Fenòmens col.lectius i transicions de fase

Pràctica 1

Programació, generador de nombres pseudo-aleatoris i reticle de spins.

JB 30/04/2024

Objectius

- Recordar:

  Sistema operatiu (windows o Linux),

  Editor (emacs, gedit, geany,. Notepad++ ...)

  Llenguatge FORTRAN

  Generadors de nombres pseudo-aleatoris

- Fer tres programes:

`P1-exercici-1.f` : genera nombres a l'atzar uniformes reals i discrets amb `mt19937`

`P1-exercici-2.f`: genera una matriu d'spins a l'atzar `LxL (L=96)`

`P1-exercici-3.f`: genera una matriu d'spins a l'atzar i mesura la imantació

- Exercici preparatori P2:mesura de l'Energia

Generadors de nombres pseudo-aleatoris

Usualment generen nombres pseudo-aleatoris U(0,1), és a dir uniformement distribuïts entre 0 i 1. (Estrictament son sempre discrets i correlacionats)

Les simulacions MC consumeixen molts nombres aleatoris. Cal que tinguin una "qualitat" per sobre els generadors habituals (RAN, RAND, etc..).

Cal que el generador ens doni nombres molt descorrelacionats, que sigui molt ràpid, amb un període llarg i amb la màxima precisió (real*8). La majoria de generadors basats en mètodes congruencials lineals senzills, no serveixen.

Generador mt19937 (1)

Usarem el generador mt19937 que genera un nombre uniforme cada vegada que se'l crida. Es basa en una tècnica anomenada Mersenne Twister de M. Matsumoto i T. Nishimura (1997).

http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/VERSIONS/FORTRAN/mt19937ar.f

Es una mica antic, però és portable (independent de la màquina i del sistema operatiu), ràpid i de qualitat acceptable. Consisteix en un codi en f77 que s'anomena `mt19937ar.f` que cal baixar del campus.

Podem simplement copiar-lo sempre sota el codi que vulguem desenvolupar i compilar normalment amb gfortran, o compilar-lo primer creant `mt19937ar.o` i després recordar de linkar aquest objecte cada vegada.

Generador mt19937 (2)

Al principi del programa és necessari inicialitzar els nombres aleatoris amb una llavor integer*4, fent servir `init_genrand(SEED)` i després ja es poden cridar els nombres un per un amb `genrand_real2()`

```
implicit none
..
integer SEED
real*8 x
real*8 genrand_real2
...
SEED= 123456
call init_genrand(SEED)

...
x = genrand_real2()
```

P1-exercici-1.f (1)

Prepareu un programa que es digui `P1-exercici-1.f` que generi `NRAND=40000` nombres uniformes a partir d'una llavor `SEED` i els tregui per pantalla. Feu que el programa calculi també la mitjana i la desviació típica dels nombres generats. Compileu i correu el programa.

Recordeu les instruccions (si teniu el codi `mt19937ar.f` afegit a la part de sota del vostre codi) :

Per compilar:

```
gfortran -O3 P1-exercici-1.f -o P1-exercici-1.exe
```

Això crea l'executable `P1-exercici-1.exe` . Per executar feu:

```
./P1-exercici-1.exe
```

Canvieu la llavor, torneu a compilar i a executar per comprovar que dona una seqüencia diferent

P1-exercici-1.f (2)

Instruccions (si teniu el codi `mt19937ar.f` en un arxiu a part)

Primer per compilar el `mt19937ar.f` :

```
gfortran -c mt19937ar.f
```

Això crea un arxiu "objecte" `mt19937ar.o` que ja no cal que torneu a compilar més

Aleshores per compilar el programa `P1-exercici-1.f` feu:

```
gfortran -O3 mt19937ar.o P1-exercici-1.f -o P1-exercici-1.exe
```

Així li dieu al compilador que afegeixi l'objecte `mt19937ar.o` en el moment de linkar.

