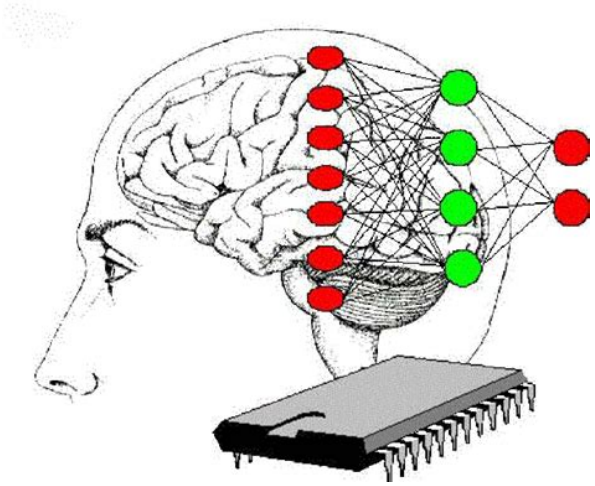

## Facultad de Ciencias

Grado en Física - Física Computacional

# Neurona de Hopfield



Autor:
Pablo Gallardo Calleja

**Resumen**

La finalidad de esta simulación es estudiar el comportamiento emergente de una red de Hopfield. Para llevar a cabo este estudio, en un primer lugar veremos como evoluciona el solapamiento de la red en función de la temperatura a la que se encuentre y del número de patrones que hayamos almacenado en ella. A continuación, calcularemos la capacidad de almacenamiento para una red formada por 100 y 400 neuronas. Por último, estudiaremos de nuevo el comportamiento emergente al cambiar la función de probabilidad de transición para hacer evolucionar la red. Para ello se deducirá una nueva función siguiendo un planteamiento similar a la probabilidad de cambio de estado de la máquina de Boltzman.

# Índice

# 1. Introducción

En 1982 el físico estadunidense John Hopfield, publicó un artículo donde explicaba un modelo matemático de una red neuronal, ahora conocido como red de Hopfield, diseñada para converger a mínimos locales. El desarrollo de esta red suponía la aparición de características computacionales espontáneas resultantes del comportamiento de elementos más simples que componen el sistema. Un ejemplo de este comportamiento sería el carácter ferromagnético de un material, o la memoria asociativa, es decir, la capacidad para recuperar la información a partir de una condición inicial deformada.

Entonces, podemos definir una red de Hopfield como un tipo de red neuronal artificial unicapa, cuya fama viene dada por su capacidad para simular como recuerda el cerebro humano imágenes o patrones distorsionados, lo que significa, simular la memoria autoasociativa. Además, cabe destacar que Hopfield consideró que a cada sistema físico se le podía asignar un carácter de potencial energético, donde los patrones a recordar son mínimos locales de energía estables.

La red de Hopfield es una red recurrente, de manera que cada neurona está conectada con el resto de neuronas, incluido ella misma, pero el valor de esta conexión será 0. Podemos observar estas conexiones en la siguiente figura:

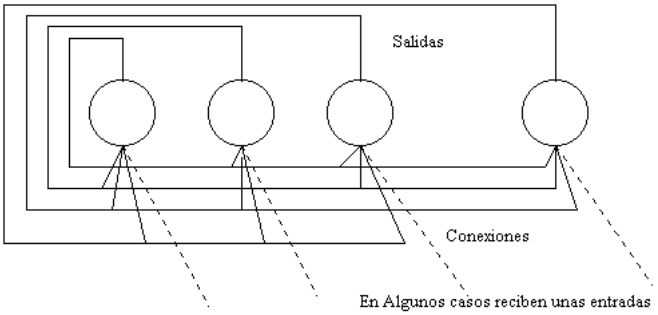



Figura 1: Funcionamiento de una red de Hopfield

Las características que debe cumplir una red recurrente son:

- Debe ser capaz de converger a un estado final que sea estable a partir de un estado incial.
- La convergencia a un estado debe cumplir un criterio de métrica, por ejemplo, el estado final será el más cercano al estado inicial.
- Debe poder almacenar cualquier número de estados estables, aunque como veremos a lo largo del estudio, existirá un límite de patrones que puede almacenar la red, siendo está capaz de converger a uno de los estados guardados.

Tras la publicación del artículo sobre la red en 1982, 3 años más tarde Tank y Hopfield publicaron un artículo sobre la resolución del problema del viajante del comercio. Desde entonces la red se ha empleado en problemas de optimización sencillos, mediante la adaptación del problema a una función coste análoga a la función de energía de Hopfield. De esta manera obtenemos una red cuyos puntos de equilibrio corresponden a las soluciones buscadas, aunque pueden no ser óptimas si no minimiza correctamente la energía del sistema. Podemos destacar algunos de los problemas de optimización que se han logrado convertir a una función energía de Hopfield, como son el problema de las N torres, conversión de analógico a digital, problemas de asignación de redes inalámbricas, restauración de imágenes (que es en el que nos centraremos a lo largo de este estudio), identificación de sistema, optimización combinatoria o el problema del comerciante.

## 2. Fundamento teórico

En una red neuronal de Hopfield se puede definir el hamiltoniano (o energía) del sistema de la forma:

$$H(\mathbf{s}) = -\frac{1}{2}\sum_{i,j} w_{ij}s_i s_j + \sum_i \theta_i s_i \tag{1}$$

Donde el valor de $\theta_i$ es:

$$\theta_i = \frac{1}{2}\sum_j w_{ij} \tag{2}$$

Que son los llamados umbrales de disparo.

Esta es una función de Lyapunov, de forma que nos asegura que, en caso de converger, será a un mínimo local de energía, independientemente de los valores $w_{ij}$ que utilicemos para definir la red. Haciendo una comparación con los potenciales en física, si existen puntos que son un mínimo local de energía, significa que la red tendrá puntos donde será estable. De manera que podemos considerar que la etapa de aprendizaje de la red consiste en añadir estos mínimos locales de energía correspondientes a los patrones que queremos recordar.

A la hora de entrenar la red hemos seguido la siguiente regla para calcular los pesos:

$$w_{ij} = \frac{1}{a(1-a)N}\sum_{\mu=1}^{P}\left(\xi_i^\mu - a\right)\left(\xi_j^\mu - a\right) \tag{3}$$

Donde N es la dimensión del patrón, $a = \langle \xi_i^\mu \rangle$, es decir, el promedio de los valores de cada patrón y $\xi_i^\mu$ nos hace referencia a la componente i-ésima del patrón $\mu$. En esta simulación, con la finalidad de tener en cuenta patrones almacenados con distinta probabilidad, hemos reescrito la función de pesos de la siguiente forma:

$$w_{ij} = \frac{1}{N}\sum_{\mu=1}^{P}\frac{1}{a^\mu(1-a^\mu)}\left(\xi_i^\mu - a^\mu\right)\left(\xi_j^\mu - a^\mu\right) \tag{4}$$

Siendo $a^\mu$ el promedio de cada patrón.

Esta regla para elegir los pesos se basa en la relación de la teoría Hebbiana con las redes neuronales artificiales. Empleando el principio de Hebb se ha buscado establecer los pesos de forma que el valor de estos se aumente si dos componentes de la red (dos neuronas) se activan simultáneamente y en caso contrario, disminuyan. De esta manera se busca ajustar los pesos de modo que se obtenga la mejor relación entre los nodos. Si nos basásemos completamente en este principio para establecer los pesos del sistema, la expresión anterior sería:

$$w_{ij} = \frac{1}{N}\sum_{k=1}^{\mu} s_i^\mu s_j^\mu \tag{5}$$

El motivo principal por el cual empleamos la expresión 4 es para extraer una mayor información total del patrón almacenado a partir de la media, empleándola como si fuera un umbral de activación de la neurona.

Un dato a tener en cuenta, es que el número de patrones que podemos almacenar en la red no es ilimitado, sino que existe un límite de patrones a partir del cual la red deja de ser capaz de recordar. Esto lo veremos con más detenimiento a lo largo del estudio.

Un punto importante a destacar de esta regla de aprendizaje son los patrones espurios o complementarios. Cuando entrenemos la red con un determinado patrón, a este le corresponderá un mínimo local de energía, pero también a su patrón complementario (es decir, al $\widetilde{s_i} = 1 - s_i$), de forma que, a la hora de evaluar un patrón en la red, puede darse el caso de que converja al patrón complementario,

esto se ve en más detenimiento en 7.2. Como ambos patrones son iguales a excepción de una traslación a todas sus componentes, podemos recuperar la información del patrón almacenado simplemente usando $\widetilde{s_i} = 1 - s_i$. Por esto podemos considerar la convergencia al patrón espurio como si fuera al patrón original.

Una vez que estemos evaluando un patrón en la red, emplearemos el coeficiente de solapamiento:

$$m^{\mu}(s) = \frac{1}{a(1-a)N} \sum_i (\xi_i^{\mu} - a)\,(s_i - \frac{1}{2}) \tag{6}$$

Como criterio para determinar si el vector introducido ha convergido o no un patrón $\mu$ almacenado. A lo largo de la simulación podremos ver como evolucionará este coeficiente en función de la temperatura y del número de patrones almacenados.

A la hora de hacer evolucionar el sistema se ha empleado el método de las cadenas de Markov, donde hemos elegido la función de Metropolis:

$$T\,(X \longrightarrow X') = min\left(1, e^{-\beta\left(E(X')-E(X)\right)}\right) \tag{7}$$

Como criterio para determinar la probabilidad de transición de fase entre un estado a otro, con $\beta = \frac{1}{T}$. Como podemos ver, en el exponente de esta función necesitamos emplear la diferencia de energía entre los dos estados para obtener la probabilidad de cambio de fase, por lo tanto, necesitamos $\Delta H = H' - H$. Como en el hamiltoniano del sistema trabajamos con una doble sumatoria, si a la hora de hacer evolucionar el sistema en cada paso debemos resolverla dos veces para cada uno de los estados, entonces la simulación tendría un elevado coste computacional y aumentaría considerablemente el tiempo de ejecución. Para solventar este problema hemos desarrollado en 7.1 una expresión para reducir el coste computacional, resolviendo la expresión $\Delta H$ para el caso particular donde patrones se diferencian en una única componente, obteniendo así:

$$\Delta H = -\sum_i w_{ik} s_i (s'_k - s_k) + \theta_k (s'_k - s_k) \tag{8}$$

Observamos que se ha podido pasar de una función cuadrática a una lineal, por lo tanto, para dimensiones elevadas de la red obtendremos una gran ventaja al usar esta simplificación.

Si nos fijamos en la función de Metropolis aparece la temperatura del sistema. Este parámetro será el que gobierne el proceso de convergencia a un patrón determinado. Si hacemos tender en la función de Metropolis $T \longrightarrow \infty$ esto hará que el valor que toma la función será siempre uno, de forma que cualquier opción de transición de fase será aceptada como válida, independientemente del valor de la diferencia del hamiltoniano. En este caso, el sistema no convergerá a los mínimos locales de energía y, por lo tanto, no alcanzará un estado estacionario correspondiente al de uno de los patrones almacenados. En consecuencia, podemos definir la temperatura crítica $T_c$ como la temperatura a partir de la cual el sistema no converge a ningún estado estacionario. El valor de esta temperatura será uno de los parámetros que estudiaremos a lo largo de la simulación.

Siguiendo un procedimiento similar a la máquina de Boltzman, obtenemos en 7.5, llegando a:

$$p_{i=on} = \frac{1}{1 + e^{\frac{\Delta E_{on,off}}{T}}} \tag{9}$$

Esta función nos da la probabilidad de transición de un spin de la red a otro. Si nos fijamos, la función obtenida es una función sigmoide que pertenece a la familia de funciones logísticas. Podemos destacar como principal propiedad de esta función el crecimiento acotado por el intervalo $0 < f(t) < 1$, esta propiedad hace idónea a la función para obtener una probabilidad de transición. Cabe destacar, que este tipo de funciones tienen un gran número de aplicaciones para realizar modelos de crecimiento de poblaciones, propagaciones de enfermedades, difusión en redes sociales...

# 3. Metodología

Podemos dividir este estudio en tres partes, una primera donde estudiaremos la capacidad de recordar de una red de Hopfield en función de la temperatura, seguida de un apartado donde estudiaremos el número máximo de patrones que es capaz de almacenar la red y para terminar repetiremos el mismo estudio empleando la función logística como función de transición. Por lo tanto:

- En esta primera parte, hemos estudiado la relación de la temperatura con la capacidad de recordar de la red, para ello hemos observado inicialmente como varía el coeficiente de solapamiento 6 y la energía, en una situación donde solo hallamos almacenado un patrón. De manera que, en un primer paso entrenamos la red para que almacene el patrón, y después le introducimos como entrada el patrón deformado mediante distintas probabilidades o con una condición generada de forma aleatoria. Tras dejar evolucionar el sistema el tiempo suficiente para que alcance el estado estacionario, obtendremos el coeficiente de solapamiento con respecto al patrón almacenado. Repetiremos este proceso para las distintas condiciones iniciales, desde temperaturas próximas a 0,00 K hasta 1,10 K. A partir de los datos obtenidos veremos como evoluciona el coeficiente de solapamiento y la energía del sistema con la temperatura Con estos resultados podremos determinar la temperatura crítica del sistema. Después, procederemos a repetir la experiencia almacenando varios patrones en la red, donde al igual que antes, obtendremos la evolución del coeficiente de solapamiento respecto a cada patrón en función de la temperatura, la energía del sistema y la temperatura crítica de este.

- En esta segunda parte, estudiaremos como varía la capacidad para recordar de la red a una temperatura de $T_c = 1{,}00 \cdot 10^{-5}$ K en función del número de patrones que almacenemos que para una red de 100 y 400 neuronas. Para ello, los patrones que serán almacenados en la red, se generaran forma aleatoria y empleando como condición inicial una leve deformación del primer patrón almacenado en la red. Para la red de 100 neuronas, llegaremos a almacenar hasta 70 patrones diferentes, por otro lado, para la red de 400 neuronas, se realizará para 120 patrones almacenados. Repetiremos este experimento varias veces para poder determinar el comportamiento del sistema sin que tenga efecto las condiciones iniciales a las que se encuentre sometido. De manera que al sistema de 100 neuronas se va a repetir la experiencia 60 veces, por otro lado, para la red de 400 neuronas, por motivos de coste y tiempo computacional, únicamente se va a repetir el experimento 20 veces. A partir de los datos obtenidos, podremos calcular la fracción entre el número de patrones almacenados y el tamaño de la red ($\alpha = \frac{P_c}{N}$).

- Para estudiar como se comporta la red de Hopfield usando una función logística como método para obtener la probabilidad de transición, se ha repetido el mismo procedimiento mencionado en los dos apartados anteriores para esta segunda función.

# 4. Resultados

## 4.1. Un Patrón almacenado

En este apartado nos centraremos en estudiar como se comporta el sistema cuando hemos almacenado un único patrón en él. Para ello vamos a determinar como varía el coeficiente de solapamiento y la energía del sistema con la temperatura, y a partir de los resultados que obtengamos determinaremos la temperatura crítica del sistema. Analizando los datos, podremos determinar si la temperatura crítica es una propiedad intrínseca del sistema o depende de las condiciones iniciales introducidas en él. Durante el estudio hemos distinguido entre dos casos, el primero, cuando la condición inicial de la red es un patrón aleatorio y en un segundo caso, cuando es deformado con una determinada probabilidad.

El patrón que hemos almacenado en la red es:

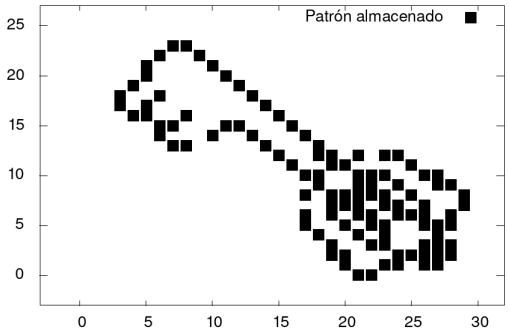



Figura 2: Patrón almacenado

El patrón tiene dimensiones de 30x30, por lo tanto, la red resultante tendrá una dimensión de 900 neuronas. Para estudiar el sistema, lo hemos hecho evolucionar 50 pasos de Monte Carlo (pMC).

Para poder repetir los resultados aquí obtenidos es necesario que vaya a la carpeta de **codigo** y después a **solapamiento_1_patron** o **solapamiento_1_patron_deformado**. Una vez dentro, compile el archivo **solapamiento_1_patron.cpp**, si quiere que se ejecute el programa directamente, o **solapamiento_1_patron_2.cpp** si quiere introducir los parámetros del programa. En 7.7 señalamos los pasos a seguir para compilar el programa y los paquetes necesarios para compilarlo. Tenga en cuenta a la hora de compilar el programa que debe incluir los ficheros **Hopfield.cpp**, **vector.cpp** y **aleatorios.f**. Una vez obtenido el ejecutable, ejecute el programa, asegúrese que las carpetas **inicio** (donde se encontrará el patrón a almacenar en la red en un fichero llamado **modelo.dat**), **caracterisiticas** y **sistema** están creadas. Tras la ejecución del programa, vaya a **caracteristicas** y encontrará dos ficheros **T_solapamiento.dat** y **T_Energia.dat**, los cuales contienen la evolución de la energía y el coeficiente de solapamiento en función de la temperatura. En esa misma carpeta encontrará un directorio llamado **evolucion** que contiene la evolución del coeficiente de solapamiento en función del número de iteraciones para distintas temperaturas. Por otro lado, en la carpeta **sistema**, se nos almacenará la condición inicial introducida al programa junto con los estados finales del sistema a distintas temperaturas. Además, en cada una de las carpetas indicadas anteriormente, donde se almacenarán los resultados, encontramos los ficheros **.plt** para generar los gráficos. Al principio de cada código encontrará una sección llamada CARACTERISTICAS donde definimos los parámetros del programa:

- N, nos indica la dimensión de la red. Tenga en cuenta que la dimensión debe coincidir con la del patrón almacenado en el fichero **modelo.dat** que encontramos en **inicio**. Para esta situación, el valor empleado es $N = 900$.

- paso, el número de pasos de Monte Carlo a realizar. La duración de la ejecución del programa viene ligada a este parámetro. Como hemos mencionado antes, hemos empleado paso= 50 pMC.

- T_max, temperatura máxima que evaluará el programa, tanto en esta simulación como en el resto se ha usado $T_{max} = 1{,}10$ K.

- T_min, temperatura a la que cambiamos el incremento de la temperatura, se recomienda usar el valor de $T_{min} = 0{,}10$ K.

- fact, factor para indicar cada cuanto guardamos los datos, el valor que hemos empleado es fact= 10000.

- hT, incremento de la temperatura en cada paso, para obtener unos buenos resultados se aconseja usar hT= 0,01 K.

- T_0 temperatura a la que empezamos a evaluar, se aconseja emplear un valor bajo como $T = 1{,}00 \cdot 10^{-7}$ K.

- probabilidad, constante con la que generamos o deformamos los patrones que introducimos a la red.

- crite_estb, número de pasos que eliminamos para considerar solo el estado estacionario. Tenga en cuenta que la ejecución del programa tiene una fuerte dependencia con este valor, en caso de ser muy pequeño, se tomarán valores del solapamiento que no pertenecen al estado estacionario. Por otro lado, si tomamos valores más grandes que el número de iteraciones, se producirá una violación de segmento y se parará la ejecución del programa. Hemos empleado un valor de crite_estb=1000.

Modificando estos parámetros se podrá modificar las características del programa.

En caso de estar ejecutando **solapamiento_1_patron_2.cpp**, los parámetros indicados anteriormente se nos pedirán que los introduzcamos por teclado. Para todos los programas utilizados en la práctica, los que emplean la terminación _2 hacen referencia a los que son necesarios introducir los parámetros por teclado, mientras que para el resto no es necesaria.

A la hora de descartar pasos para calcular la media y desviación de los datos, hemos empleado 7.3 como criterio de convergencia para eliminar los datos que no estén en el estado estacionario.

### 4.1.1. Patrón aleatorio

El patrón inicial que hemos introducido a la red es:

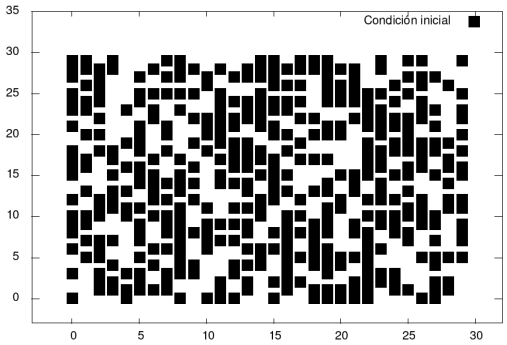



Figura 3: Condición inicial

Como podemos ver, la situación inicial de partida del patrón está muy desviada del patrón inicial.

La convergencia del solapamiento en función de la temperatura es la siguiente:

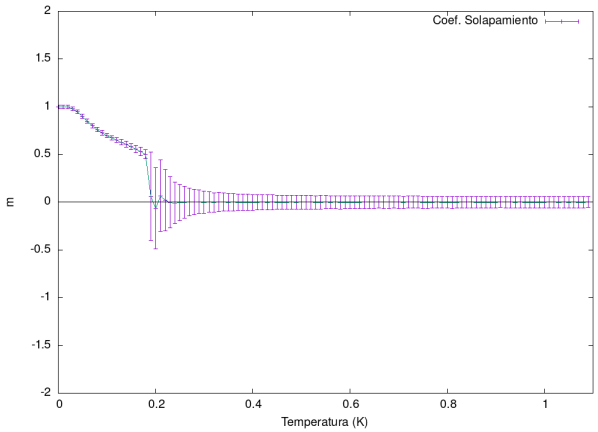



Figura 4: Coeficiente de solapamiento en función de la temperatura

Si nos fijamos el coeficiente de solapamiento va decayendo con la temperatura. Podemos destacar los valores que toma el coeficiente en torno a T=0,20 K, donde se produce una fuerte caída en la tendencia que sigue el coeficiente de solapamiento, por lo tanto, nos estamos acercando ya a la temperatura crítica. Si observamos la evolución del coeficiente de solapamiento en función del número de iteraciones para cada temperatura, se puede determinar una temperatura crítica de T=0,2400$\pm$0,0029K. A partir de esta temperatura, el sistema no es capaz de alcanzar en ningún momento el estado estacionario.

