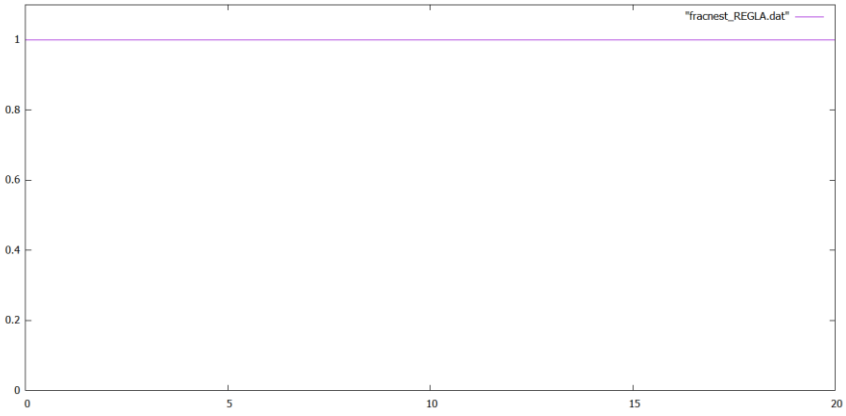
fracnest_REGLA90



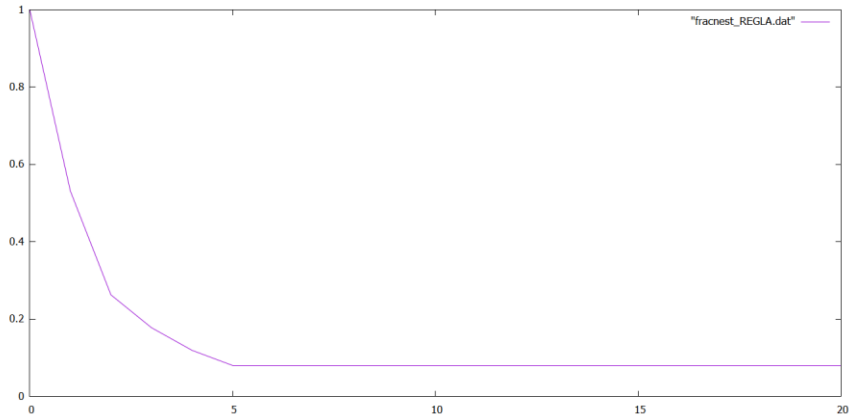
fracnest_REGLA94



fracnest_REGLA126



fracnest_REGLA150



fracnest_REGLA182

D – DISCUSIÓN.

Según se desprende de los cálculos y gráficos obtenidos mediante este programa, se confirma la suposición de que al formarse atractores la fracción de estados visitados por el ACE disminuye hasta un valor estacionario.

Según el número de pasos necesarios para consolidarse los atractores en cada regla, la fracción descenderá con mayor o menor rapidez, lo cual a su vez es un índice de la velocidad de evolución del ACE.

También, como era de esperar a la vista del gráfico de atractores del ejercicio anterior para la regla 150, en el que se ve que no se forma ninguno a medida que avanzan los pasos de evolución, la fracción de estados visitados es igual a 1 constante, es decir todos los estados tienen la misma probabilidad de ser visitados en cada paso de evolución en esta regla.

Bibliografía

- Wolfram S. , “Celular Automata”. *Los Alamos Science.* Vol. 9, pag. 2 – 21. 1983.
- Wolfram S. , “Statistical Mechanics of Cellular Automata”. *Reviews of Modern Physics.* Vol. 55, pag. 601 – 644. 1983.
- Wolfram S. , “Universality and Complexity in Cellular Automata”. *Physica D.* Vol. 10, pag. 1 – 35. 1984.
- Apuntes de la Asignatura. “Temas 12 – 15”. 2018.