P1-exercici-1.f (3)

El programa quedarà aproximadament així:

```
program p1
implicit none
integer*4 SEED, iRAND, NRAND
real*8 x, sum, sum2, sigma, genrand_real2
SEED=541766
NRAND=40000
open (unit=11,file='p1-ex1.dat', status = 'unknown')
call init_genrand(SEED)
sum=0.0d0
sum2=0.0d0
do iRAND=1,NRAND
   x=genrand_real2()
   write(11,*) iRAND,x
   write(*,*) iRAND,x
   sum = sum + x
   sum2=sum2+x*x
Enddo
close(11)
sum = sum/real(NRAND)
sum2=sum2/real(NRAND)
sigma = dsqrt(sum2-sum*sum)
write(*,*) sum, sigma
stop
end
```

A la part de sota hi haureu de tenir el codi `mt19937ar.f` , o tenir l'objecte `mt19937ar.o` en el mateix directori

P1-exercici-1.f

Exemple sortida per pantalla:

```
     39968  0.29927634052000940
     39969   8.4582888754084706E-002
     39970  0.72298080427572131
     39971  0.97612574999220669
     39972  0.31926823756657541
     39973  0.96064286935143173
     39974  0.99496153206564486
     39975  0.35769390873610973
     39976  0.56014761398546398
     39977   6.0718942200765014E-002
     39978  0.77447260566987097
     39979   9.4961621100082994E-002
     39980  0.69634983758442104
     39981  0.74853946384973824
     39982  0.84966650628484786
     39983  0.11687804036773741
     39984   7.6516683911904693E-002
     39985  0.52156925480812788
     39986  0.91755404556170106
     39987  0.76605715439654887
     39988  0.55308032850734890
     39989  0.27363070799037814
     39990  0.98618936026468873
     39991   4.2920378735288978E-002
     39992  0.11325025348924100
     39993  0.18097336939536035
     39994  0.12224724935367703
     39995  0.91789229493588209
     39996  0.74289377732202411
     39997  0.66207375936210155
     39998  0.35844469862058759
     39999  0.48815323458984494
     40000  0.47462705569341779
 mean =   0.49849988834222314
 sigm =   0.28853543051795189
```

P1-exercici-1.f (4)

L'algoritme de Metropolis requereix seleccionar elements de la retícula aleatòriament. Primer verifiquem que som capaços de generar nombres discrets J uniformes en (1,L), anant especialment en compte amb els extrems 1, L.

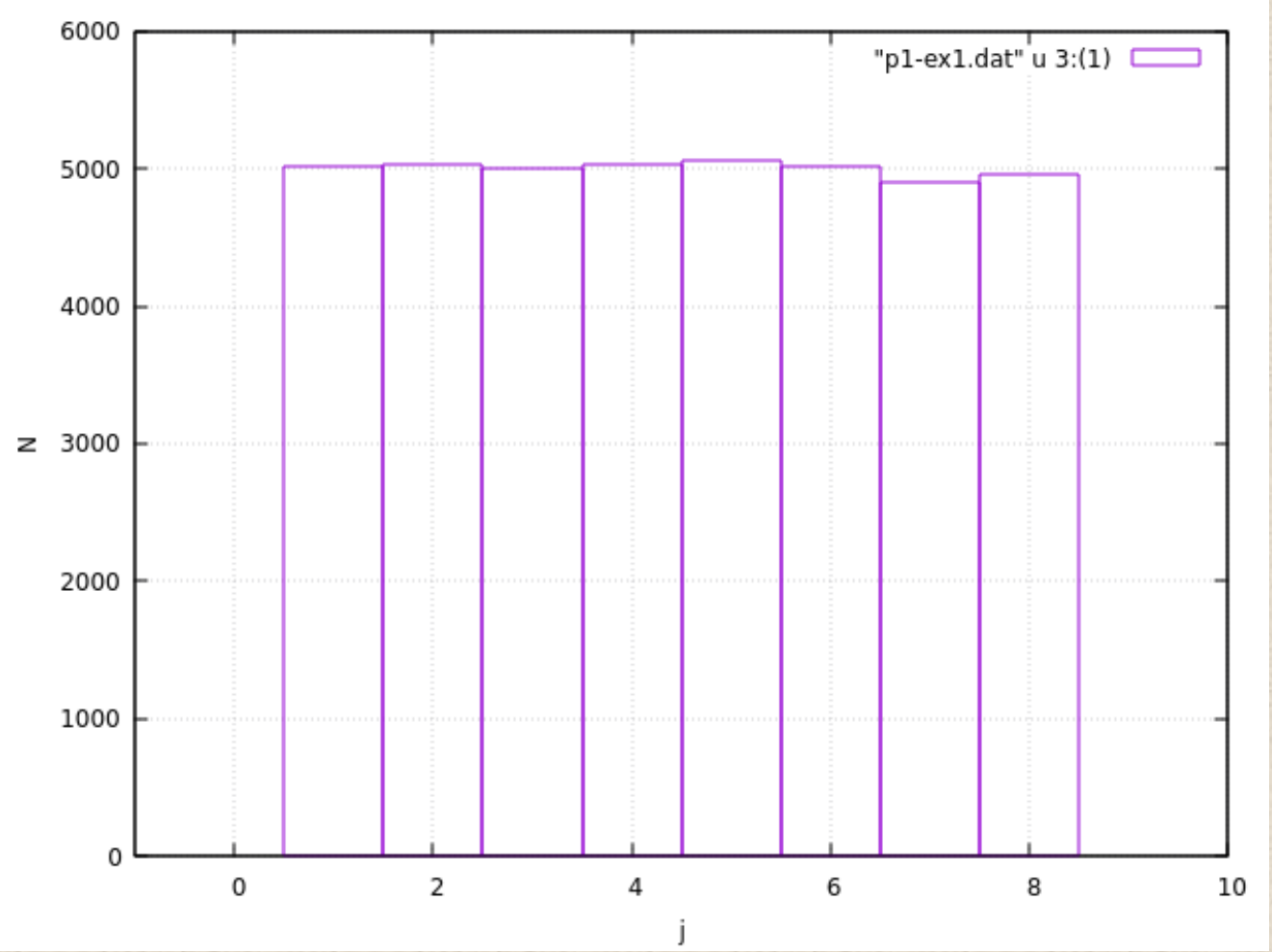
- Modifica el programa P1-exercici-1.f perquè assigni un valor enter entre 1 i 8 a cadascun dels valors.

```
L=8
do iRAND=1,NRAND
   x=genrand_real2()
   jLATTICE =ceiling(x*L)  !valor enter entre 1 i L
   write(11,*) iRAND,x,jLATTICE
   sum = sum + x
   sum2=sum2+x*x
Enddo
```