Si analizamos la energía del sistema en función de la temperatura:

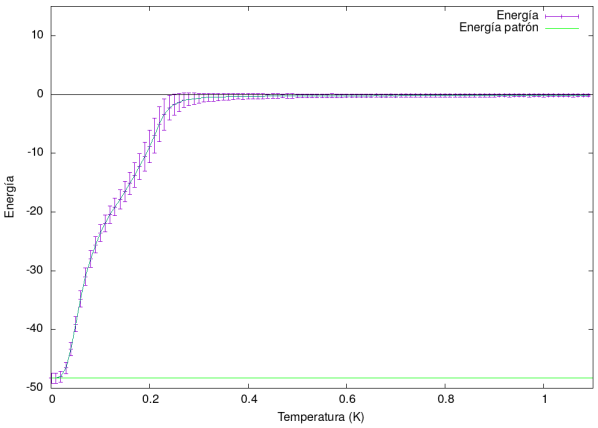
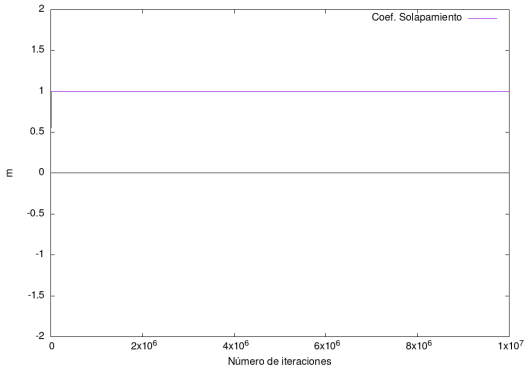



Figura 5: Energía en función de la temperatura

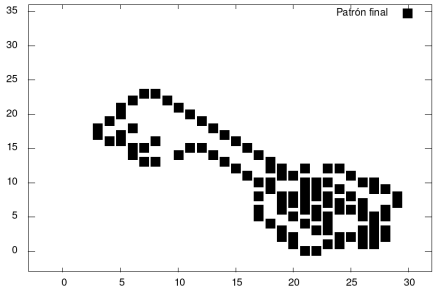
Observando la curva a temperaturas bajas, se puede ver converge perfectamente a la energía del patrón almacenado, pero conforme vamos aumentando la temperatura, esa convergencia va desapareciendo y tiende a tomar valores de energía más elevados. Una vez superada la temperatura crítica, los valores energéticos que toma el sistema serán los correspondientes a configuraciones aleatorias.

Estudiando la convergencia del coeficiente de solapamiento en función del número de iteraciones y el estado final del patrón, para distintas temperaturas:

- $T = 1{,}00 \cdot 10^{-7}$ K:



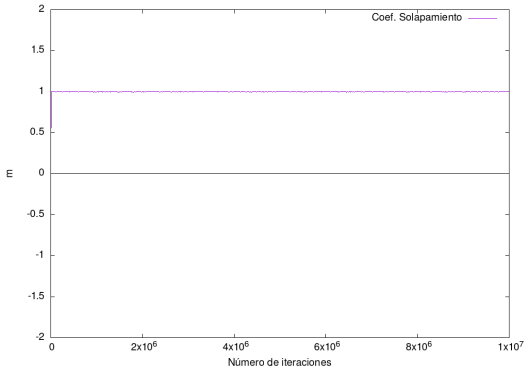

(a) Solapamiento a T=1,00 $\cdot 10^{-7}$ K



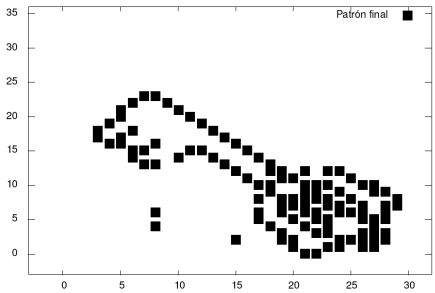

(b) Estado final a T=1,00 $\cdot 10^{-7}$ K

Como habíamos visto previamente en 4 a temperaturas bajas, la convergencia al estado estacionario, es prácticamente directa y muy estable, ya que el coeficiente de solapamiento no sufre ninguna fluctuación. Además, observamos como se ha recuperado por completo el patrón original.

- $T = 2{,}00 \cdot 10^{-2}$ K:



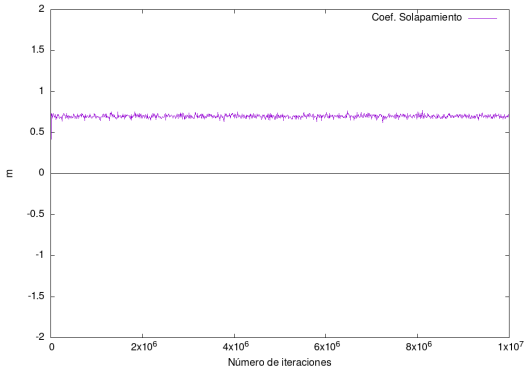

(a) Solapamiento a T=2,00 $\cdot 10^{-2}$ K



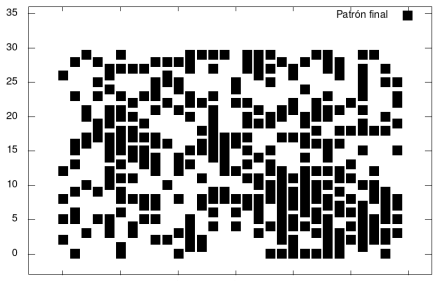

(b) Estado final a T=2,00 $\cdot 10^{-2}$ K

Vemos que la convergencia sigue siendo prácticamente instantánea, aunque, una vez alcanzado el estado estacionario se puede observar unas pequeñas fluctuaciones del sistema, estas variaciones en el estado de equilibrio producen que se recupere perfectamente el patón original salvo un conjunto de 4 puntos.

- $T = 0{,}10$ K:




(a) Solapamiento a T=0,10 K

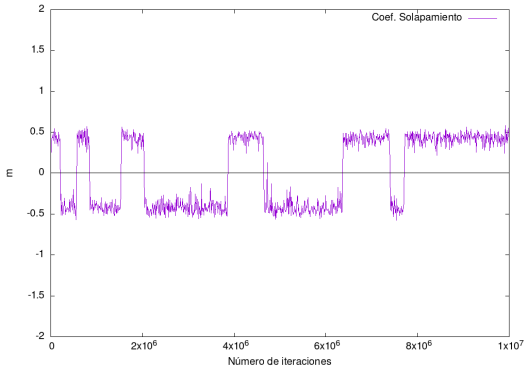



(b) Estado final a T=0,10 K

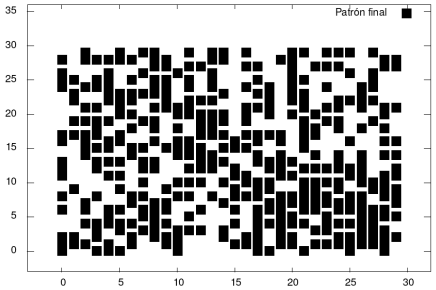
Aquí podemos observar como el sistema ya no llega a alcanzar el total solapamiento con el patrón

almacenado, pero si alcanza el estado estacionario perfectamente. Como el valor del solapamiento en el estado estacionario no es la unidad, ya no se logra recuperar toda la información del patrón original. Si nos fijamos la forma del patrón se puede discernir levemente entre la nube de puntos, pero no se logra recuperar en su totalidad.

- $T = 0{,}20$ K:




(a) Solapamiento a T=0,20 K

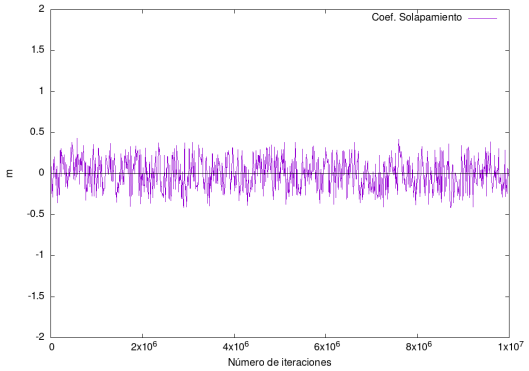



(b) Estado final a T=0,20 K

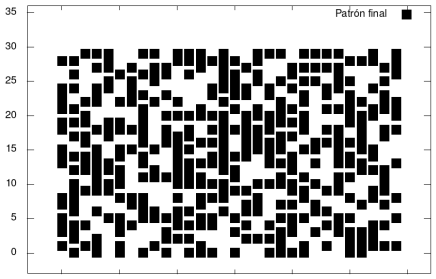
Como podemos observar, el sistema ya no alcanza un único estado estacionario, sino que va oscilando entre dos posibles valores del coeficiente de solapamiento sin llegar a converger a uno perfectamente. Además, a esta temperatura, ya no se puede lograr recuperar la información del sistema, quedando una nube de puntos sin apenas relación entre ellos.

Para temperaturas superiores a la crítica, el sistema ya no alcanza en ningún momento un estado estacionario y se queda oscilando en torno al 0. Como ya no converge a ningún estado estacionario, resulta imposible recuperar la información inicial del sistema:

- $T = 0{,}25$ K:

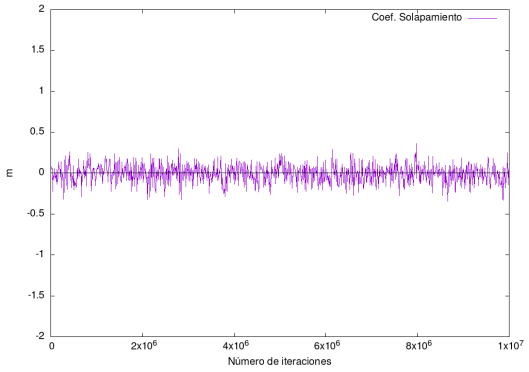



(a) Solapamiento a T=0,25 K

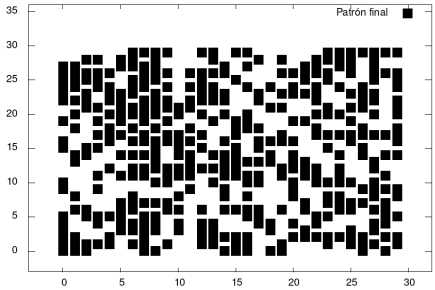



(b) Estado final a T=0,25 K

- $T = 0{,}30$ K:

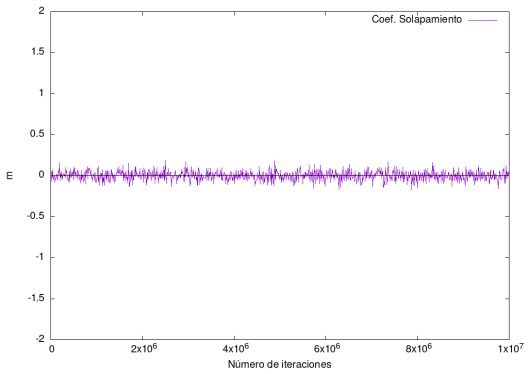



(a) Solapamiento a T=0,30 K

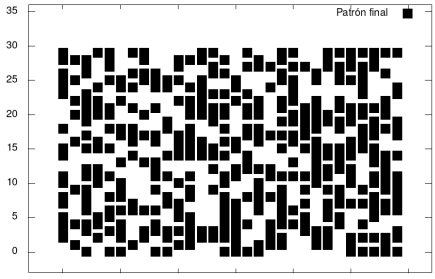



(b) Estado final a T=0,30 K

- $T = 1{,}00$ K:




(a) Solapamiento a T=1,00 K




(b) Estado final a T=1,00 K

### 4.1.2. Patrón deformado con probabilidad 20 %

La condición inicial que hemos introducido en el sistema es:

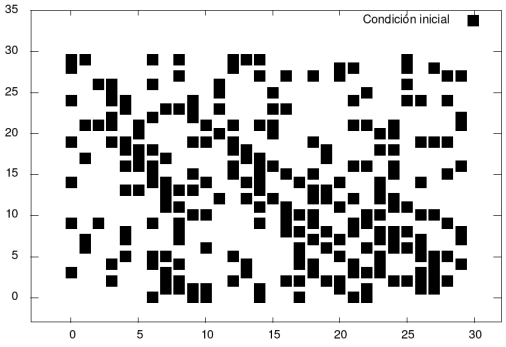



Figura 13: Condición inicial

Al haber establecido una deformación pequeña, apenas se ha modificado el patrón original, pudiendo discernir la forma original del patrón en la nube de puntos.

Si nos fijamos en la evolución del coeficiente de solapamiento con la temperatura:

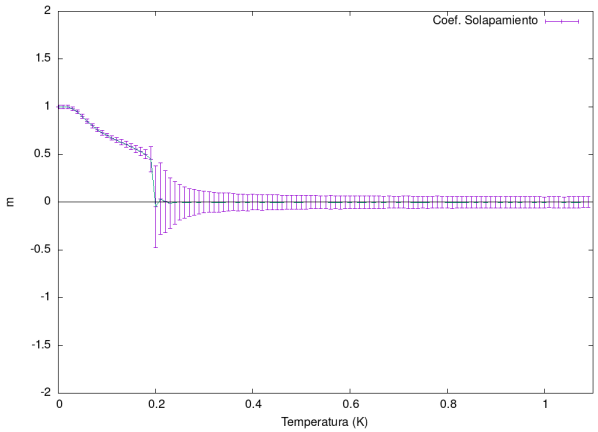



Figura 14: Solapamiento frente a temperatura

Si observamos la gráfica, podemos ver perfectamente como el coeficiente de solapamiento va decayendo con la temperatura hasta llegar un punto donde el valor del coeficiente se aproxima cada vez más a 0. A partir de los datos obtenidos, hemos determinado una temperatura crítica de: $T_c = 0{,}2400 \pm 0{,}0029$ K.

Si volvemos a analizar como evoluciona la energía del sistema:

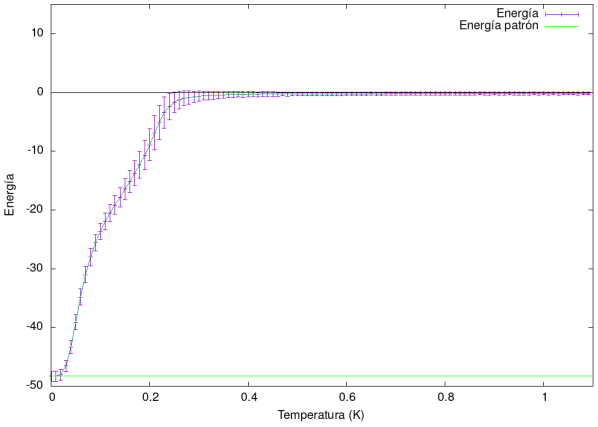



Figura 15: Energía frente a temperatura

Como podemos ver, a temperaturas bajas, converge perfectamente al valor de la energía del patrón almacenado, mientras que conforme vamos aumentando la temperatura empieza a tomar valores más energéticos, de forma que se va alejando de la posición de equilibrio. Cuando superamos la temperatura crítica, al igual que en el apartado anterior, observamos como toma valores cercanos al 0, que corresponde con máximos de energía.

Tanto en esta situación como en posteriores, me abstendré en mostrar la evolución de $m$ en función del número de iteraciones y el estado final del sistema si son equivalentes a los resultados obtenidos en 4.1.1.

### 4.1.3. Patrón deformado con probabilidad 50 %

La condición inicial que hemos introducido en el sistema es:

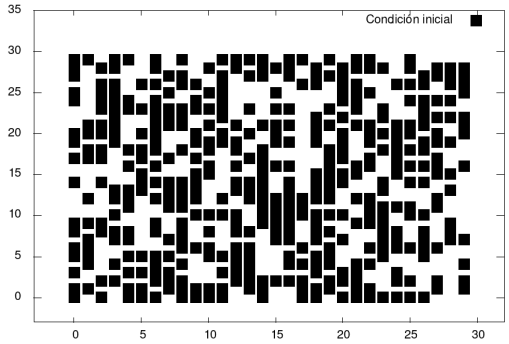



Figura 16: Condición inicial

Tras la simulación, el coeficiente de solapamiento en función de la temperatura es el siguiente:

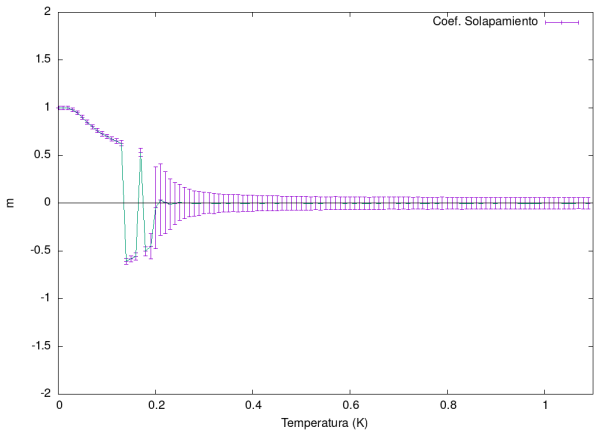



Figura 17: Solapamiento frente a temperatura

En esta situación podemos ver como el sistema, dependiendo de la temperatura, converge a los estados espurios, cosa que no ocurría en los dos casos anteriores, aun así, podemos observar como el coeficiente tiende a 0 con la temperatura. Podemos determinar de nuevo la temperatura crítica del sistema, obteniendo un valor de $T_c = 0{,}2400 \pm 0{,}0029$ K.

Al igual que antes, estudiamos como evoluciona la energía del sistema:

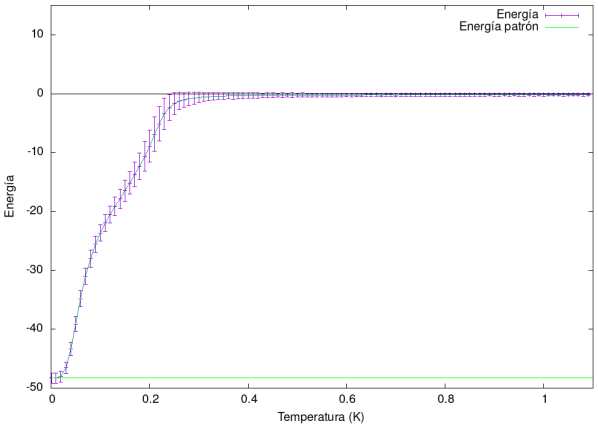



Figura 18: Energía frente a temperatura

Si nos fijamos en las temperaturas bajas, la energía converge a la energía del patrón almacenado en la red, mientras que conforme aumentamos la temperatura, este valor diverge a valores de energía más elevados, correspondiente a estados no estacionarios con configuraciones aleatorias.

### 4.1.4. Patrón deformado con probabilidad 70 %

La condición inicial que hemos introducido en el sistema es:

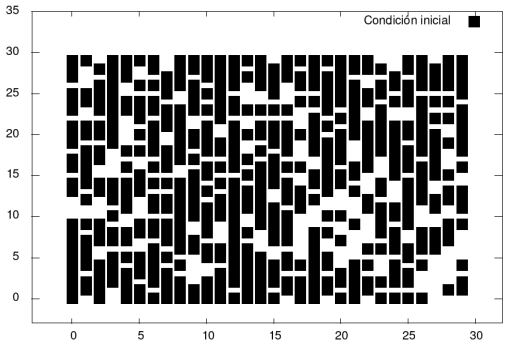



Figura 19: Condición inicial

Tras la simulación, el coeficiente de solapamiento que alcanza a cada temperatura es el siguiente:

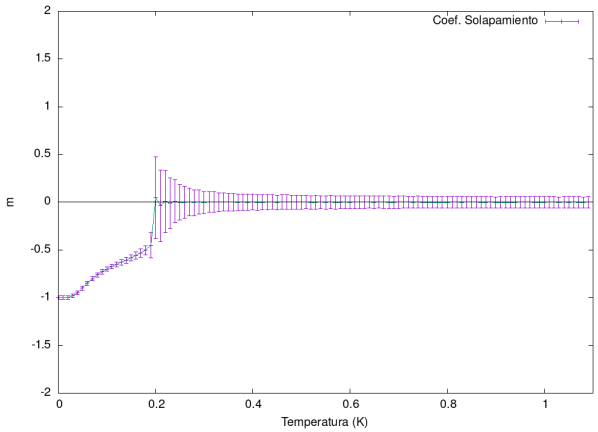



Figura 20: Solapamiento frente a temperatura

Se puede ver en este caso predomina la convergencia al patrón complementario, pero, al igual que en los otros casos, esta convergencia tiende a 0 con la temperatura, donde se queda oscilando en torno a la posición de equilibrio. Si recurrimos de nuevo a la evolución del coeficiente de solapamiento a cada temperatura, obtenemos una temperatura crítica del sistema de $T_c = 0{,}2400 \pm 0{,}0029$ K.

Al igual que antes, estudiamos como evoluciona la energía del sistema:

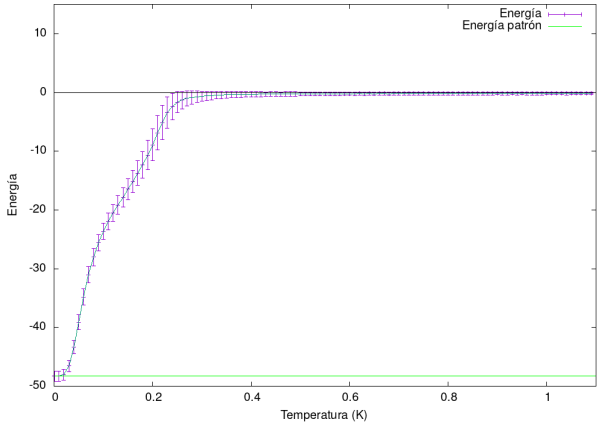
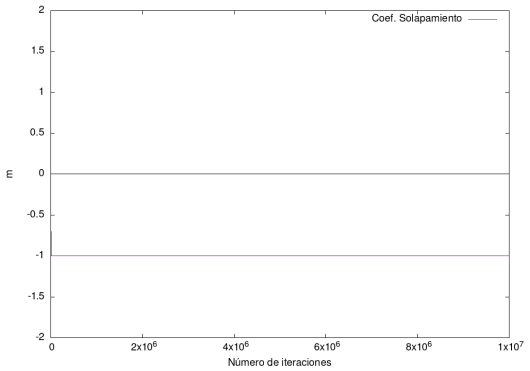



Figura 21: Energía frente a temperatura

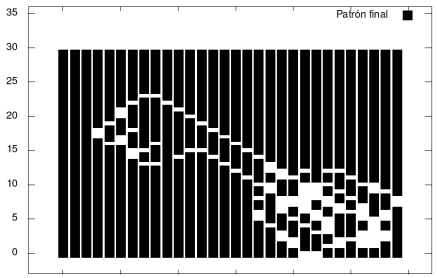
Si nos fijamos en la convergencia de la energía en función de la temperatura, vemos como sigue la misma tendencia obtenida en los 3 apartados anteriores, de manera, que hemos comprobado de forma experimental que la energía del patrón espurio es un mínimo local de energía, y coincide con la del patrón almacenado.

Estudiando la convergencia del coeficiente de solapamiento en función del número de iteraciones y el estado final del patrón para distintas temperaturas, observamos:

- $T = 1{,}00 \cdot 10^{-7}$ K:

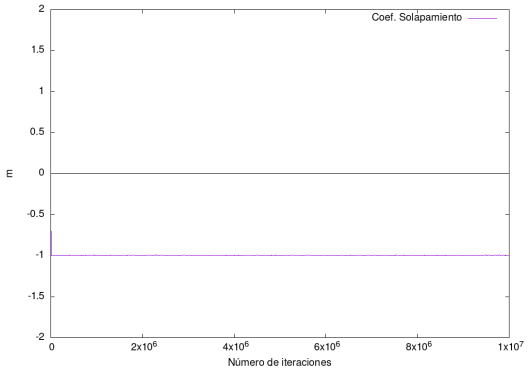



(a) Solapamiento a T=$1{,}00 \cdot 10^{-7}$ K

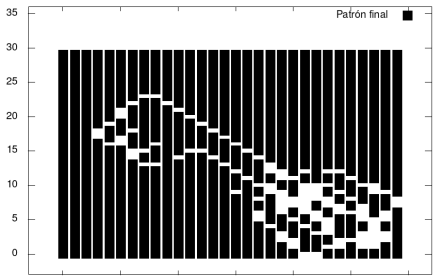



(b) Estado final a T=$1{,}00 \cdot 10^{-7}$ K

- $T = 2{,}00 \cdot 10^{-2}$ K:

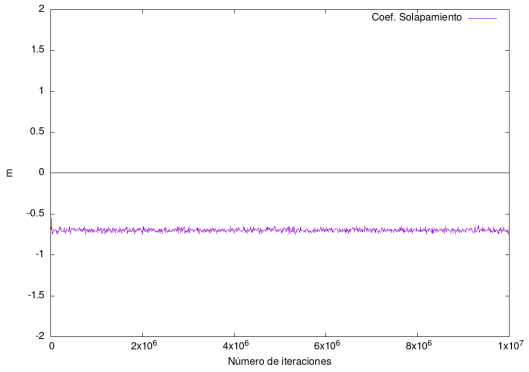



(a) Solapamiento a T=$2{,}00 \cdot 10^{-2}$ K

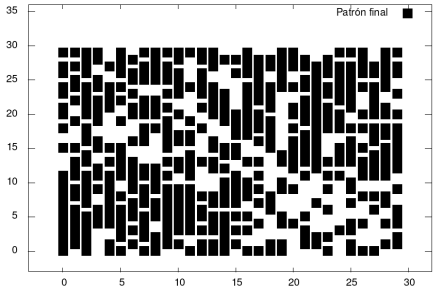



(b) Estado final a T=$2{,}00 \cdot 10^{-2}$ K

- $T = 0{,}10$ K:

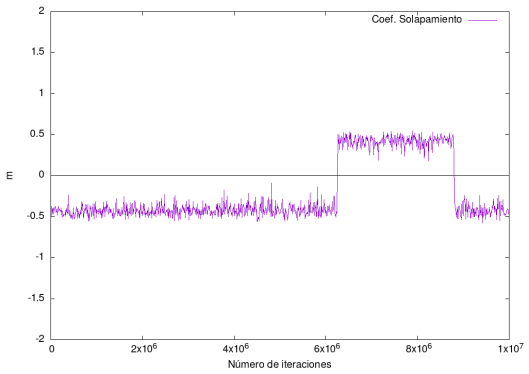



(a) Solapamiento a T=0,10 K

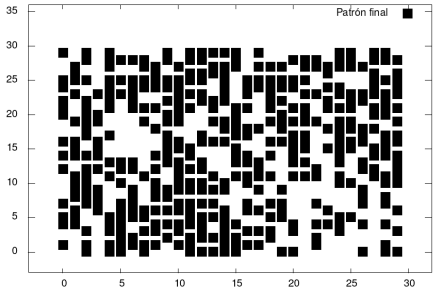



(b) Estado final a T=0,10 K

- $T = 0{,}20$ K:




(a) Solapamiento a T=0,20 K

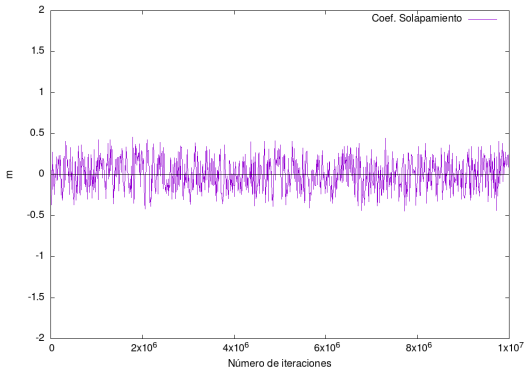



(b) Estado final a T=0,20 K

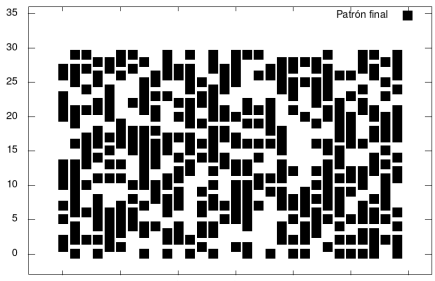
Como en esta situación la condición inicial introducida a la red es más cercana al estado espurio, podemos ver como a la hora de converger lo hace al patrón complementario. Aun así, los resultados obtenidos son similares a los apartados anteriores.

Para temperaturas superiores a la crítica, el sistema ya no alcanza en ningún momento un estado estacionario y se queda oscilando en torno al 0. Como ya no converge a ningún estado estacionario, resulta imposible recuperar la información inicial del sistema:

- $T = 0{,}25$ K:

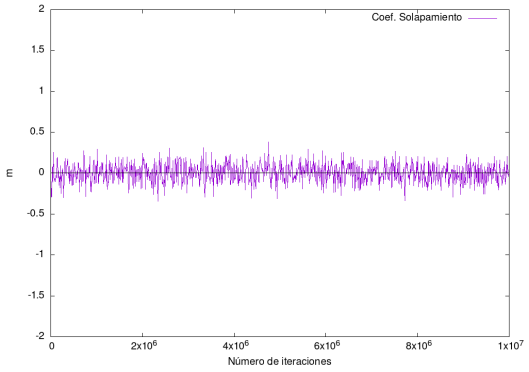



(a) Solapamiento a T=0,25 K

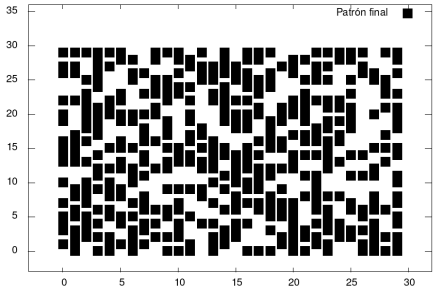



(b) Estado final a T=0,25 K

- $T = 0{,}30$ K:

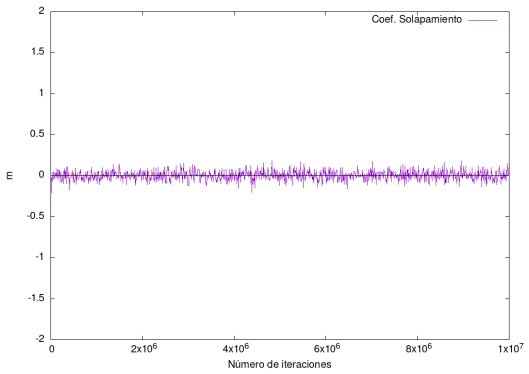



(a) Solapamiento a T=0,30 K

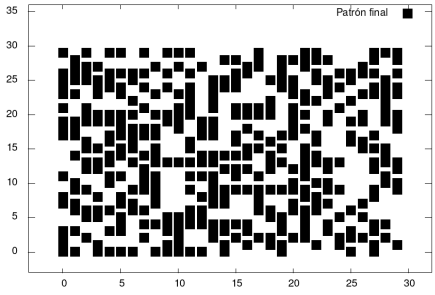



(b) Estado final a T=0,30 K

- $T = 1{,}00$ K:




(a) Solapamiento a T=1,00 K

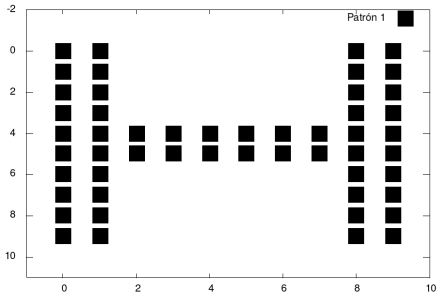



(b) Estado final a T=1,00 K

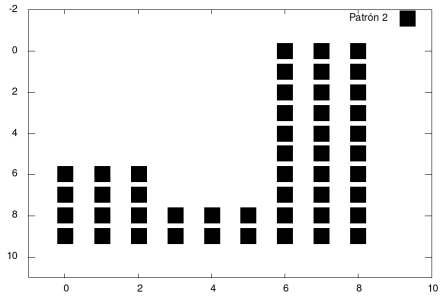
## 4.2. Varios patrones almacenados, con una condición inicial deformada

En este apartado vamos a estudiar como evoluciona el coeficiente de solapamiento en función de la temperatura y el número de patrones almacenados. Además, determinaremos de forma experimental la temperatura crítica. Al igual que en los apartados anteriores, también estudiaremos como varía la energía del sistema.

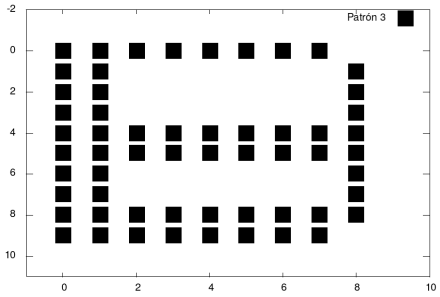
Los patrones que vamos a almacenar en la red son:



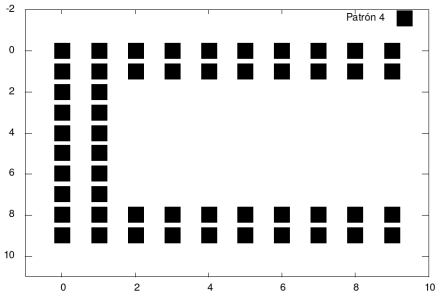

(a) Patrón 1



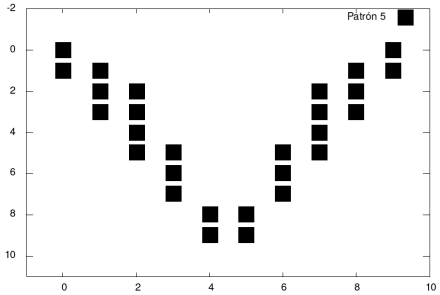

(b) Patrón 2



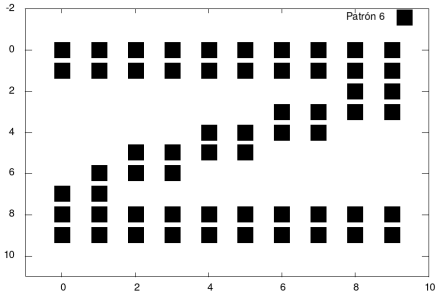

(a) Patrón 3



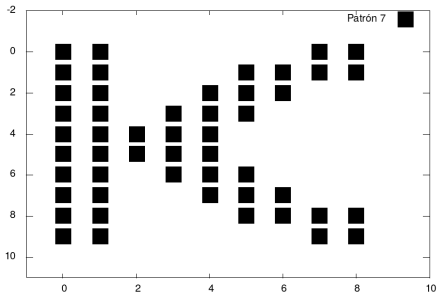

(b) Patrón 4




(a) Patrón 5




(b) Patrón 6




(a) Patrón 7

En esta sección vamos a trabajar con una red de 100 neuronas, ($N = 100$). Como su tamaño es considerablemente menor que la empleada en el apartado previo, solamente emplearemos 30 pMC para hacer evolucionar a la red. Introduciremos como condición inicial:

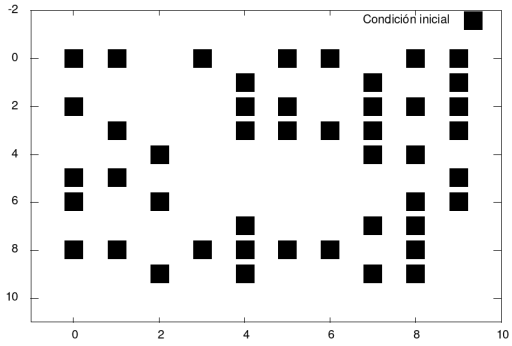



Figura 33: Condición inicial

Para reproducir los resultados obtenidos en esta experiencia, es necesario ir a **codigo** y después a **varios_patrones**, compilamos y ejecutamos el archivo **varios_patrones.cpp** o **varios_patrones_2.cpp**. En este caso los ficheros generados se almacenarán en **caracteristicas**, dentro de esta carpeta, pondremos ver las carpetas correspondientes a la red para cada número de patrones almacenados. Dentro de cada una de ellas encontraremos la evolución de la energía y el solapamiento en función de la temperatura, junto con el directorio **evolucion**, en él encontraremos la evolución del coeficiente de solapamiento en función del número de iteraciones. Por otro lado, en la carpeta de **sistema**, volvemos a encontrar una carpeta para cada uno de los patrones almacenados, una vez dentro se podrá ver el estado final del sistema para cada una de las temperaturas. En este código podemos modificar los parámetros del programa de la misma forma que en el apartado 4.1, a excepción de un nuevo parámetro a tener en cuenta. El parámetro N_max_patrones nos indica el número máximo de patrones que vamos a almacenar en la red, además como la dimensión de la red es menor se recomienda disminuir el valor de fact a 100, en caso de no hacerlo, se producirían errores en los resultados obtenidos. Estos patrones deben de estar previamente almacenados en **inicio/modelo.dat**. A no ser que añada nuevos patrones al fichero **modelo.dat**, a la hora de introducir este parámetro deberá elegir un número entre 1 y 7, sino, se producirán problemas en la ejecución. Como veremos a continuación, nos hemos centrado en el estudio de la red cuando almacenamos 2, 4 y 7 patrones almacenados, por lo tanto, se podrán encontrar en las respectivas carpetas de este número de patrones, los ficheros *.plt* para representar los datos.

Se ha seguido como criterio de estabilidad 7.3.

### 4.2.1. Dos patrones almacenados

Empezamos viendo como converge el coeficiente de solapamiento:

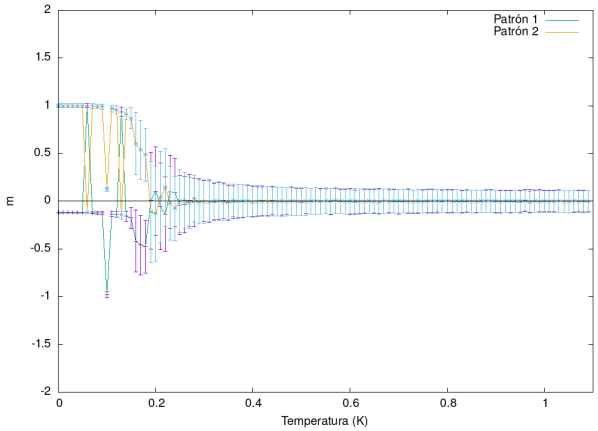
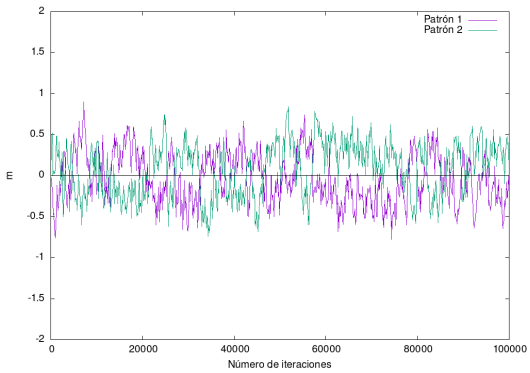


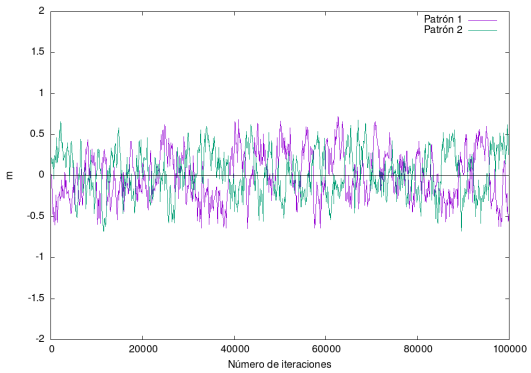

Figura 34: Coeficiente de solapamiento

Lo primero que podemos destacar es que, al igual que en el caso de un patrón, el coeficiente de solapamiento va decayendo con la temperatura, hasta llegar a un punto donde se queda oscilando en torno a 0, para ambos patrones. Por otro lado, podemos ver como para temperaturas bajas en general converge al patrón 1. Si nos fijamos en las temperaturas donde el coeficiente de solapamiento medio del sistema se hace 0, podemos ver que es superior a $T = 0{,}20$ K, dándonos una idea de por donde va a estar situada la temperatura crítica del sistema. Recurriendo a los ficheros de la evolución de la energía en función del tiempo, obtenemos una temperatura crítica de $T_c = 0{,}2600 \pm 0{,}0029$ K;

Observando como evoluciona el coeficiente de solapamiento a temperaturas en torno a la temperatura crítica para confirmar el resultado:




(a) Solapamiento a T=0,25 K




(b) Estado final a T=0,26 K

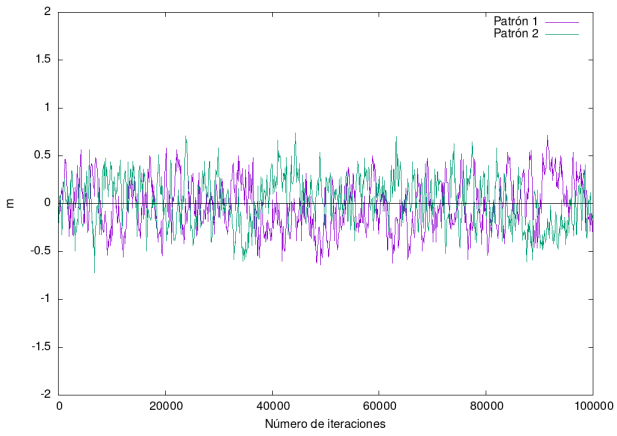



Figura 36: Estado final a T=0,27 K

Si nos fijamos a $T = 0{,}25$ K, existen pequeñas regiones donde converge el sistema a un estado estacionario, pero a partir de $T = 0{,}26$ K, no converge a ningún estado en toda la evolución.

Observando como evoluciona la energía del sistema en función de la temperatura:

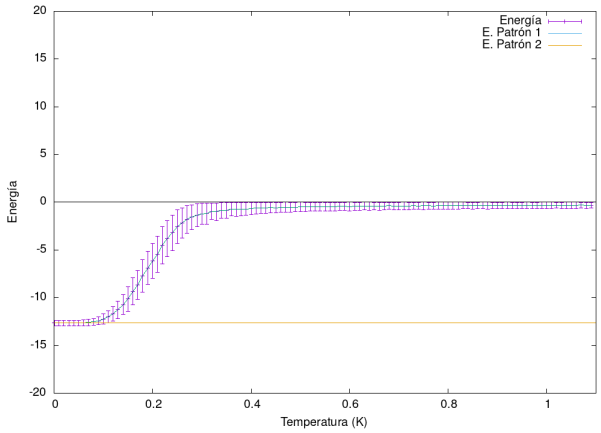



Figura 37: Energía en función de T

Como en este caso coincide que los patrones almacenados tienen la misma energía, entonces, las líneas que marcan sus energías coinciden. Como podemos ver ocurre lo mismo que en 4.1, la energía tiende a alcanzar valores más altos conforme vamos aumentando la temperatura, hasta llegar a un punto donde se queda oscilando cerca del 0.

La situación final del sistema a distintas temperaturas es:

- $T = 1{,}00 \cdot 10^{-7}$ K:

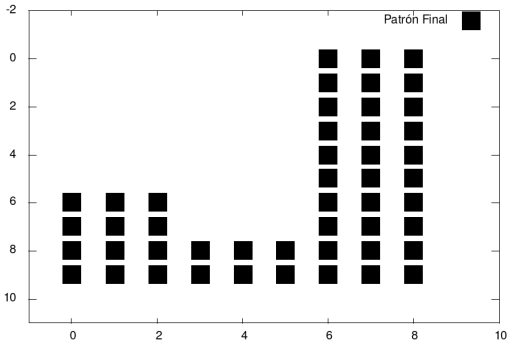



Figura 38: Estado final del sistema a $1{,}00 \cdot 10^{-7}$ K

Como podemos ver a esta temperatura, el patrón 2 se recuerda por completo.

- $T = 2{,}00 \cdot 10^{-2}$ K:

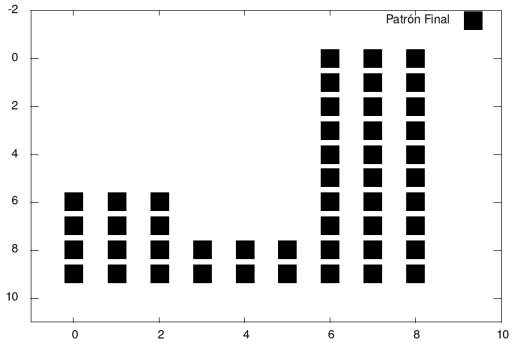



Figura 39: Estado final del sistema a $T = 2 \cdot 10^{-2}$ K

- $T = 0{,}10$ K:

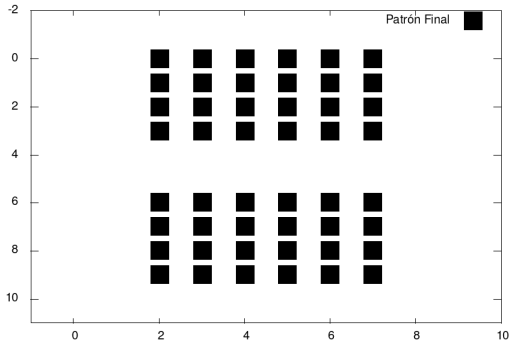



Figura 40: Estado final del sistema a $T = 0{,}10$ K

Al contrario que en las otras situaciones, en este caso vemos que converge al patrón espurio del patrón 1, por lo tanto, esta temperatura corresponderá con uno de los picos observados en 34.

- $T = 0{,}20$ K:

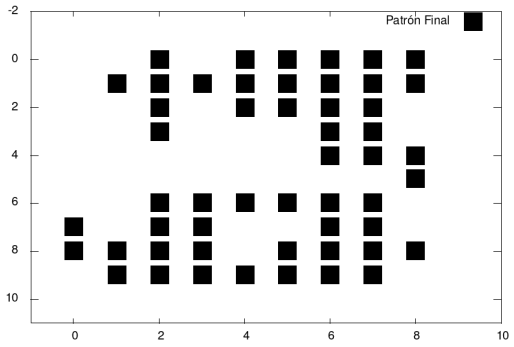



Figura 41: Estado final del sistema a $T = 0{,}20$ K

Al acercarnos a la temperatura crítica podemos ver como ya no se puede recuperar por completo la información.