- Representa l'histograma de valors emprant la funció smooth de gnuplot. Assegura't que cada caixa tingui al voltant de 5000 valors

```
set term png
set out "ex1.png"
set boxwidth 1
set xrange [-1:10]
plot "p1-ex1.dat" u 3:(1) smooth freq w boxes
```

Exemple resultats



Generar una matriu d'spins a l'atzar (1)

Crearem una matriu `S` que contindrà els valors de la configuració d'spins. En principi la recorrerem amb dos index `I` i `J` que vagin cadascun de `1` a `L`, on `L` indica la mida de la xarxa quadrada (`N=LxL`)

Anem a dimensionar el programa pensant que la mida será ajustable. Arribarem com a màxim a `L=96`, encara que usualment treballarem amb `L`'s inferiors

```
INTEGER*4 L
PARAMETER(L=96)
INTEGER*2 S(1:L,1:L)
```

Per inicialitzar la xarxa, volem transformar els nombres uniformes U(0,1) i obtenir un conjunt de valors `1` o `-1` a l'atzar (50% probabilitat) per tal d'omplir la matriu `S(1:L,1:L)`.

Per fer-ho usarem els nombres uniformes que generarem amb genrand_real2()

Si `genrand_real2()` < 0.5 assignarem S=+1

Si `genrand_real2()` >=0.5 assignarem S=-1

Generar una matriu d'spins a l'atzar(2)

El programa quedarà alguna cosa així

```
implicit none
integer*4 SEED, i,j, L
PARAMETER (L=96)
integer*2 S(1:L,1:L)
real*8 genrand_real2
SEED=23456
call init_genrand(SEED)
do i=1,L
   do j=1,L
      if (genrand_real2().lt.0.5D0) then
         S(i,j)=1
      else
         S(i,j)=-1
      endif
   enddo
enddo
```

P1-exercici-2.f

Prepareu un programa que es digui `P1-exercici-2.f`

El programa ha de generar una matriu de mida `N=LxL=96x96` amb spins a l'atzar i escriure-la en un arxiu de sortida

Creeu un arxiu `"P1-configuration.conf"` i escriviu tots els valors `S(I,J)` de la columna `J` per a cada fila `I`, emprant la instrucció `write(unit,*) S(I,:)`.