- $T = 0{,}26$ K:

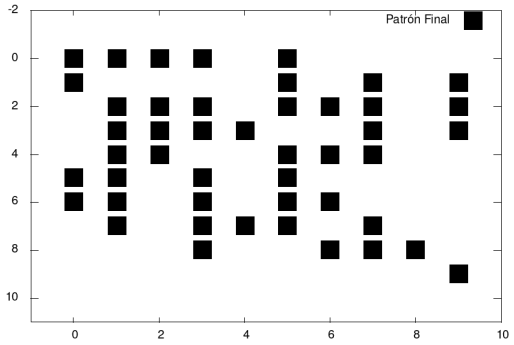
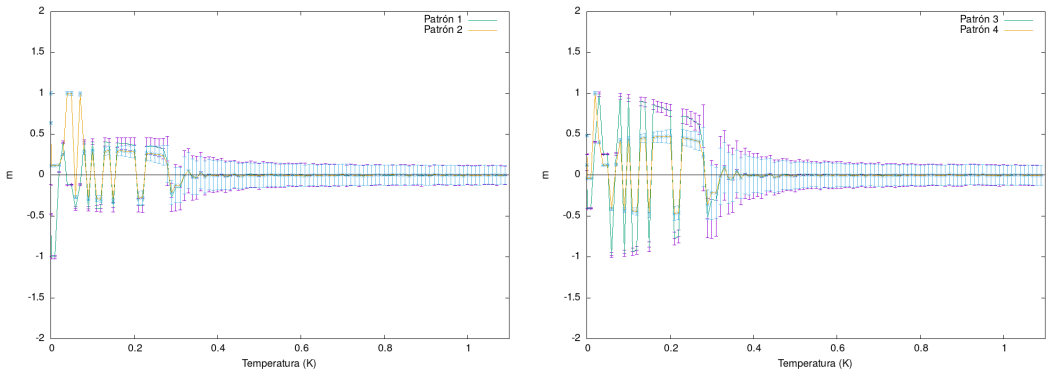



Figura 42: Estado final del sistema a $T = 0{,}26$ K

Al estar en la temperatura crítica, ya no se logra recuperar la información de los patrones almacenados.

### 4.2.2. Cuatro patrones almacenados

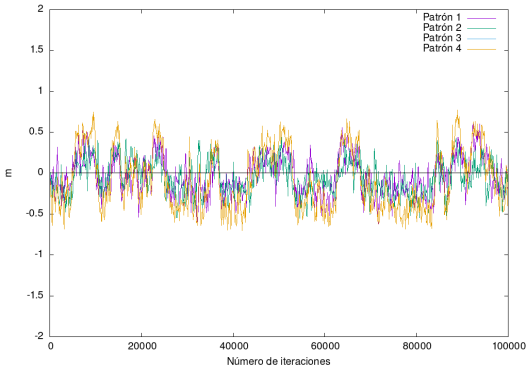
Empezamos viendo como converge el coeficiente de solapamiento:



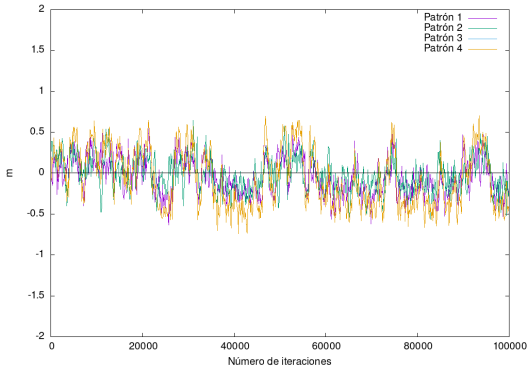

(a) Coeficiente de solapamiento (b) Coeficiente de solapamiento

Inicialmente se observa que el sistema no está convergiendo a dos patrones simultáneamente, lo que significa que el conjunto de patrones almacenados son suficientemente diferentes unos a otros. Por otro lado, al tener más estados posibles a los que converger, se puede ver como a temperaturas bajas oscila mucho más que en los otros casos que hemos visto. Otra característica que podemos ver cuando los coeficientes de solapamiento convergen totalmente a 0 en la gráfica, lo hacen a una temperatura cercana a 0,35 K, lo que significa que se ha producido un aumento de la temperatura crítica del sistema. Si al igual que antes observamos como varía el solapamiento con el número de iteraciones, podemos obtener una temperatura crítica de $T_c = 0{,}3600 \pm 0{,}0029$ K.

Si nos fijamos en como evoluciona el coeficiente de solapamiento a temperaturas en torno a la temperatura crítica para confirmar el resultado:




(a) Solapamiento a T=0,35 K




(b) Solapamiento a T=0,36 K

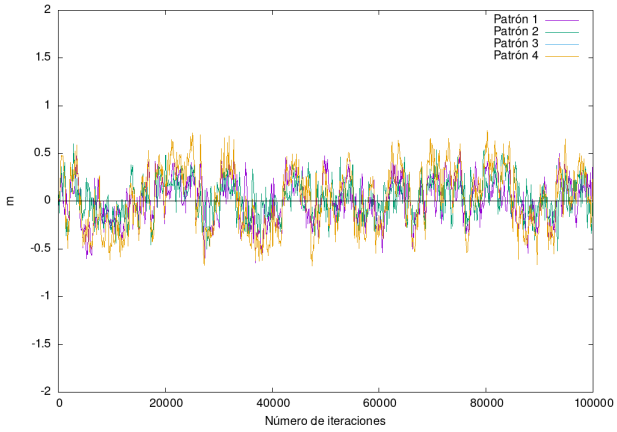



Figura 45: Solapamiento a T=0,37 K

Como podemos ver a $T = 0{,}35$ K, existen pequeñas regiones donde converge el sistema a un estado estacionario, pero a partir de $T = 0{,}36$ K, no converge a ningún estado en toda la evolución, de manera que $T_c = 0{,}3600 \pm 0{,}0029$. Además, como habíamos supuesto, se ha producido un aumento de la temperatura crítica del sistema. Esto lo podemos interpretar como que al añadir más patrones estamos adicionando más mínimos locales al sistema, haciendo que tarde más en llegar a una temperatura crítica.

Analizando como evoluciona la energía del sistema en función de la temperatura:

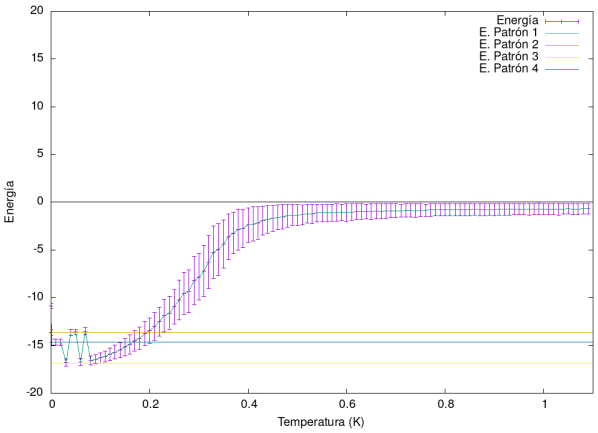



Figura 46: Energía en función de T

Observamos que a bajas temperaturas la energía converge entre los distintos mínimos correspondientes a los patrones almacenados, coincidiendo con los puntos donde el coeficiente de solapamiento

converge a un determinado patrón.

La situación final del sistema a distintas temperaturas es:

- $T = 1{,}00 \cdot 10^{-7}$ K:

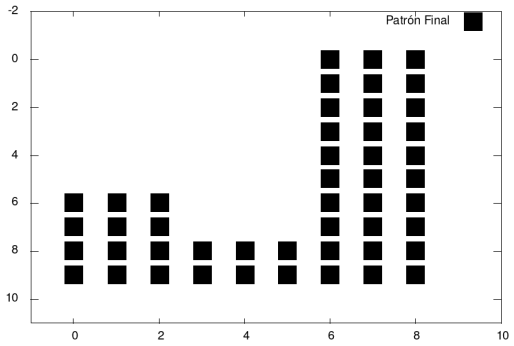



Figura 47: Estado final del sistema a $1{,}00 \cdot 10^{-7}$ K

Como vemos, a temperaturas bajas el estado final corresponde con el patrón 2.

- $T = 2{,}00 \cdot 10^{-2}$ K:

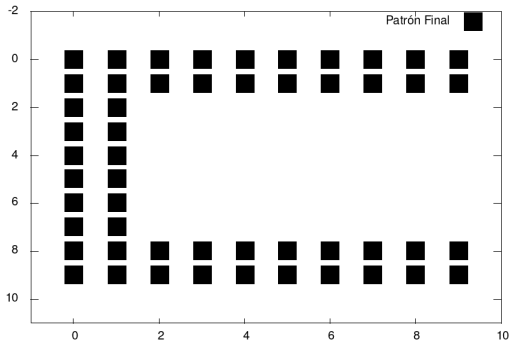



Figura 48: Estado final del sistema a $T = 2{,}00 \cdot 10^{-2}$ K

Se puede destacar que a esta temperatura se produce un cambio y el sistema converge a otro patrón distinto.

- $T = 0{,}10$ K:

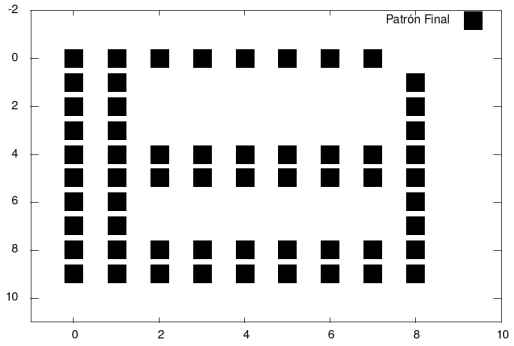



Figura 49: Estado final del sistema a $T = 0{,}10$ K

Al igual que antes, a esta temperatura se produce un cambio y el sistema converge a otro patrón distinto.

- $T = 0{,}20$ K:

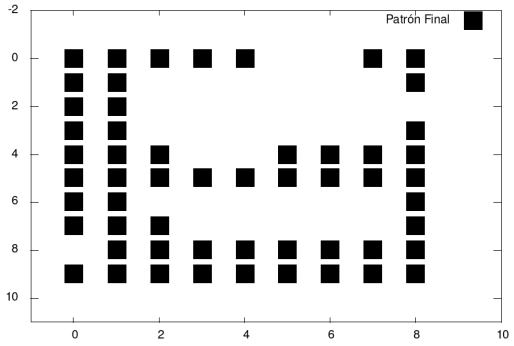



Figura 50: Estado final del sistema a $T = 0{,}20$ K

Al acercarnos a la temperatura crítica podemos ver como ya no se puede recuperar por completo la información.

- $T = 0{,}36$ K:

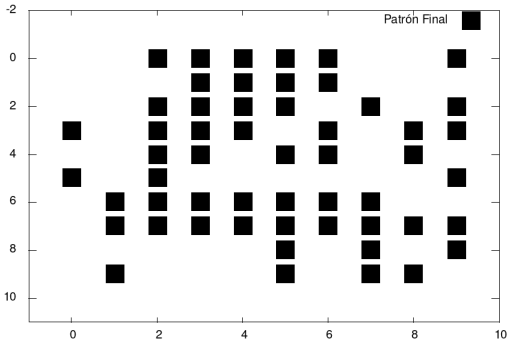
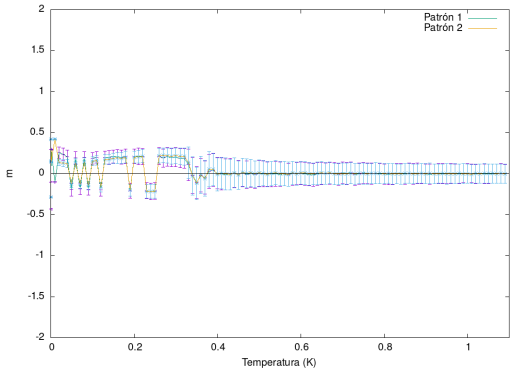



Figura 51: Estado final del sistema a $T = 0{,}36$ K

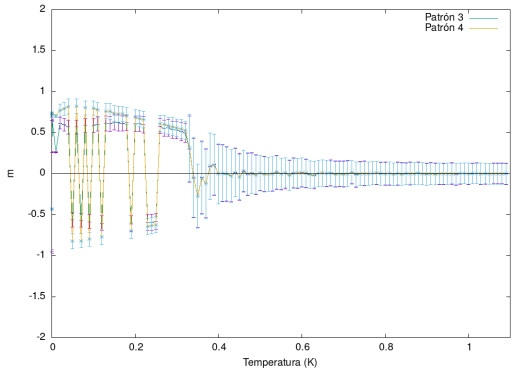
Al estar en la temperatura crítica, ya no se logra recuperar la información de los patrones almacenados.

### 4.2.3. Siete patrones almacenados

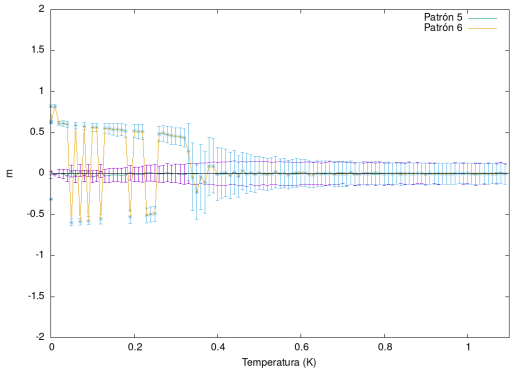
Empezamos viendo como converge el coeficiente de solapamiento:



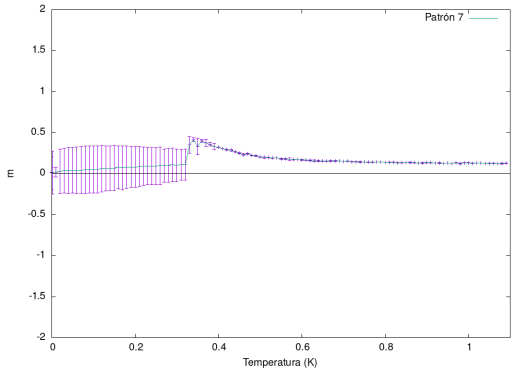

(a) Coeficiente de solapamiento




(b) Coeficiente de solapamiento

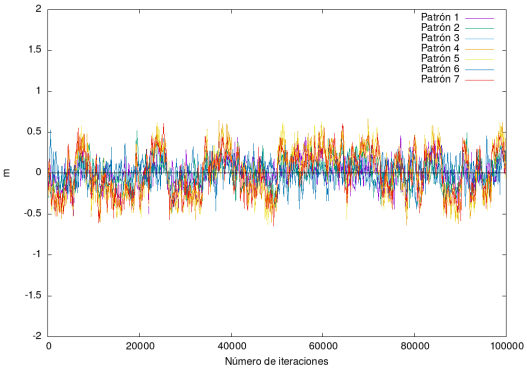



(a) Coeficiente de solapamiento

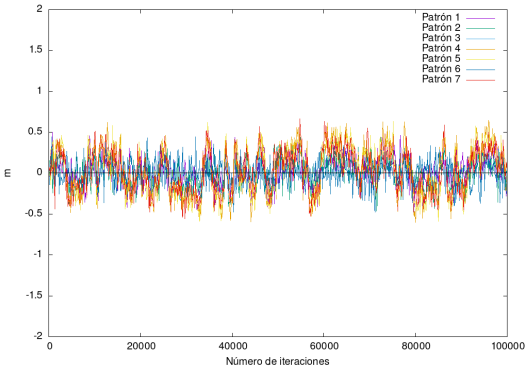



(b) Coeficiente de solapamiento

Se puede observar como evoluciona el coeficiente de solapamiento a temperaturas en torno a la temperatura crítica para confirmar el resultado:




(a) Solapamiento a T=0,44 K




(b) Solapamiento a T=0,45 K

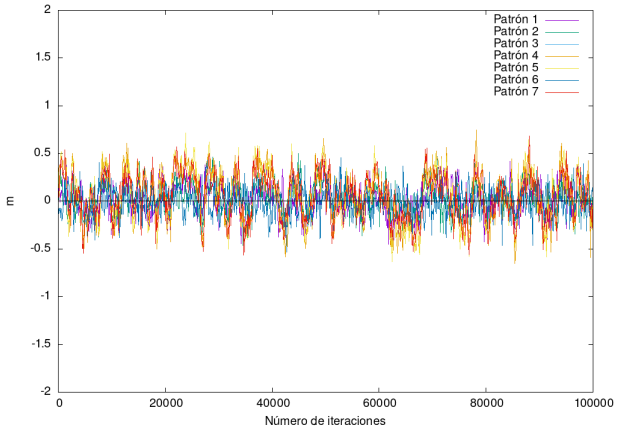



Figura 55: Solapamiento a T=0,46 K

Si nos fijamos en $T = 0{,}44$ K, existen pequeñas regiones donde converge el sistema a un estado estacionario, pero a partir de $T = 0{,}45$ K no converge a ningún estado en toda la evolución. Por lo tanto, la temperatura crítica es $T_c = 0{,}4500 \pm 0{,}0029$ K. Además, al igual que en el caso anterior, se ha producido un aumento de la temperatura crítica.

Observando como evoluciona la energía del sistema en función de la temperatura:

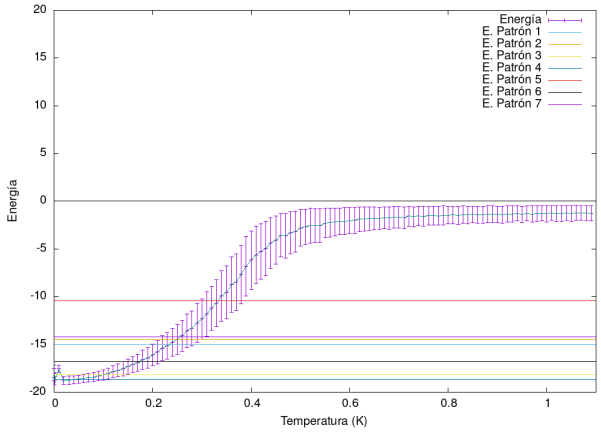



Figura 56: Energía en función de T

Como podemos ver, a bajas temperaturas, la energía converge entre los distintos patrones almacenados, coincidiendo con los puntos donde el coeficiente de solapamiento converge a un determinado patrón, aunque la energía a la que tiende el sistema principalmente es la del patrón 5. Podemos destacar que la curva de convergencia es mucho menos pronunciada que en los casos anteriores, esta es

una consecuencia del aumento de la temperatura crítica.

La situación final del sistema a distintas temperaturas es:

- $T = 1{,}00 \cdot 10^{-7}$ K:

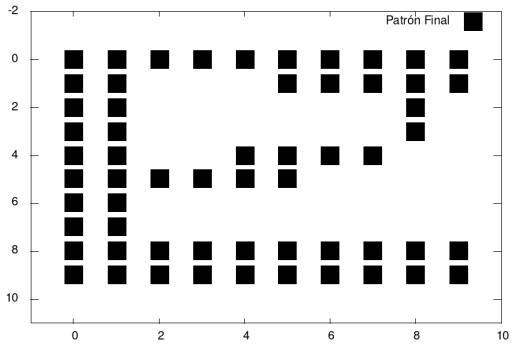



Figura 57: Estado final del sistema a $1{,}00 \cdot 10^{-7}$ K

Como podemos ver, debido a que hemos adicionado más patrones al sistema, cada vez le cuesta más a la red recuperar la información. Este fenómeno se puede deber a que los patrones no son suficientemente distintos.

- $T = 2{,}00 \cdot 10^{-2}$ K:

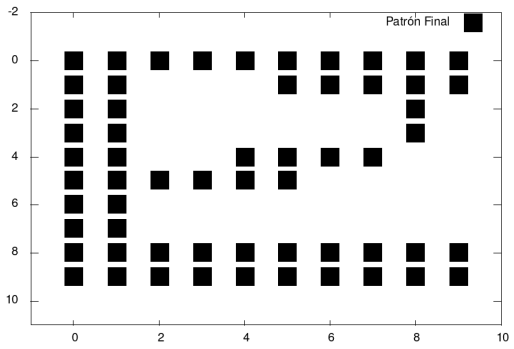



Figura 58: Estado final del sistema a $T = 2{,}00 \cdot 10^{-2}$ K

- $T = 0{,}10$ K:

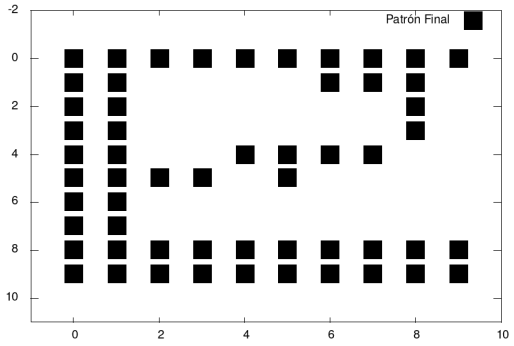



Figura 59: Estado final del sistema a $T = 0{,}10$ K

- $T = 0{,}20$ K:

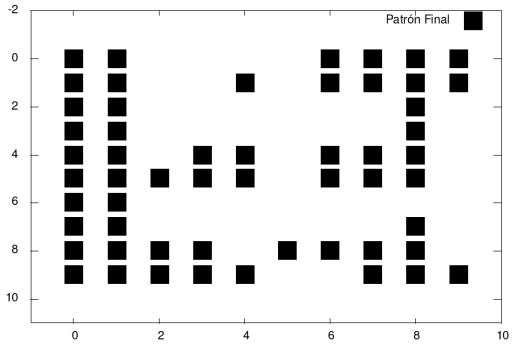



Figura 60: Estado final del sistema a $T = 0{,}20$ K

Al acercarnos a la temperatura crítica vemos como ya no se puede recuperar por completo la información.

- $T = 0{,}44$ K:

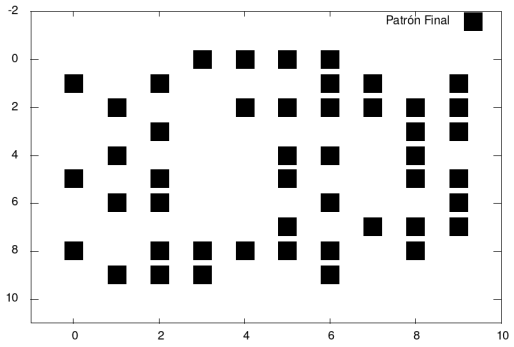



Figura 61: Estado final del sistema a $T = 0{,}44$ K

Al estar en la temperatura crítica, ya no se logra recuperar la información de los patrones almacenados.

## 4.3. Capacidad de almacenamiento de la red

Antes de empezar la discusión de los resultados obtenidos, para poder generar los datos es necesario ir a la carpeta de **codigo** y una vez dentro abra **almacenamiento_patrones**. Una vez dentro, al igual que se ha explicado en apartados anteriores, compile y ejecute el archivo **almacenamiento.cpp** o **almacenamiento_2.cpp**. Es importante que se asegure que en ese mismo directorio se encuentre ya creada las carpetas **inicio** y **resultados_por_partes**. En caso de que no se encuentren, se generan problemas a la hora de ejecutar el programa. En la carpeta de **inicio**, encontraremos los patrones que se han generado para almacenarlos en el sistema, por otro lado, en **resultados_por_partes**, encontraremos los resultados obtenidos del solapamiento para cada una de las repeticiones del programa. Para modificar los parámetros del programa se sigue el mismo procedimiento de apartados anteriores, a excepción de 2 nuevos parámetros añadidos:

- N_rep, nos indica cuantas veces queremos repetir la simulación del sistema. Hemos empleado un valor de 60 para el caso de una red de 100 neuronas, y 20 para el caso de 400.

- N_min, nos indica el número mínimo de patrones a almacenar en la red. Hemos tomado un valor de 1 para ambas situaciones.

- N_max_patrones, parámetro para indicar el número máximo de patrones a almacenar en la red. Hemos usado un valor de 120 para la red de 400 neuronas, y de 70 para la de 100.

El resto de parámetros son los mismos que se han empleado en los apartados anteriores. En esta situación el valor de la temperatura permanece constante a lo largo de todo el experimento, por lo que se aconseja coger un valor cercano a 0,00 K. En nuestro caso hemos empleado $T = 1{,}00 \cdot 10^{-5}$ K. Una vez establecido los parámetros del programa, compilado y ejecutado, los resultados obtenidos de la simulación se generarán en una archivo llamado **resultados.dat**, que se creará en el mismo directorio del código **almacenamiento.cpp**. Al igual que en los otros apartados, con **almacenamiento_2.cpp**, te pedirá que introduzca los parámetros anteriores por pantalla, el resto del proceso es similar al resto de los apartados. Como veremos a continuación, hemos estudiado en la dependencia del solapamiento para una red de 100 y 400 neuronas, por lo tanto, para generar los gráficos obtenidos será necesario ejecutar el archivo **almacenamiento_100.plt** para el caso de 100 neuronas, y **almacenamiento_400.plt** para 400 neuronas.

En ese apartado pretendemos estudiar la capacidad para recordar de la red en función del número de patrones que se hayan almacenado. La gráfica con los resultados obtenidos para el sistema de 100 neuronas:

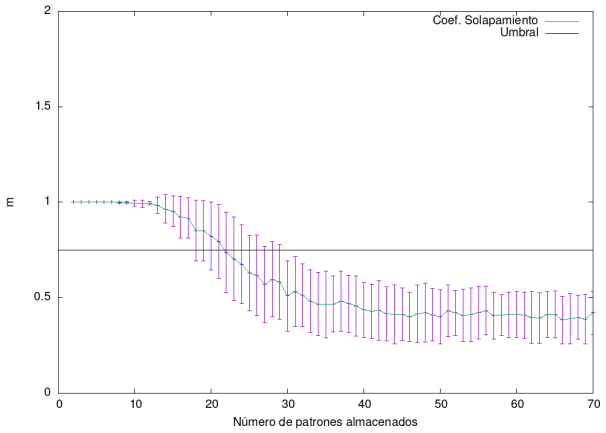



Figura 62: Almacenamiento para 100 neuronas

Como podemos ver, al igual que con la temperatura, la capacidad para recordar de la red decae con el número de patrones almacenados, hasta llegar a un punto donde no es capaz de recordar el patrón deformado. Si empleamos que $m > 0{,}75$ es el umbral a partir del cual podemos considerar que un patrón ha sido almacenado correctamente, entonces podemos determinar la fracción $\alpha = \frac{P_c}{N}$, que nos indica la fracción máxima de patrones que es capaz de almacenar la red obteniendo $\alpha = 0{,}21 \pm 0{,}01$. Por lo tanto, el número máximo de patrones que es capaz de almacenar la red en esta situación es de $21 \pm 1$ patrones. Cabe destacar que el resultado de esta gráfica es bastante sensible a las condiciones iniciales y a la semilla empleada para obtener los resultados, es por ello que al haber hecho el experimento tantas veces, los resultados obtenidos salen con una gran incertidumbre.

Si ahora analizamos los gráficos obtenidos para 400 neuronas, observamos:

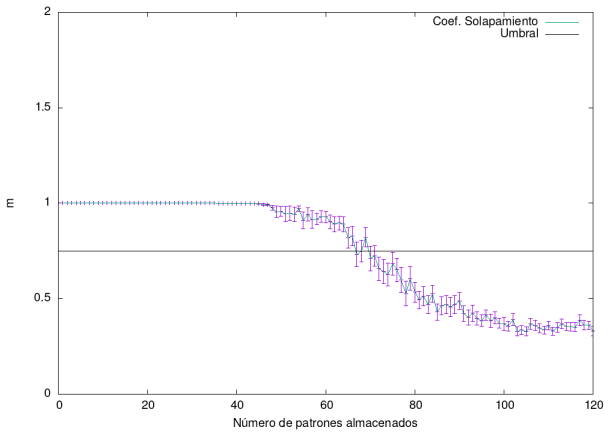



Figura 63: Almacenamiento para 400 neuronas

Lo primero que podemos destacar es que al haber tomado un menor número de repeticiones del experimento, la transición de un estado a otro es más picuda que 62. Además, las incertidumbres de los datos son mucho menores. Al igual que antes podemos ver que el solapamiento del sistema decae con el número de patrones almacenados. Si en esta situación calculamos el valor de $\alpha$ obtenemos $\alpha = 0{,}1725 \pm 0{,}0025$, es decir, el máximo número de patrones que puede almacenar la red es de $69 \pm 1$ patrones. Observamos como la fracción ha decaído con el tamaño de la red, pero el número de patrones que es capaz de almacenar ha aumentado.

## 4.4. Evolución de la red empleando una función logística

Ahora que ya sabemos como se comporta la red de Hopfield en función de la temperatura y el número de patrones almacenados, estudiaremos como evolucionará la red, si en vez de emplear la función de Metropolis para hacer evolucionar la red se emplease una función logística. Hemos obtenido esta función siguiendo un procedimiento similar al de la máquina de Boltzman, en 7.5 explicamos brevemente la deducción de esta regla. Por lo tanto, para obtener la probabilidad asociada a cada cambio de spin, emplearemos:

$$p_{i=on} = \frac{1}{1 + e^{\frac{\Delta E}{T}}} \tag{10}$$

La finalidad de este estudio es determinar la convergencia al estado estacionario, la temperatura crítica del sistema y la evolución de la energía. Para estudiar el comportamiento de la convergencia y obtener datos comparables con los obtenidos en apartados anteriores, emplearemos:

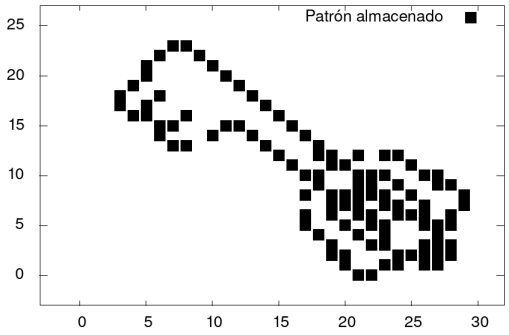



Figura 64: Patrón almacenado

Para empezar, nos centraremos en estudiar como converge cuando almacenamos un único patrón en función de la temperatura.

Para poder obtener los datos generados en esta experiencia es necesario ir a **codigo** y después a **L_1_patron_deformado** o **L_1_patron_aleatorio**. Una vez ahí, compile y ejecute el código **L_solapamiento.cpp** o **L_solapamiento_2.cpp** para modificar los parámetros del programa. Se sigue un procedimiento similar al empleado en 4.1, además puede emplear los mismos valores que recomendamos en dicho apartado.

Para descartar los datos que no pertenezcan al estado estacionario se empleará el mismo criterio seguido en 7.4. Además, como hemos observado en apartados anteriores, la condición final del sistema viene totalmente determinada por el coeficiente de solapamiento final y, como el propósito de este estudio es comparar el funcionamiento entre la función logística y la de Metropolis, de ahora en adelante me abstendré de mostrar las condiciones finales del sistema.

### 4.4.1. Patrón deformado con probabilidad 20 %

El patrón inicial que hemos introducido a la red neuronal es:

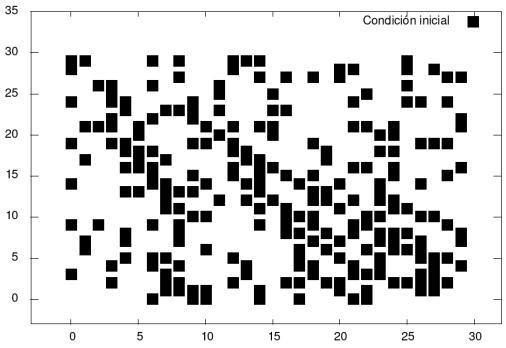



Figura 65: Condición inicial

La convergencia del solapamiento con la temperatura es la siguiente:

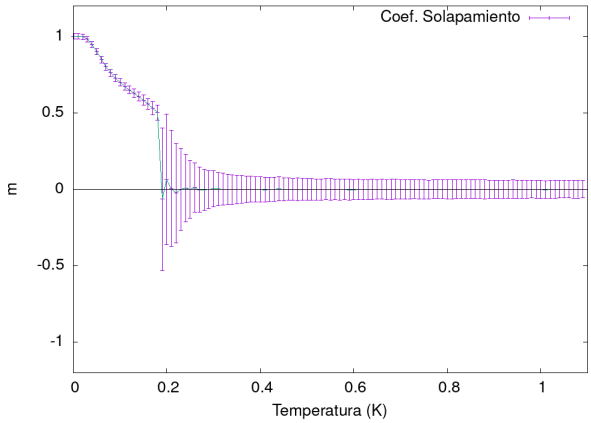



Figura 66: Coeficiente de solapamiento en función de la temperatura

Como podemos ver, tenemos un comportamiento similar que en el caso de la función de Metropolis, la red converge a bajas temperaturas, pero conforme vamos aumentando desaparece la convergencia.

Podemos comparar la gráfica del coeficiente de solapamiento en función de la temperatura superpuesta con la obtenida en 4.1.2, obteniendo:

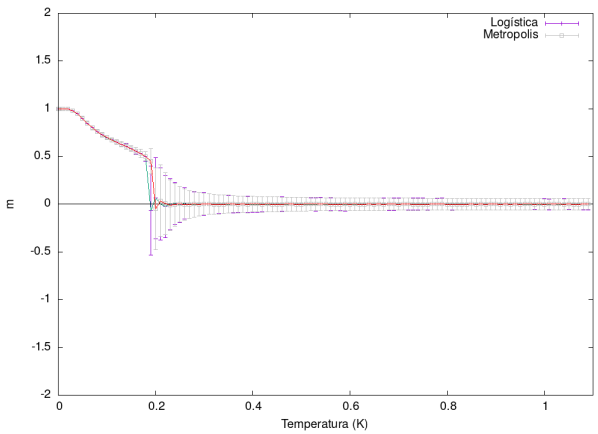
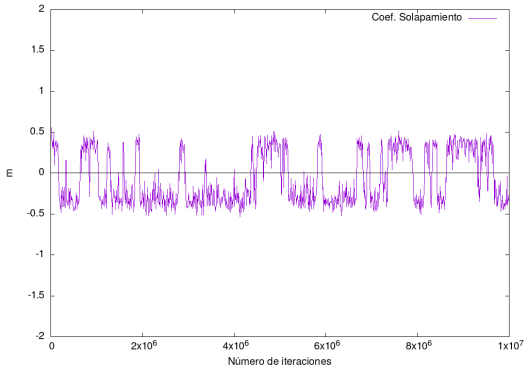


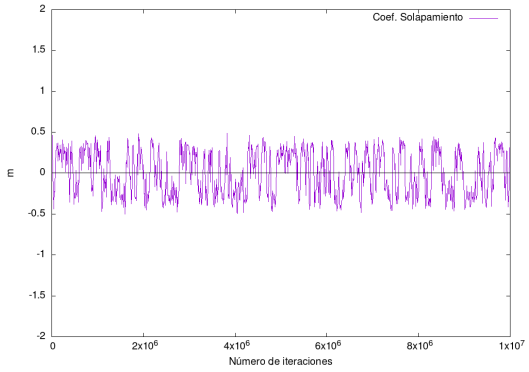

Figura 67: Comparación del coeficiente de solapamiento en función de la temperatura

Si nos fijamos, las curvas obtenidas por ambos métodos se solapan casi perfectamente a todas las temperaturas, aun así, si nos fijamos en la temperatura de 0,20 K, observamos como la función sigmoide converge a 0 ligeramente antes que la de Metropolis.

Si observamos la evolución del coeficiente de solapamiento en función del número de iteraciones, podemos determinar la temperatura crítica del sistema. Usando como referencia T=0,20 K por ser el punto donde podemos observar un mayor descenso del coeficiente de solapamiento, podemos obtener:




(a) Solapamiento a T=0,22 K




(b) Solapamiento a T=0,23 K

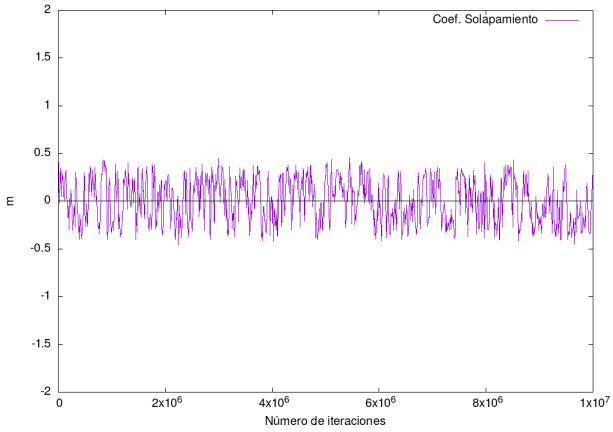



Figura 69: Solapamiento a T=0,24 K

Obtenemos una temperatura crítica de $T_c = 0{,}2400 \pm 0{,}0029$ K es la temperatura a la que el sistema deja de converger a un estado estacionario, obteniendo un resultado igual al apartado anterior.

Observando la gráfica de la energía del sistema en función de la temperatura:

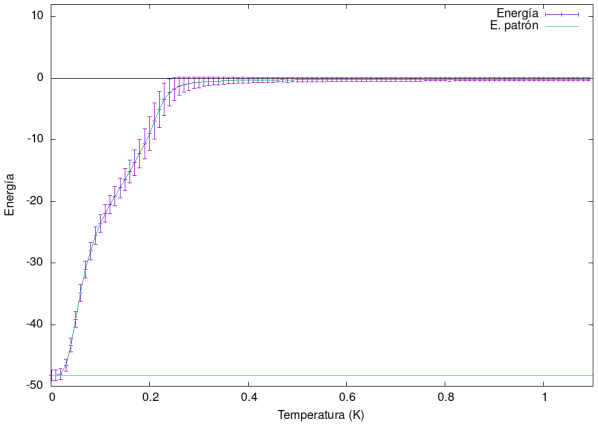



Figura 70: Energía en función de la temperatura

Al igual que antes, podemos mostrar la gráfica superpuesta con los resultados del apartado anterior:

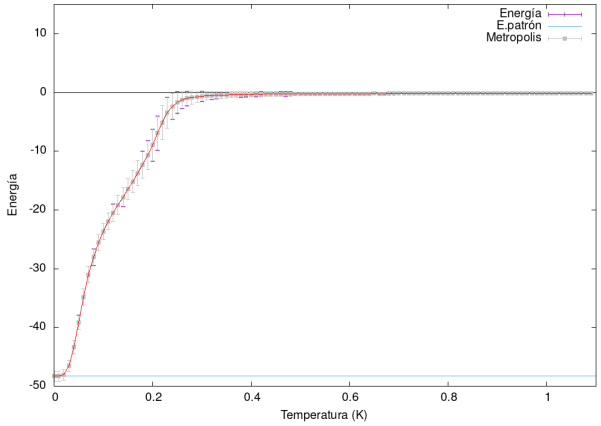



Figura 71: Comparación de la energía en función de la temperatura

Es destacable que obtenemos una superposición total de los datos obtenidos por ambos métodos.

### 4.4.2. Patrón deformado con probabilidad 50 %

El patrón inicial que hemos introducido a la red es:

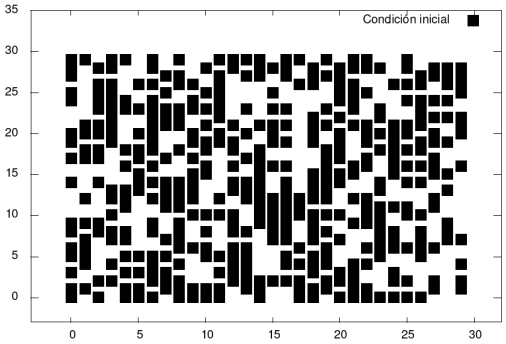



Figura 72: Condición inicial

La convergencia del solapamiento con la temperatura superpuesta con la obtenida en 4.1.2:

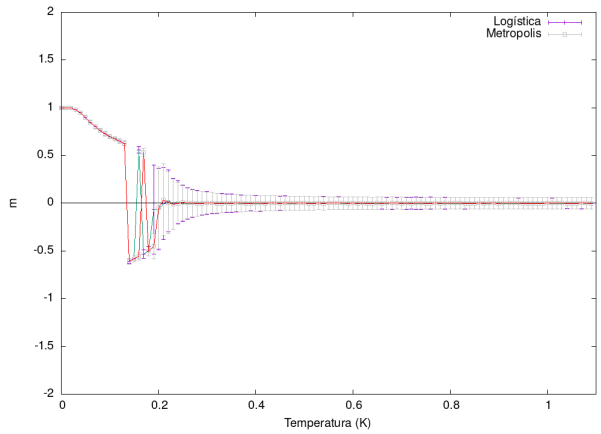
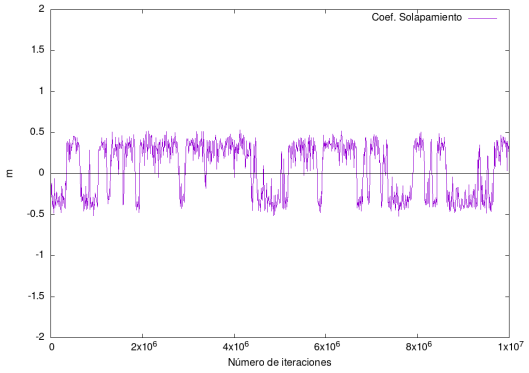
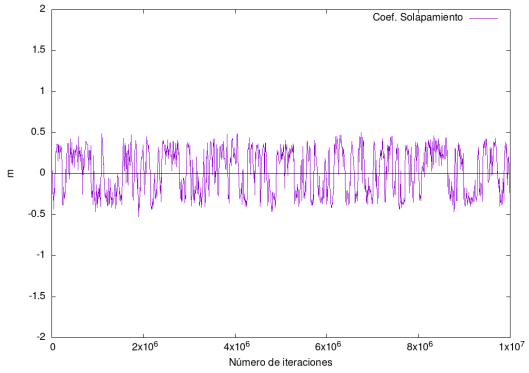



Figura 73: Superposición de coeficientes de solapamiento

Si determinamos la temperatura crítica:




(a) Solapamiento a T=0,22 K




(b) Solapamiento a T=0,23 K

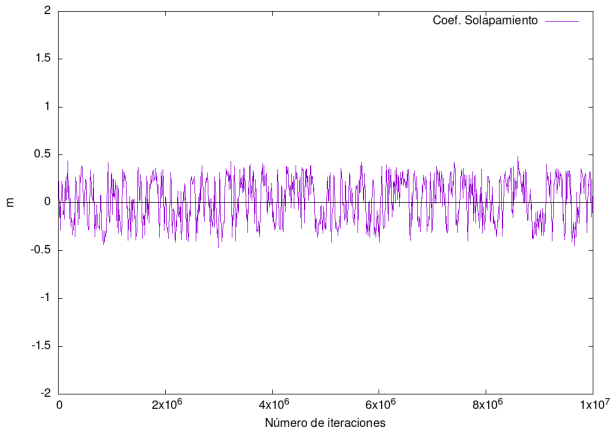



Figura 75: Solapamiento a T=0,24 K

Como podemos observar, $T_c = 0{,}2400 \pm 0{,}0029$ K es la temperatura a la que el sistema deja de converger a un estado estacionario, obteniendo un resultado igual al apartado anterior.

Si observamos la evolución de la energía:

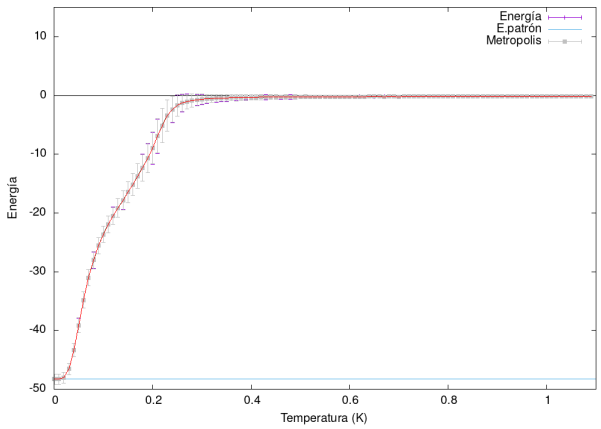



Figura 76: Comparación de la energía en fucnión de la temperatura

Podemos observar como obtenemos una superposición total de los datos obtenidos por ambos métodos.

### 4.4.3. Patrón deformado con probabilidad 70 %

El patrón inicial que hemos introducido a la red neuronal es:

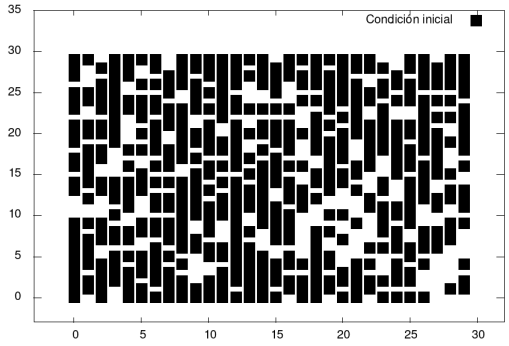



Figura 77: Condición inicial

La convergencia del solapamiento con la temperatura superpuesta con la obtenida en 4.1.4:

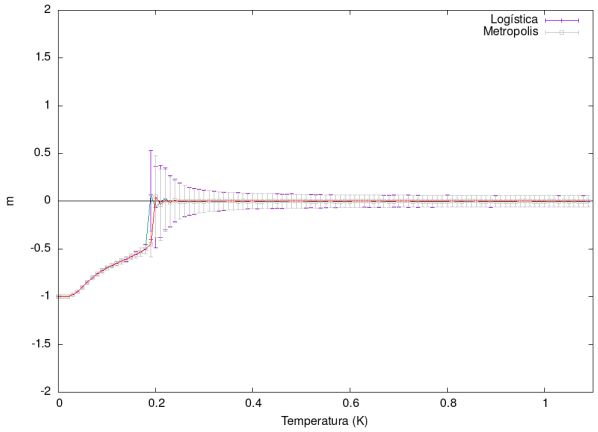
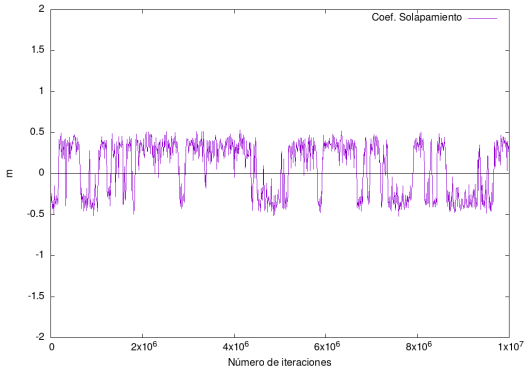


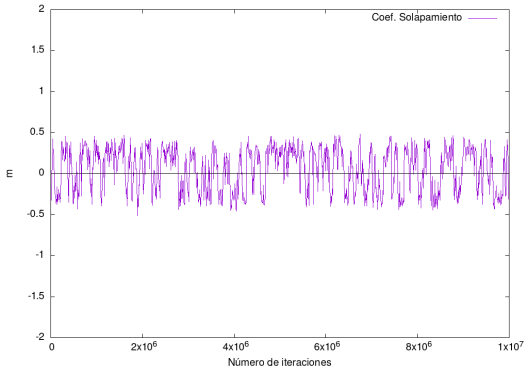

Figura 78: Superposición de coeficientes de solapamiento

Podemos observar el mismo comportamiento entre el método de la función logística y el de la función de Metropolis que en las dos experiencias anteriores.

Si determinamos la temperatura crítica:




(a) Solapamiento a T=0,22 K




(b) Solapamiento a T=0,23 K

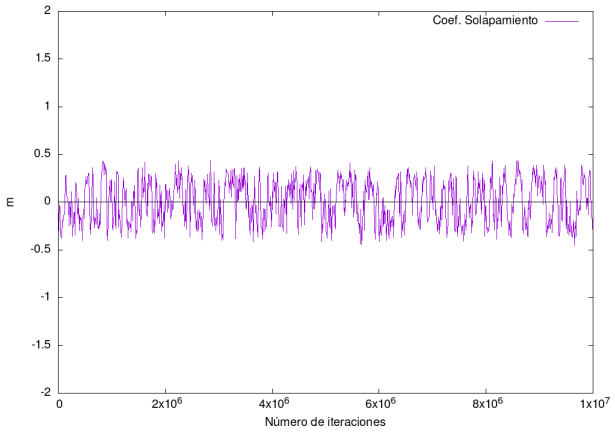



Figura 80: Solapamiento a T=0,24 K

Se obtiene una temperatura crítica de $T_c = 0{,}2400 \pm 0{,}0029$ K.

Si observamos la evolución de la energía:

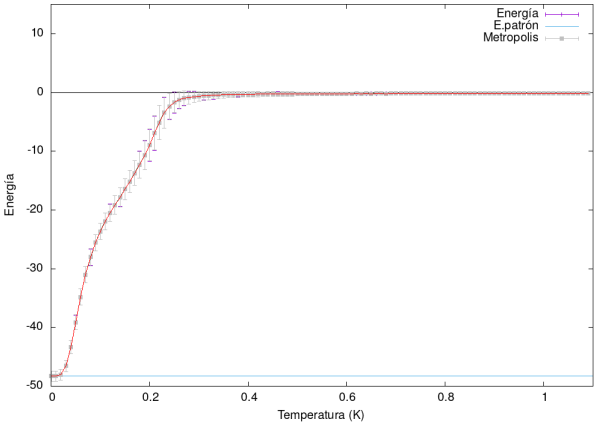



Figura 81: Comparación de la energía en función de la temperatura

### 4.4.4. Patrón aleatorio

El patrón inicial que hemos introducido a la red neuronal es:

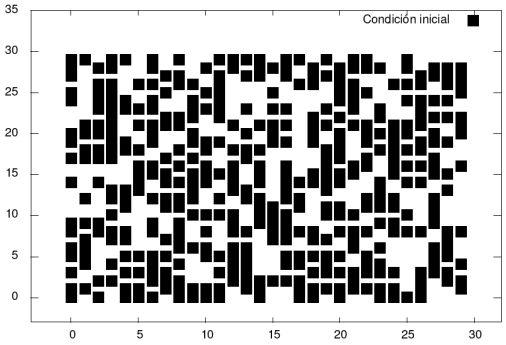



Figura 82: Condición inicial

La convergencia del solapamiento con la temperatura superpuesta con la obtenida en 4.1.1:

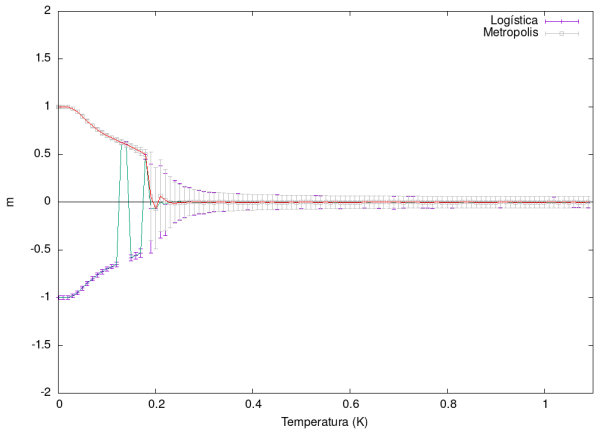
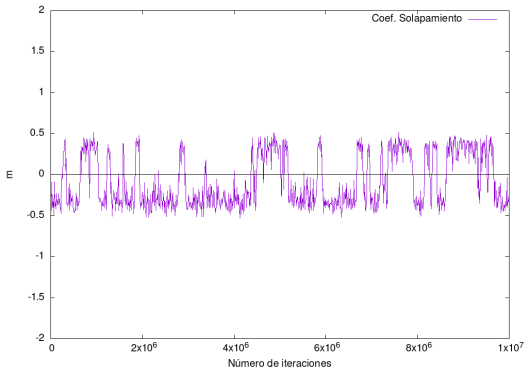
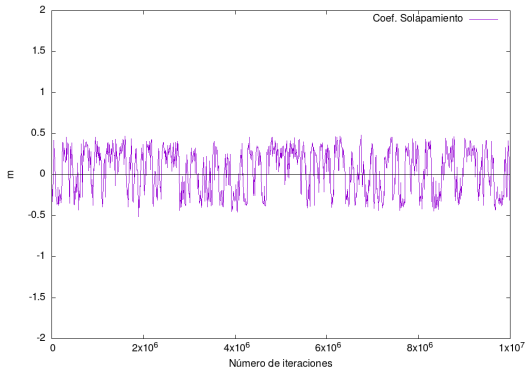



Figura 83: Superposición de coeficientes de solapamiento

Si determinamos la temperatura crítica:




(a) Solapamiento a T=0,22 K




(b) Solapamiento a T=0,23 K

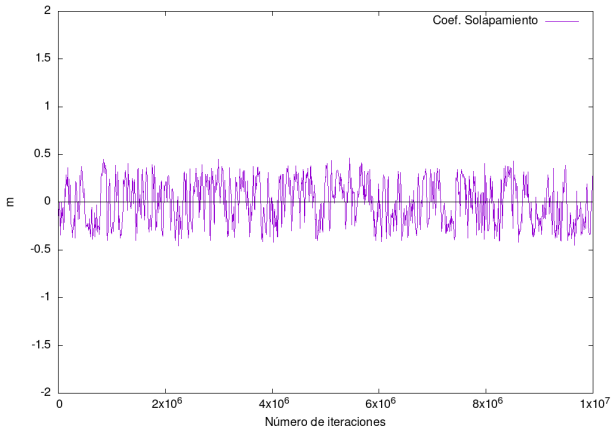



Figura 85: Solapamiento a T=0,24 K

Obtenemos de nuevo una temperatura crítica de $T_c = 0{,}2400 \pm 0{,}0029$ K.

Si observamos la evolución de la energía:

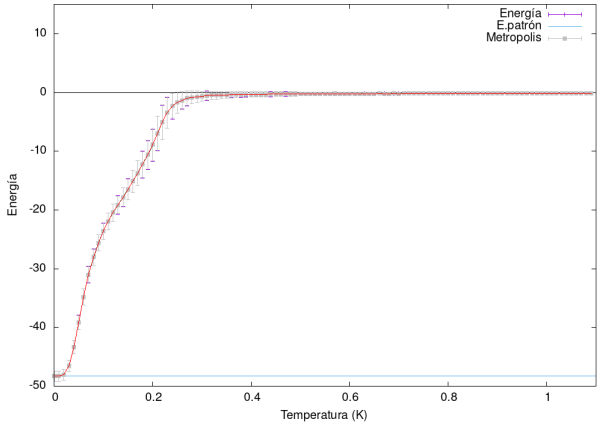



Figura 86: Comparación de la energía en fucnión de la temperatura

## 4.5. Evolución mediante la función logística para varios patrones

Este apartado podría considerarse una continuación del apartado anterior, ya que vamos a estudiar como evoluciona el coeficiente de solapamiento para el caso en el que almacenemos varios patrones en la red. Para ello llevaremos a cabo el estudio empleando los patrones usados en 4.2, a partir de lo cual podremos comparar los datos obtenidos por ambos métodos. Como hemos visto antes, a la hora de hacer evolucionar la red, se comporta como el método de la función de Metropolis, por lo tanto, emplearemos el mismo número de pasos que en 4.2 para obtener los resultados.

La condición inicial introducida al sistema es:

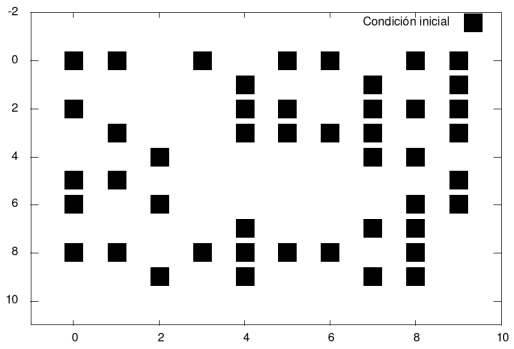



Figura 87: Condición inicial

Para poder repetir la experiencia vaya a la carpeta **codigo** y después a **L_varios_patrones**. Una vez en ella, compile y ejecute el código **varios_patrones.cpp** o **varios_patrones_2.cpp**. Para modificar los parámetros de programa se sigue el mismo procedimiento que en 4.2. Además, puede emplear los mismos valores que recomendamos en dicho apartado.

### 4.5.1. Dos patrones almacenados

Empezamos viendo como converge el coeficiente de solapamiento:

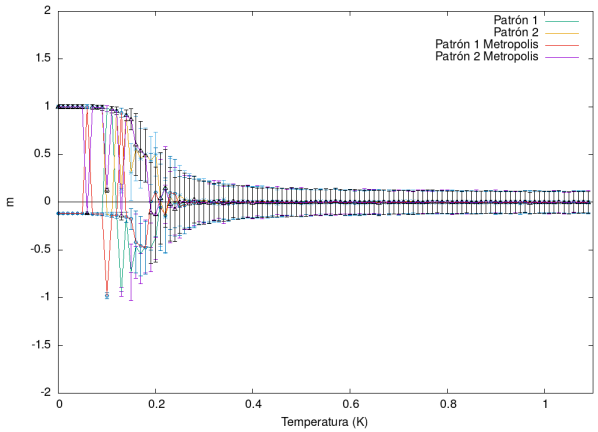



Figura 88: Coeficiente de solapamiento

Podemos mostrar los resultados de la gráfica sin las incertidumbres para una mejor visualización:

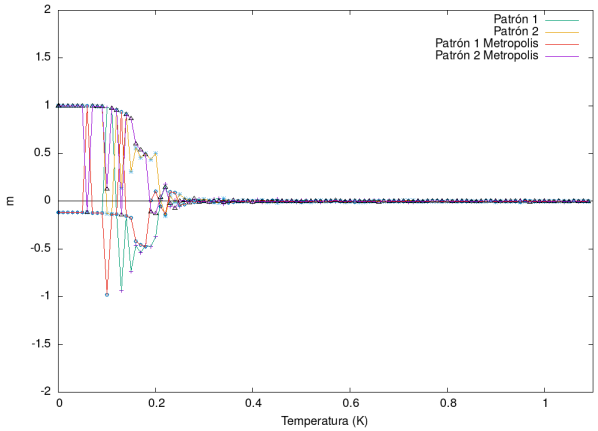
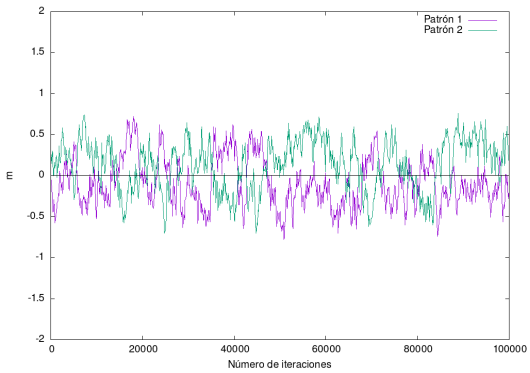


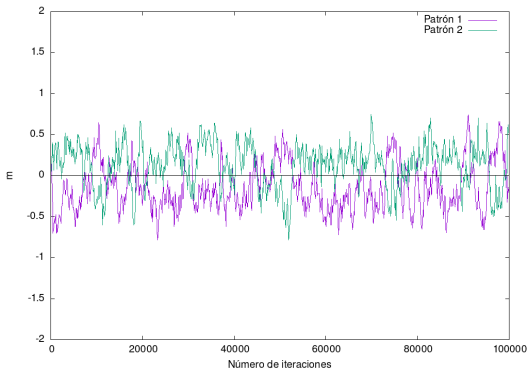

Figura 89: Coeficiente de solapamiento

Conforme añadimos patrones podemos observar más diferencias en la convergencia entre las dos funciones, fijándonos donde hay zonas en las que cada método converge a un patrón distinto. Si nos fijamos, en las temperaturas donde el coeficiente de solapamiento medio del sistema se hace 0, podemos ver que es superior a $T = 0{,}20$ K, dándonos una idea de por donde va a estar situada la temperatura crítica del sistema. Recurriendo a los ficheros de la evolución de la energía en función del tiempo, podemos obtener una temperatura crítica de $T_c = 0{,}2600 \pm 0{,}0029$ K.

Nos fijamos en la evolución del coeficiente de solapamiento a temperaturas cercanas a la crítica:




(a) Solapamiento a T=0,25 K




(b) Solapamiento a T=0,26 K

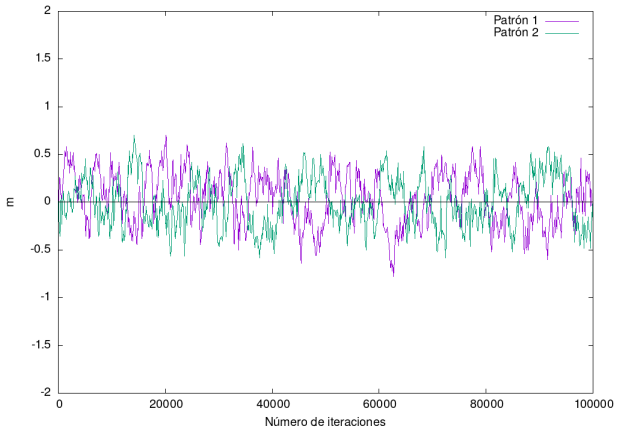



Figura 91: Solapamiento a T=0,27 K

Podemos ver que a partir de $T = 0{,}26$ K no converge en ningún momento a un estado estacionario.

Observando la evolución de la energía del sistema en función de la temperatura:

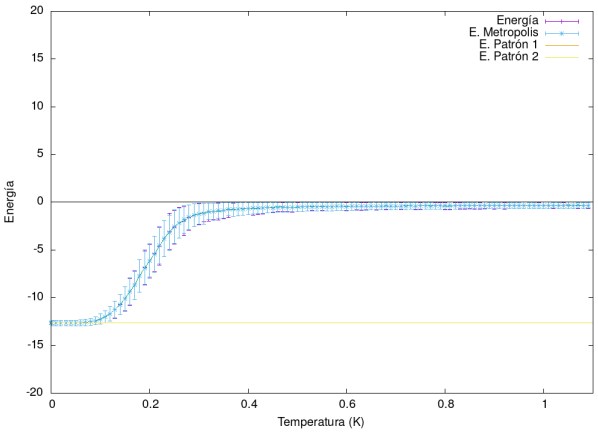
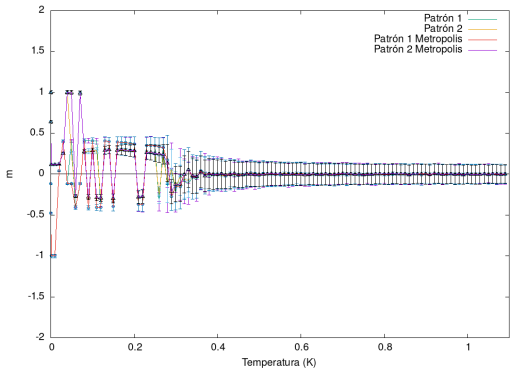



Figura 92: Energía en función de T

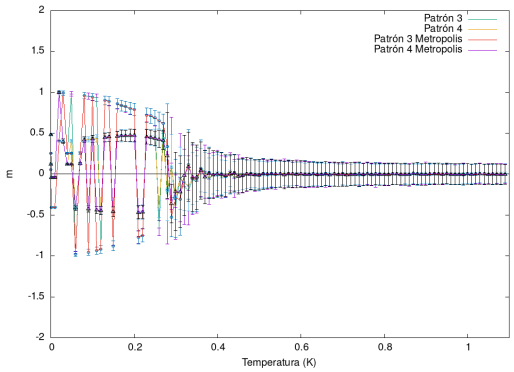
Podemos observar que la convergencia de la energía es idéntica en ambos métodos.

### 4.5.2. Cuatro patrones almacenados

Empezamos viendo como converge el coeficiente de solapamiento. He dividido el gráfico en dos partes para comparar de forma más clara los resultados:

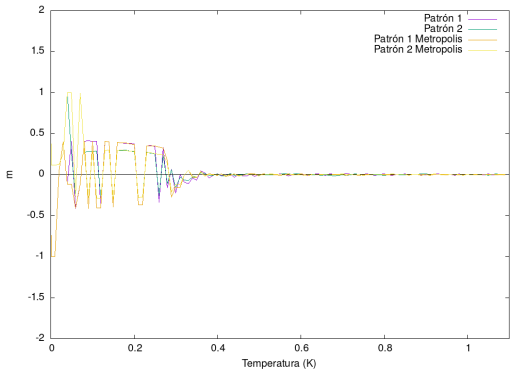



(a) Coeficiente de solapamiento

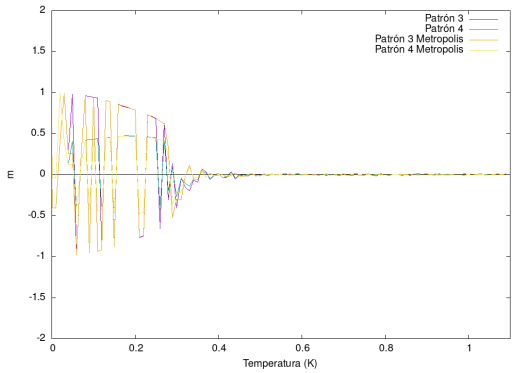



(b) Coeficiente de solapamiento

Sí mostramos de nuevo los resultados sin las incertidumbres:



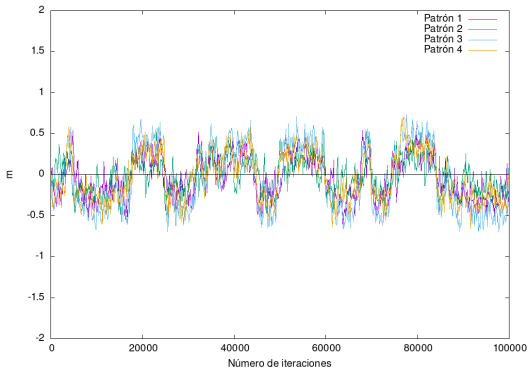

(a) Coeficiente de solapamiento



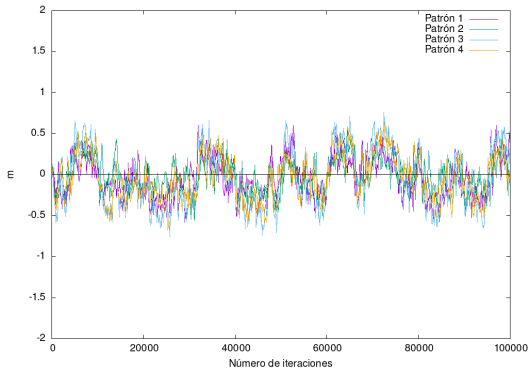

(b) Coeficiente de solapamiento

Como podemos ver, el número de diferencias entre el coeficiente de solapamiento de ambos sistemas, ha aumentado en comparación al caso anterior. Podemos destacar como los patrones convergen al 0 del coeficiente de solapamiento al mismo tiempo, algo que no ocurría en los casos anteriores. Recurriendo a los ficheros de la evolución de la energía en función del tiempo, podemos obtener una temperatura crítica de $T_c = 0{,}3600 \pm 0{,}0029$ K.

Podemos observar como evoluciona el coeficiente de solapamiento a temperaturas en torno a la temperatura crítica para confirmar el resultado:




(a) Solapamiento a T=0,35 K




(b) Solapamiento a T=0,36 K

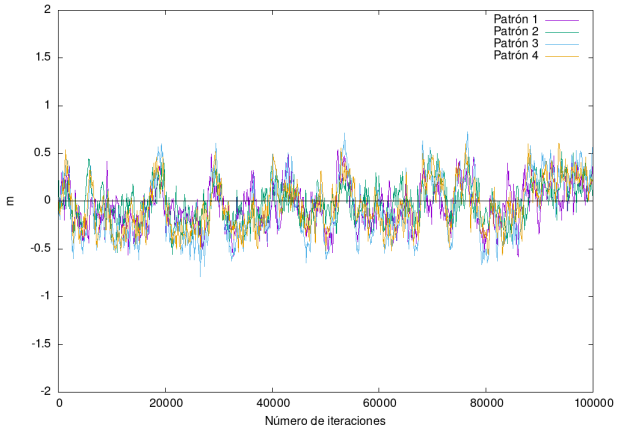



Figura 96: Solapamiento a T=0,37 K

Como podemos ver a partir de $T = 0{,}36$ K no converge en ningún momento a un estado estacionario.

Si analizamos como evoluciona la energía del sistema en función de la temperatura:

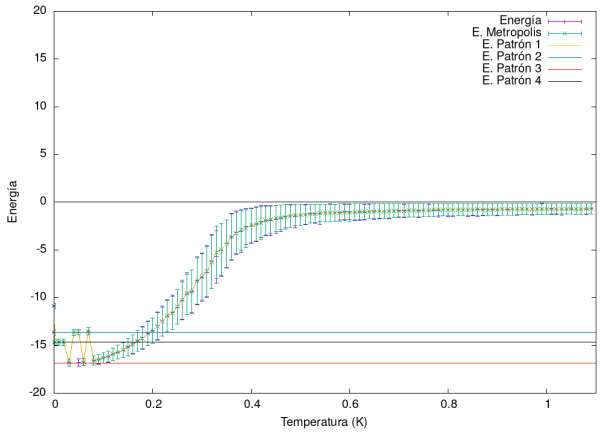
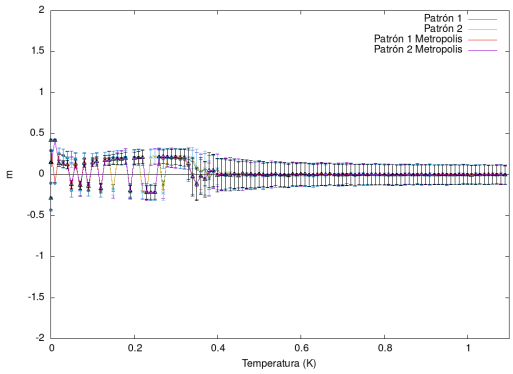



Figura 97: Energía en función de T

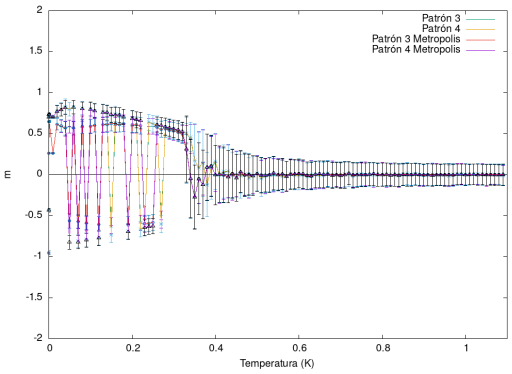
Observamos que a temperaturas bajas, la única diferencia observable es el patrón al que converge el sistema, algo que ya se podía esperar habiendo visto la gráfica del coeficiente de solapamiento en función de la temperatura. Conforme va aumentando la temperatura, esas diferencias desaparecen, y convergen a 0 a la misma velocidad. Podemos ver al igual que en 4.2, la pendiente de la curva de la energía se va suavizando con el número de patrones.

### 4.5.3. Siete patrones almacenados

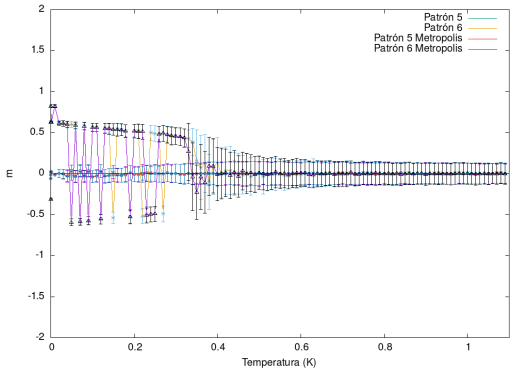
Empezamos viendo como converge el coeficiente de solapamiento. Para tener una mayor legibilidad, al igual que en el apartado anterior, vamos a dividirlo en varios gráficos.



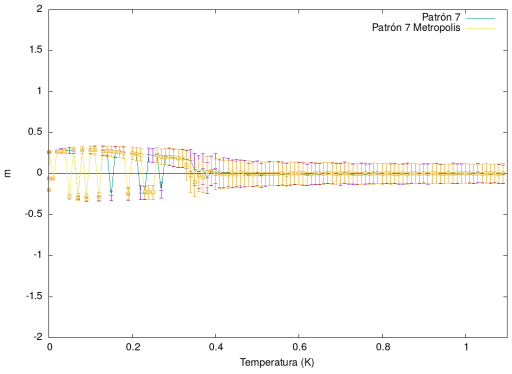

(a) Coeficiente de solapamiento




(b) Coeficiente de solapamiento

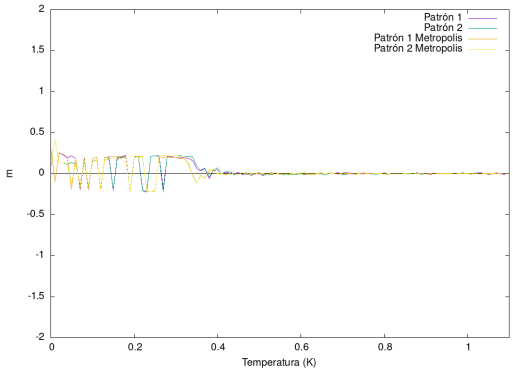



(a) Coeficiente de solapamiento

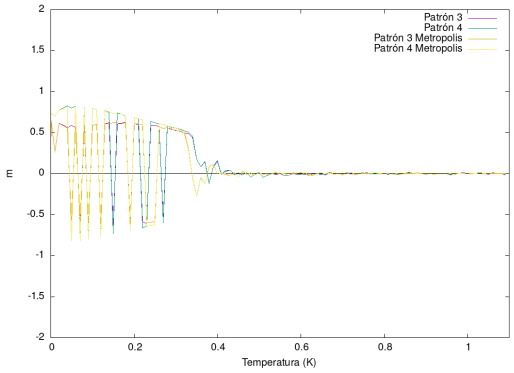



(b) Coeficiente de solapamiento

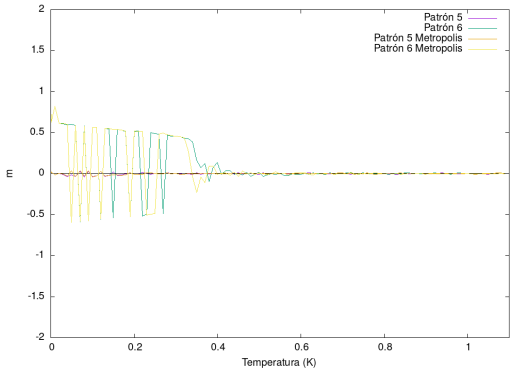
Si representamos sin las incertidumbres:



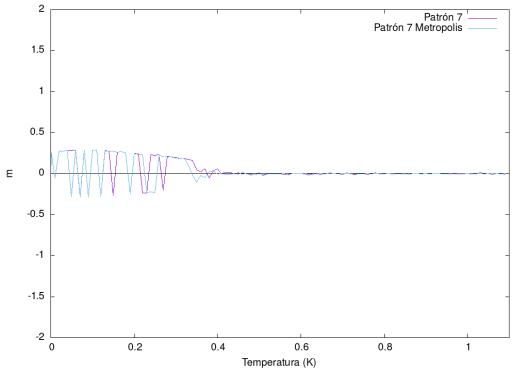

(a) Coeficiente de solapamiento




(b) Coeficiente de solapamiento



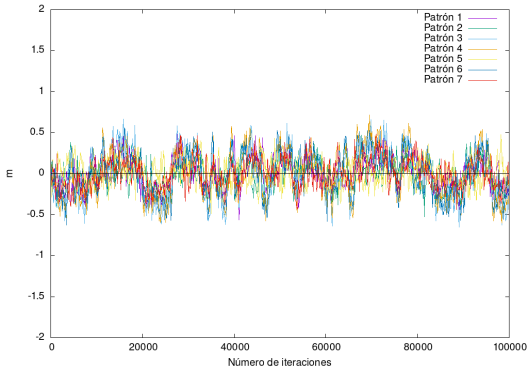

(a) Coeficiente de solapamiento



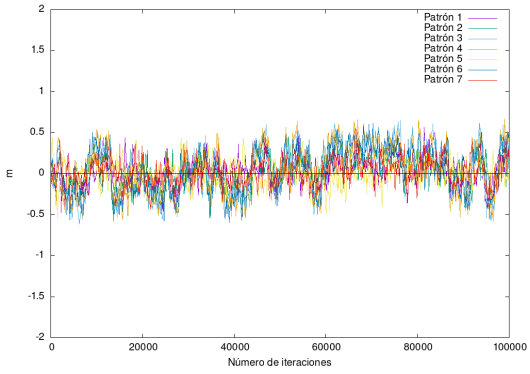

(b) Coeficiente de solapamiento

Conforme aumentan el número de patrones, el coeficiente de solapamiento tarda más en converger en las dos funciones. Recurriendo a los ficheros de la evolución del coeficiente de solapamiento en función del tiempo, podemos obtener una temperatura crítica de $T_c = 0{,}4500 \pm 0{,}0029$ K.

Para confirmar el resultado observaremos la evolución del coeficiente de solapamiento a temperaturas cercanas a la crítica:




(a) Solapamiento a T=0,44 K




(b) Solapamiento a T=0,43 K

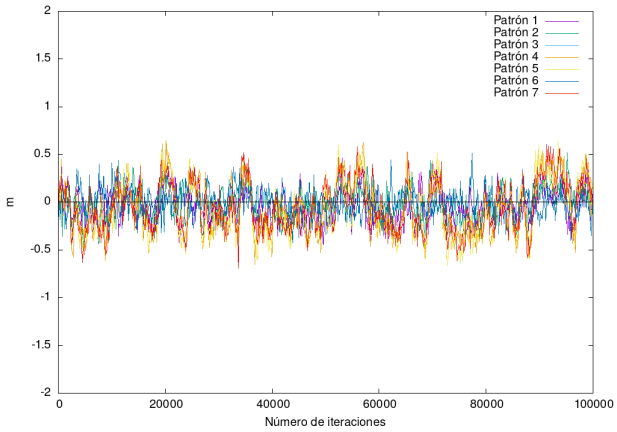



Figura 103: Solapamiento a T=0,46 K

A $T = 0{,}44$ K existen pequeñas regiones donde converge el sistema a un estado estacionario, pero a partir de $T = 0{,}46$ K no converge a ningún estado estacionario en ningún momento.

Analizando la evolución de la energía del sistema en función de la temperatura:

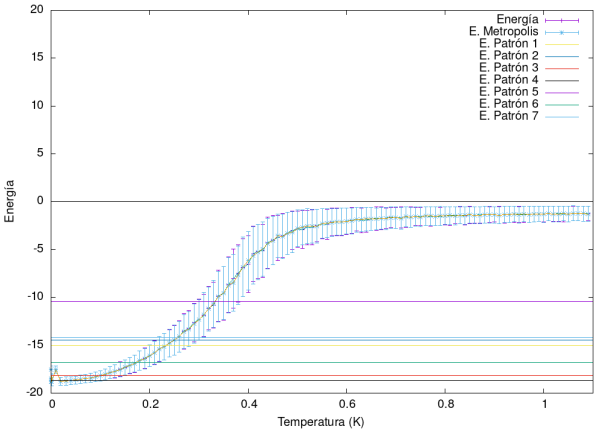



Figura 104: Energía en función de T

Observando esta gráfica notamos que el aumento de la temperatura crítica ha producido que la curva de convergencia de la energía se suavice.

## 4.6. Capacidad de almacenamiento de la red empleando la función logística

Antes de empezar a discutir los datos obtenidos del sistema, para poder generar los datos es necesario ir a la carpeta de **codigo** y abra **L_almacenamiento_patrones**. Al igual que se ha explicado en apartados anteriores, compile el archivo **almacenamiento.cpp** o **almacenamiento_2.cpp** y ejecútelo para obtener los resultados. El funcionamiento del programa es similar al explicado en 4.3.

Si exponemos los resultados obtenidos para el sistema de 100 neuronas, obtenemos la siguiente gráfica:

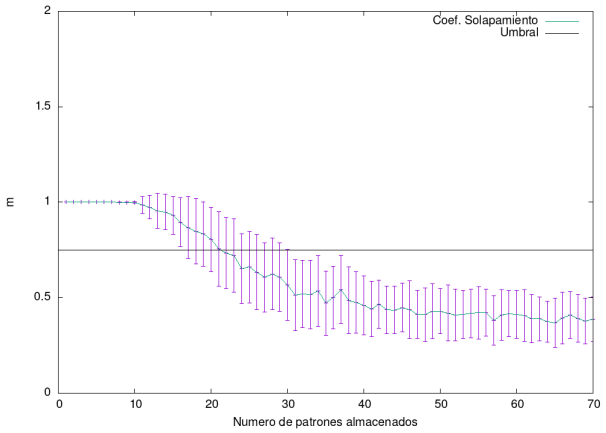



Figura 105: Almacenamiento para 100 neuronas

Si nos fijamos, al igual que antes la capacidad de recordar de la red va disminuyendo con el número de patrones almacenados. Si volvemos a calcular la fracción $\alpha$, obtenemos un resultado de $\alpha = 0{,}21 \pm 0{,}01$, es decir, el máximo de patrones que se pueden almacenar en la red recuperando la información es de $21 \pm 1$ patrones. Podemos destacar que es el mismo resultado obtenido para la función de Metropolis.

Si ahora analizamos los gráficos obtenidos para 400 neuronas, observamos lo siguiente:

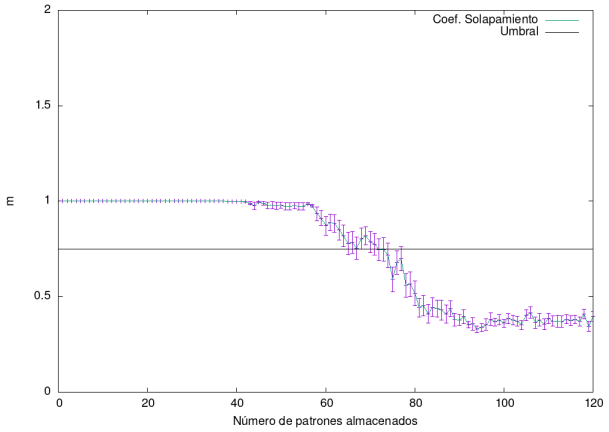



Figura 106: Almacenamiento para 400 neuronas

Al igual que antes podemos obtener un valor de $\alpha = 0{,}1775 \pm 0{,}0025$. Este resultado es ligeramente mayor al del apartado anterior, pudiendo almacenar $71 \pm 1$ patrones distintos en la red.

# 5. Conclusión

A partir de este estudio podemos llegar a las siguientes conclusiones:

- A partir de los resultados obtenidos en 4.1 hemos podido observar que el coeficiente tiende a 0 al aumentar la temperatura. Por lo tanto, a temperaturas altas, el sistema no es capaz de converger a un estado almacenado.

  Además, hemos podido observar como el patrón almacenado en la red se corresponde con un mínimo de energía. Conforme se va aumentando la temperatura del sistema, la energía tomará valores correspondientes a los estados con configuraciones aleatorias, es decir, los estados correspondientes a cualquier valor de energía. También, si nos fijamos en la gráfica de la energía de 4.1.4 donde la convergencia que predomina es al patrón espurio, podemos ver que el resultado

de la curva es el mismo que en los apartados anteriores. De esta forma, hemos podido demostrar experimentalmente que la energía del patrón espurio es un mínimo local que toma el mismo valor que el del patrón almacenado.

Por último, hemos comprobado la existencia de una temperatura crítica a partir de la cual el sistema es incapaz de converger a un estado estacionario. Como hemos obtenido el mismo valor de la temperatura crítica para distintas condiciones iniciales, queda demostrado que el valor de esta no depende del patrón inicial introducido en la red.

- Sí observamos los resultados obtenidos en 4.2, podemos concluir que el coeficiente de solapamiento va a tender a 0 independientemente del número de patrones que tengamos almacenados. Observando como varía la energía en los distintos casos, podemos ver como la curva cada vez converge más lentamente a 0, lo cual nos indica que el valor de la temperatura crítica del sistema va variando conforme añadimos más patrones. Podemos interpretar esta variación de la temperatura crítica del sistema como el efecto que produce la adición de patrones almacenados al sistema provocando que el pozo de energía sea mayor. Por lo tanto, la adición de patrones hace que el sistema sea más estable al aumento de la temperatura.

  Por otro lado, hemos observado que el aumento de la temperatura crítica implica que se podrá recuperar la información completa o parcialmente a mayores temperaturas. Esto lo hemos analizado en el estado final del sistema, fijándonos en el estado a $T = 0,20$ K cuando almacenamos 4 patrones y cuando almacenamos 7 patrones. En el caso donde tenemos un mayor número de patrones almacenados se ha podido recuperar una mayor información del patrón distorsionado.

  Por último, hemos podido observar que la convergencia del coeficiente de solapamiento a uno de los patrones almacenados no solo depende de la distancia en norma a cada uno de los mínimos locales de energía, sino que también depende de la temperatura.

- A partir de los resultados obtenidos en 4.3, hemos concluido que la capacidad de recuperar información en la red decae con el número de patrones almacenados, llegando a un punto donde la red es incapaz de recordar. También hemos observado que el número máximo de patrones que se pueden almacenar en la red, siendo posible recuperar la información del sistema, depende del tamaño de la red, por lo tanto, al tener un sistema de mayor dimensión, se podrá almacenar un mayor número de patrones.

  Los gráficos obtenidos del coeficiente de solapamiento en función del número de patrones son curvas decrecientes, que en la parte final de esta tienden a tomar valores que anulan su pendiente. Si la gráfica obtenida hubiera presentado alguna discontinuidad se trataría de un cambio de fase de $1^{\text{o}}$ orden, pero como vemos, no se da esa situación, sino que la convergencia es continua hasta alcanzar el nuevo estado. Esto significa que estamos frente a un cambio de fase de $2^{\text{o}}$ orden.

  Además, hemos podido observar que la incertidumbre de los datos decae con el tamaño del sistema, siendo la desviación de la red de 400 neuronas considerablemente menor.

- A partir de las gráficas del solapamiento en el apartado 4.4 podemos comprobar que no solo la función logística es capaz de converger a un estado estacionario a distintas temperaturas, sino que lo hace a una velocidad similar al algoritmo que implementa la función de Metropolis. Si nos fijamos en los resultados obtenidos a lo largo de esta experiencia, los resultados obtenidos en general, tanto para la función sigmoide como la de Metropolis, son similares.

  Observando los datos, podemos ver que, en general, a bajas temperaturas se comporta de la misma forma que la función de Metropolis. Sin embargo, conforme vamos aumentando la temperatura, aparecen diferencias en la convergencia al patrón almacenado, convergiendo en algunos caso al espurio. Para observar la velocidad de convergencia de cada función, debemos recurrir

a la energía del sistema. Como hemos visto, en los 4 casos las dos gráficas se superponen perfectamente, por lo tanto, la velocidad de convergencia a uno de los patrones almacenados es la misma con ambas funciones.

Si nos fijamos en los valores que hemos obtenido de $T_c$ en esta situación, podemos ver que son los mismos que en 4.1. Este resultado reafirma la hipótesis que nos habíamos planteado previamente sobre si la temperatura crítica es una propiedad intrínseca del sistema o no. Por lo tanto, a partir de los resultados podemos concluir que $T_c$ no depende de la función que hayamos empleado para hacer evolucionar el sistema.

También podemos destacar, a la vista de los resultados obtenidos en este apartado, que la función logística favorece el comportamiento emergente de la red de Hopfield.

- Por otro lado, si recurrimos a los resultados obtenidos en 4.5, podemos observar que el comportamiento de la red es similar al obtenido en 4.2. Si nos fijamos en la evolución del coeficiente de solapamiento en función de la temperatura, podemos ver que hay regiones donde converge a patrones distintos que en el caso de la función de Metropolis, pero a la hora de converger a 0 el coeficiente de solapamiento, convergen en ambos casos a una velocidad similar.

  A partir de los resultados obtenidos de la temperatura crítica, vemos que toma los mismos valores que los obtenidos en 4.2, de manera que podemos asegurar que el valor de $T_c$ no depende de la función empleada para hacer evolucionar la red. Además, al igual que antes, hemos podido observar que el valor de la temperatura crítica aumenta con el número de patrones almacenados, por lo tanto, este aumento será un comportamiento intrínseco del sistema y no tiene relación con la función empleada para hacer evolucionar la red.

  La energía del sistema a temperaturas elevadas será el mismo en ambos casos, más precisamente, las únicas diferencias que podemos encontrar en la energía de los dos sistemas es a bajas temperaturas, donde varía dependiendo del patrón al que converja.

- Por último, si observamos 4.6, podemos ver que al igual que en el caso de la función de Metropolis, la capacidad para recordar del sistema decae con la temperatura, hasta llegar a un punto donde el sistema es incapaz de recuperar la información. Los resultados obtenidos en este apartado son los mismos que para el caso de la función de Metropolis, a excepción de la fracción de solapamiento para la red de 400 neuronas, pero si observamos las Gaussianas correspondientes obtenemos:

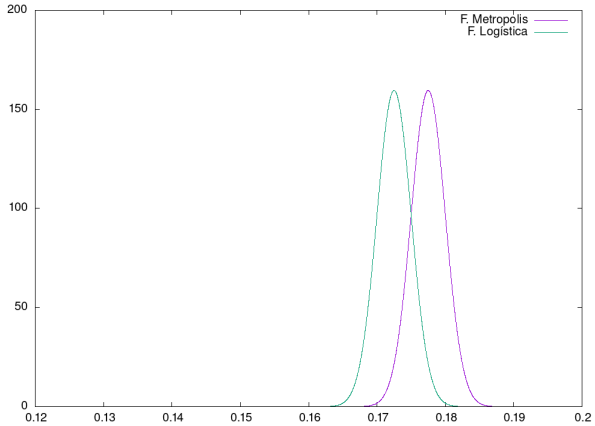



Figura 107: Gaussianas

Si nos fijamos en la gráfica, existe una región donde las gaussianas se solapan, por lo tanto,

podemos considerar equivalentes a los resultados obtenidos.

Además, podemos destacar que la incertidumbre de los datos decae con el tamaño del sistema, siendo la desviación de la red de 400 neuronas, considerablemente menor.

A partir de estos resultados, podemos considerar que empleando el método de las Cadenas de Markov tanto para la función de Metropolis o logística, dará lugar a la aparición de una red neuronal de Hopfield.

Como hemos visto a lo largo del experimento, cuando trabajamos a bajas temperaturas, el sistema, siempre que esté trabajando dentro del límite de patrones que se pueden almacenar, converge a un mínimo de energía. Por lo tanto, hemos demostrado que este modelo matemático resulta útil para resolver problemas de optimización en los que a la solución le corresponda un mínimo de energía. También la red de Hopfield tiene una gran capacidad para recordar los patrones almacenados a partir de una condición inicial deformada, en consecuencia, podemos aplicar este modelo para situaciones donde sea necesario eliminar el ruido de señales obtenidas de experimentos, problemas de reconocimiento de voz o faciales.

# 6. Agradecimientos

Me gustaría agradecer a mi mejor amigo Luis Herrero Díaz y mi padre Miguel Ángel Gallardo Escobar, por su apoyo y consejos a la hora de realizar este informe.

# 7. Apéndices

## 7.1. Optimización del cálculo de diferencias de energía

A la hora de ejecutar la simulación, si cada vez que calculamos $\Delta H = H(\mathbf{s}') - H(\mathbf{s})$ resolvemos el hamiltoniano correspondiente al vector $\mathbf{s}$ y $\mathbf{s}'$, sería muy costoso computacionalmente. Por ello, vamos a desarrollar la expresión de la diferencia del hamiltoniano para el caso particular donde los vectores se diferencian en una única componente, con la intención de reducir los tiempos de ejecución.

Para ello partimos de dos vectores $\mathbf{s}'$, $\mathbf{s}$ que verifican:

$$s'_i = s_j, \forall i = j \neq k$$
$$s'_k \neq s_k$$

Por lo tanto si recurrimos a la expresión de $\Delta H$:

$$\Delta H = -\frac{1}{2}\sum_{i,j} w_{ij} s'_i s'_j + \sum_i \theta_i s'_i + \frac{1}{2}\sum_{i,j} w_{ij} s_i s_j - \sum_i \theta_i s_i$$

Podemos reescribir:

$$\Delta H = -\frac{1}{2}\sum_{i,j} \left(w_{ij} s'_i s'_j - w_{ij} s_i s_j\right) + \sum_i \left(\theta_i s'_i - \theta_i s_i\right)$$

Ahora empleamos que los vectores se diferencian en una única componente, de manera:

$$\Delta H = -\frac{1}{2}\left(\sum_i \left(w_{ik} s_i (s'_k - s_k)\right) + \sum_j \left(w_{kj} s'_j (s'_k - s_k)\right)\right) + \left(\theta_k (s'_k - s_k)\right)$$

Por último, empleamos que la matriz de pesos es simétrica, de forma que cumple $w_{ij} = w_{ji}$, de manera:

$$\Delta H = -\sum_i \left(w_{ik} s_i (s'_k - s_k)\right) + \theta_k (s'_k - s_k)$$

Si estudiamos la complejidad de cada algoritmo para ver cuál es más eficiente, obtenemos que en el caso de la situación inicial, al desarrollar la expresión de la diferencia de energía, se comportaría con una complejidad de aproximadamente $O\left(n^2\right)$. En cambio, una vez desarrollado para el caso particular, la complejidad del algoritmo pasa a $O(n)$. Por lo tanto, comparando obtenemos:

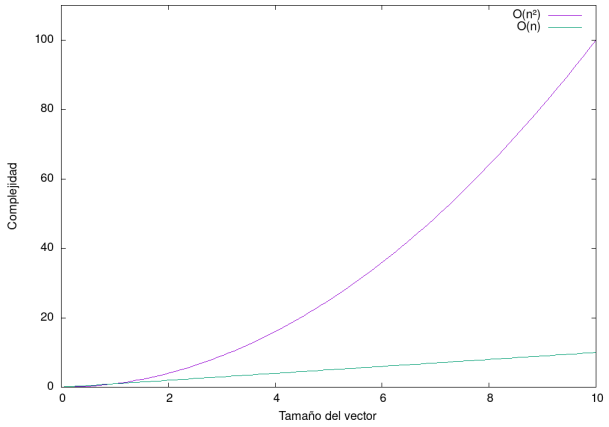



Figura 108: Órdenes de complejidad para la diferencia de energía

Podemos ver fácilmente que con tan solo n=10, el número de iteraciones necesarias para el caso inicial es mucho mayor que el de la expresión que hemos deducido.

## 7.2. Energía de los patrones espurios

En esta sección nos preguntamos sobre cuál será la relación entre la matriz de pesos del patrón original, y el patrón complementario (el patrón que se obtiene a cambiar cada spin por su contrario). Durante la demostración emplearemos la siguiente notación $v$ para un elemento del patrón original, y $\widetilde{v}$, para un elemento del complementario. Partimos de la ecuación de los pesos del patrón original:

$$w_{ij} = \frac{1}{N} \sum_{\mu=1}^{P} \frac{\left(\xi_i^\mu - a\right)\left(\xi_j^\mu - a\right)}{a(1-a)}$$

Es trivial que $a(1-a) = \widetilde{a}(1-\widetilde{a})$, además, como empleamos una notación de 0 y 1, existe las siguientes relaciones entre las componentes del patrón original y el complementario, $\widetilde{a} = 1 - a$ y $\widetilde{\xi_i^\mu} = 1 - \xi_i^\mu$. Empleamos estas propiedades:

$$w_{ij} = \frac{1}{N} \sum_{\mu=1}^{P} \frac{\left(\xi_i^\mu - a\right)\left(\xi_j^\mu - a\right)}{\widetilde{a}(1-\widetilde{a})}$$

$$w_{ij} = \frac{1}{N} \sum_{\mu=1}^{P} \frac{\left(\xi_i^\mu - 1 + 1 - a\right)\left(\xi_j^\mu - 1 + 1 - a\right)}{\widetilde{a}(1-\widetilde{a})}$$

$$w_{ij} = \frac{1}{N} \sum_{\mu=1}^{P} \frac{\left(\xi_i^\mu - 1 + \widetilde{a}\right)\left(\xi_j^\mu - 1 + \widetilde{a}\right)}{\widetilde{a}(1-\widetilde{a})}$$

$$w_{ij} = \frac{1}{N} \sum_{\mu=1}^{P} \frac{\left(1 - \xi_i^\mu - \widetilde{a}\right)\left(1 - \xi_j^\mu - \widetilde{a}\right)}{\widetilde{a}(1-\widetilde{a})}$$

$$w_{ij} = \frac{1}{N} \sum_{\mu=1}^{P} \frac{\left(\widetilde{\xi_i^\mu} - \widetilde{a}\right)\left(\widetilde{\xi_j^\mu} - \widetilde{a}\right)}{\widetilde{a}(1-\widetilde{a})} = \widetilde{w}_{ij}$$

Por lo tanto, hemos probado que la matriz de pesos del patrón original coincide con la matriz del complementario. Como sabemos que en la energía existirá un mínimo correspondiente para el patrón almacenado, y que la matriz de pesos es la misma para el complementario y el original, entonces podemos deducir que, por cada patrón almacenado, existirán dos mínimos, uno correspondiente al original y otro para el complementario.

Estudiamos ahora el hamiltoniano del complementario, para ello partimos de:

$$H(\mathbf{s}) = -\frac{1}{2}\sum_{i,j} w_{ij}s_i s_j + \sum_i \theta_i s_i$$

Empleamos la relación de las componentes del complementario y el patrón original ($s_i = 1 - \widetilde{s_i}$):

$$H(\mathbf{s}) = -\frac{1}{2}\sum_{i,j} w_{ij}(1-\widetilde{s_i})(1-\widetilde{s_j}) + \sum_i \theta_i(1-\widetilde{s_i})$$

$$H(\mathbf{s}) = -\frac{1}{2}\sum_{i}\left(\sum_j (w_{ij}(1-\widetilde{s_i})(1-\widetilde{s_j})) - 2\theta_i(1-\widetilde{s_i})\right)$$

Como $\theta_i = \frac{1}{2}\sum_j w_{i,j}$ podemos reescribir de forma:

$$H(\mathbf{s}) = -\frac{1}{2}\sum_{i,j} (w_{ij}(1-\widetilde{s_i})(1-\widetilde{s_j}) - w_{i,j}(1-\widetilde{s_i}))$$

Sacamos factor común en el interior de la sumatoria:

$$H(s) = -\frac{1}{2}\sum_{i,j} (w_{ij}(1-\widetilde{s_i})(1-\widetilde{s_j}-1))$$

$$H(\mathbf{s}) = \frac{1}{2}\sum_{i,j} (w_{ij}(1-\widetilde{s_i})\widetilde{s_j}) = \frac{1}{2}\sum_{i,j} (w_{ij}(\widetilde{s_j} - \widetilde{s_i}\widetilde{s_j}))$$

Volvemos a emplear que $\theta_i = \frac{1}{2}\sum_j w_{i,j}$:

$$H(\mathbf{s}) = \frac{1}{2}\sum_{j}\left(\sum_i ((-w_{ij})\widetilde{s_j}\widetilde{s_i}) + 2\theta_j\widetilde{s_j}\right)$$

Separamos los términos:

$$H(\mathbf{s}) = -\frac{1}{2}\sum_{j,i} w_{ij}\widetilde{s_j}\widetilde{s_i} + \sum_j \theta_j\widetilde{s_j}$$

Usamos la propiedad de simetría de la matriz de pesos:

$$H(\mathbf{s}) = -\frac{1}{2}\sum_{i,j} w_{ij}\widetilde{s_i}\widetilde{s_j} + \sum_i \theta_i\widetilde{s_i}$$

Por último, como vimos antes, la matriz de pesos de un patrón y la de su complementario, es la misma, por lo tanto, $w_{i,j} = \widetilde{w}_{i,j}$ y $\theta_{i,j} = \widetilde{\theta}_{i,j}$, de manera:

$$H(\mathbf{s}) = -\frac{1}{2}\sum_{i,j} \widetilde{w_{ij}}\widetilde{s_i}\widetilde{s_j} + \sum_i \widetilde{\theta_i}\widetilde{s_i} = H(\widetilde{\mathbf{s}})$$

Podemos observar que el complementario del patrón almacenado no solo también le corresponderá un mínimo en la energía, sino que tendrá el mismo valor.

## 7.3. Criterio de estabilidad para la convergencia al estado estacionario con la función de Metropolis

Debemos tener en cuenta que la convergencia a un estado estacionario no es instantánea, sino que necesita un número de pasos que dependerá de la temperatura a la que estemos trabajando. Para tener una idea del número de pasos que son necesarios para alcanzar el estado estacionario, tomamos medidas de como evoluciona el coeficiente de solapamiento para distintas temperaturas, introduciendo como patrón inicial uno generado aleatoriamente. El resultado obtenido es:

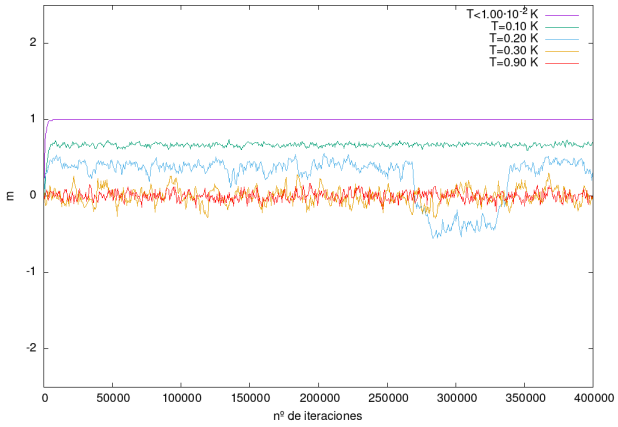



Figura 109: Evolución del coeficiente de solapamiento a diferentes temperaturas

Para realizar la gráfica hemos observado el tiempo de 0,50 pMC, podemos ver la convergencia al estado estacionario (en caso de converger) es bastante rápida, de manera que necesitamos esperar un número de pasos mayor a 0,50 pMC antes de empezar a tomar la media del coeficiente de solapamiento, y la energía. Para asegurarnos la convergencia plena para todas las temperaturas se ha elegido descartar los datos para la media y desviación, de aproximadamente los primeros 12,00 pMC, de esta forma nos aseguramos que hayamos alcanzado el estado estacionario.

Los valores que presentaran una gran desviación típica nos indican que a esa temperatura el solapamiento está convergiendo a varios estados estacionarios, es decir, durante un tiempo corto converge a un estado y posteriormente se rompe la estabilidad convergiendo a otro, siguiendo este proceso de manera repetitiva. Esto nos informa que el sistema está acercándose a la temperatura crítica.

Para poder obtener los datos del criterio de estabilidad, tenemos que abrir la carpeta de **codigo** y después **estabilidad**. Una vez en ella, compilamos el archivo **estabilidad_1_patron.cpp** o **estabilidad_1_patron_2.cpp** y ejecutamos. Una vez que se haya ejecutado el código, si abrimos la carpeta **caracterisiticas** veremos dos ficheros, **T_solapamiento.dat** y **T_Energia.dat**, y una carpeta **evolucion**. Si abrimos la carpeta de evolución nos encontraremos los ficheros con la evolución del coeficiente del solapamiento a las distintas temperaturas. Una vez en este punto, ejecute el archivo de gnuplot **estabilidad.plt**, y se generará un gráfico con la convergencia a distintas temperaturas. Para modificar los parámetros del programa o introducirlos por pantalla, recomendamos emplear los mismos valores que en 4.1, salvo que en esta situación no será necesario incluir el valor del criterio de estabilidad.

## 7.4. Criterio de estabilidad para la convergencia al estado estacionario con la función logística

Al igual que antes, la convergencia al estado estacionario no es instantánea, por lo que necesitamos desarrollar un criterio para saber que pasos debemos descartar a la hora de analizar los datos. Para ello, seguiremos el mismo procedimiento que el apartado anterior, y graficamos el coeficiente de solapamiento en función del número de iteraciones realizadas por el programa, obteniendo:

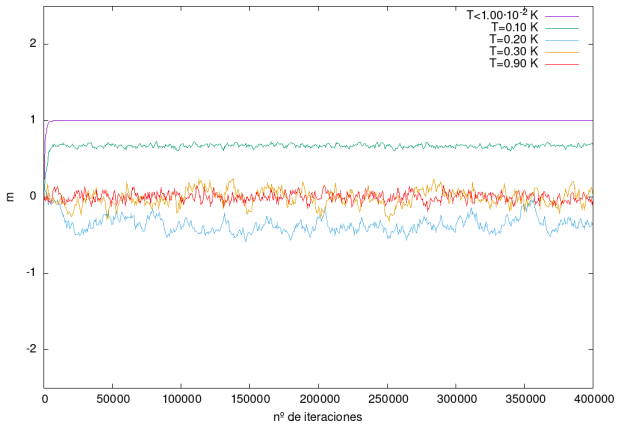



Figura 110: Evolución del solapamiento a distintas temperaturas

Podemos observar como la convergencia al estado estacionario se da aparentemente a la misma velocidad que en el caso anterior. Por lo tanto, podemos usar el mismo criterio de antes para estudiar la convergencia al estado estacionario.

Para obtener el gráfico de la evolución del solapamiento a distintas temperaturas, vamos a **codigo** y después **L_estabilidad**. Una vez allí, compilamos el código y ejecutamos **L_estabilidad.cpp** o **L_estabilidad_2.cpp**. Al igual que en 7.3, vamos a la carpeta de evolución y ejecutamos **estabilidad.plt** para generar la gráfica que mostramos. Para introducir o modificar los valores del programa se debe seguir el mismo procedimiento que en 7.3.

## 7.5. Deducción de la función logística mediante el planteamiento de una máquina de Boltzman

En este apartado vamos a deducir un nuevo criterio para hacer evolucionar la red de Hopfield. Para ello realizaremos un procedimiento parecido al de la Máquina de Boltzman. Partiendo de la energía del sistema, podemos considerar que la relación de energía entre dos vectores que se diferencian en una única componente es:

$$E_{i=on} + Tk_b \cdot ln(p_{i=on}) = E_{i=off} + Tk_b \cdot ln(p_{i=off})$$

Donde el término $Tk_b \cdot ln(p_{i=on})$, sería la probabilidad de que ese spin se encuentre activo. Si despejamos de la expresión anterior llegamos a:

$$\frac{-\Delta E_{on,off}}{k_b T} = ln\left(\frac{p_{i=on}}{p_{i=off}}\right)$$

Podemos emplear la relación siguiente relación $p_{i=on} = 1 - p_{i=off}$, sustituyendo y despejando llegamos a la expresión:

$$p_{i=on} = \frac{1}{1 + e^{\frac{\Delta E_{on,off}}{Tk_b}}}$$

Como realmente nuestra red no es un sistema físico real, el concepto de temperatura tiene un significado más artificial, por lo tanto, podemos incluir la constante de Boltzman dentro del valor de esta temperatura artificial:

$$p_{i=on} = \frac{1}{1 + e^{\frac{\Delta E_{on,off}}{T}}} \tag{11}$$

Obteniendo la ecuación que nos da nuestra probabilidad de transición.

## 7.6. Cálculos de incertidumbres

Para la temperatura crítica, hemos considerado una incertidumbre de tipo B:

$$u_B = \frac{\delta}{\sqrt{12}}$$

Tomando como sensibilidad $\delta_{T_c} = 0{,}01$ K.

En cambio, para los valores obtenidos del coeficiente de solapamiento en función de la temperatura y la energía, hemos empleado la desviación estándar de los resultados una vez alcanzado el estado estacionario:

$$\sigma = \sqrt{\frac{\sum_{i=0}^{N}(x_i - \langle x \rangle)^2}{N-1}}$$

Por otro lado, para la incertidumbre empleada en la fracción crítica $\alpha$, hemos considerado el máximo en valor absoluto entre el factor posterior y previo, es decir:

$$\epsilon = max\left(|\alpha_{-1} - \alpha|, |\alpha_{+1} - \alpha|\right)$$

O lo que es lo mismo, el incremento de la fracción crítica para cada patrón almacenado.

## 7.7. Compilar

En esta sección pretendo dar una serie de indicaciones sobre cómo compilar los programas. A la hora de realizar los programas y probar su funcionamiento, hemos empleado como sistema operativo Debian, usando los siguientes paquetes:

- Para compilar los programas es necesario el compilador g++:

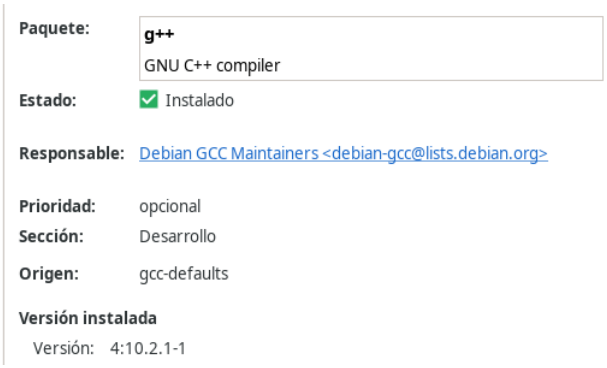



Figura 111: Paquete g++

- Como empleamos un generador de números aleatorios escrito en Fortran, será necesario el compilador de gfortran, para ello hemos empleado los paquetes:


Figura 112: Paquete gfortran-10

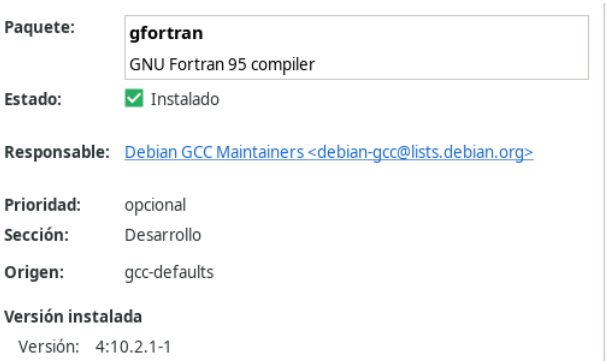



Figura 113: Paquete gfortran


Figura 114: Paquete g77

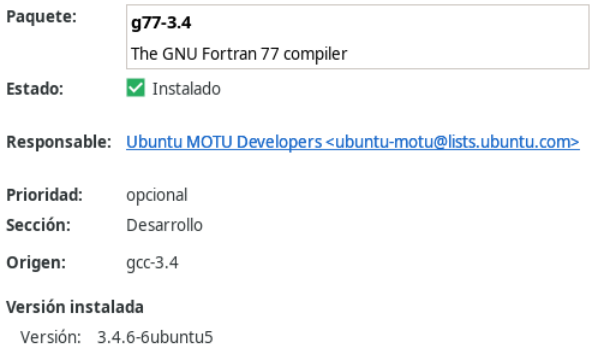



Figura 115: Paquete g77-3.4

Es posible que se produzcan fallos en la compilación o ejecución del programa en caso de no tenerlos instalados.

Para compilar y ejecutar los programas vaya a la carpeta donde se encuentre el código, una vez que se encuentre en ella, siga los siguientes pasos:

- Abra una terminal en el mismo directorio donde se encuentre el archivo que quiere compilar.

- Empleamos el compilador g++ para generar los ejecutables usando el siguiente comando en la terminal:
  **g++ -o (nombre del ejecutable) -Wall Hopfield.cpp aleatorios.f vector.cpp (nombre del fichero que desea compilar)**

**Advertencia:** Una vez compilado el programa, le aparecerá en pantalla una lista de **Warmings**, producidas porque el fichero **aleatorios.f** está escrito en Fortran77:

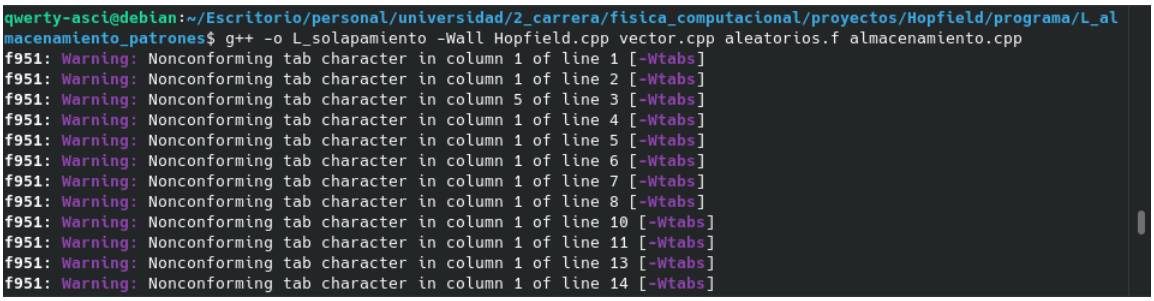



Figura 116: Captura de pantalla de las alertas

Puede ignorar estas alertas perfectamente, ya que no influyen en la ejecución del programa.

- Una vez compilado se generará el ejecutable en esa misma carpeta. Para ejecutar el programa introduzca en la terminal **./(nombre del ejecutable)**.

Para ejecutar los archivos **.plt** para generar los gráficos, simplemente use el comando **gnuplot (nombre del archivo)**.

# Referencias


[1] https://es.wikipedia.org/wiki/Funci%C3%B3n_sigmoide

[2] https://es.wikipedia.org/wiki/Funci%C3%B3n_log%C3%ADstica

[3] https://kripkit.com/red-de-hopfield/

[4] https://es.abcdef.wiki/wiki/Cognitive_model#Associative_memory

[5] http://www.lcc.uma.es/~lfranco/Clase5-optimizacion-Hopfield.pdf

[6] https://ergodic.ugr.es/cphys/LECCIONES/ising/ising-SLIDES.pdf

[7] https://es.wikipedia.org/wiki/Funci%C3%B3n_de_Liapunov

[8] https://es.abcdef.wiki/wiki/Hopfield_network#Working_principles_of_discrete_and_continuous_Hopfield_networks

[9] https://es.wikipedia.org/wiki/Red_neuronal_recurrente

[10] https://es.wikipedia.org/wiki/Teor%C3%ADa_hebbiana

[11] https://es.wikipedia.org/wiki/Hopfield_(RNA)

[12] https://numerentur.org/hopfield/

[13] http://rna.50webs.com/tutorial/RNA_hopfield.html

[14] https://stackoverflow.com/questions/13293762/gfortran-error-trying-to-exec-f951-execvp-no-such-

[15] https://stackoverflow.com/questions/15685089/gfortran-error-trying-to-exec-f951-execvp