Creeu un script de gnuplot que dibuixi per la pantalla i en un arxiu .jpeg la configuració que hi ha guardada a l'arxiu `"P1-configuration.conf"`

Utilitzeu la instrucció:

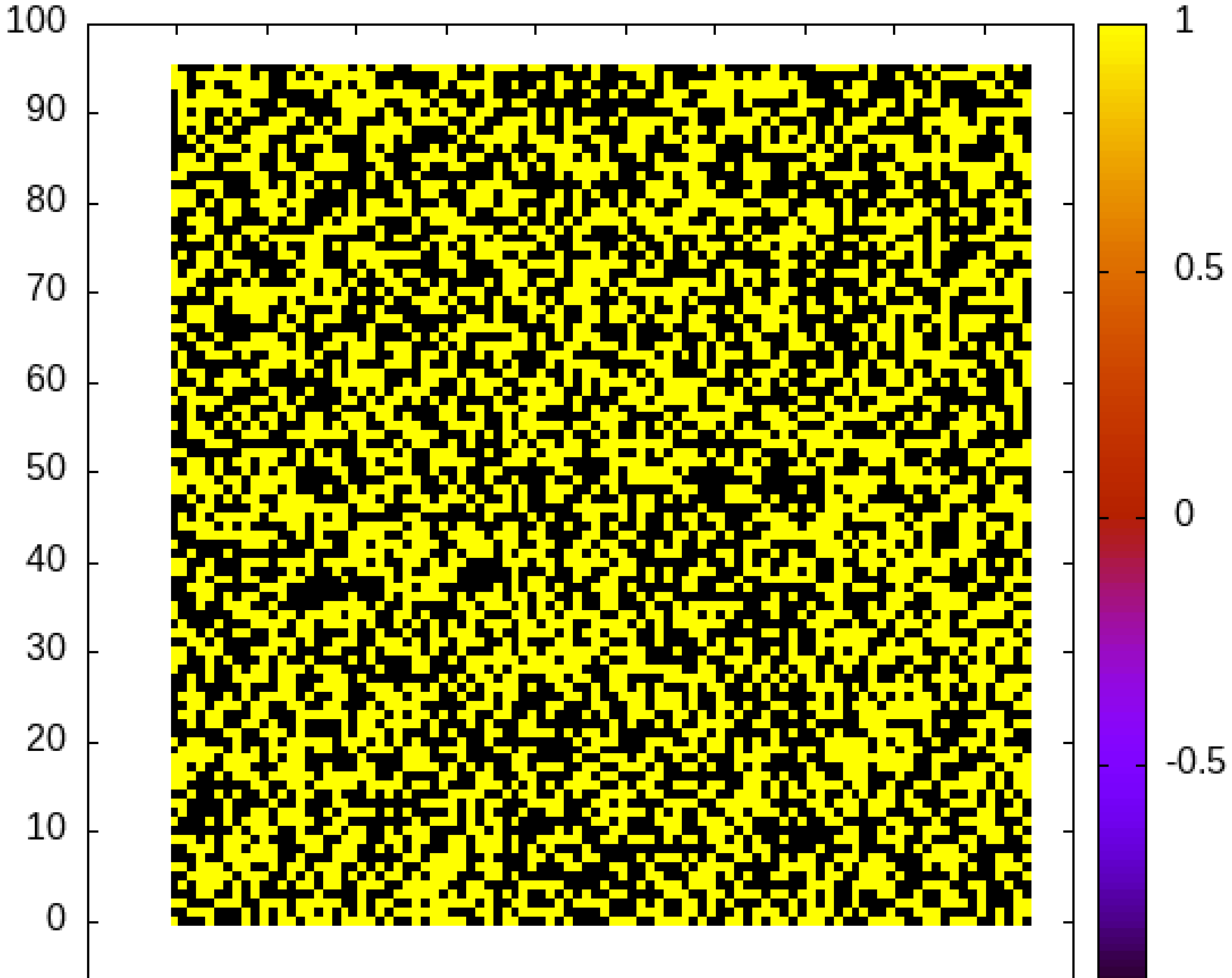
```
plot "P1-configuration.conf" matrix w image not
```

Generar una matriu d'spins a l'atzar(3)

El programa quedarà alguna cosa així

```
implicit none
integer*4 SEED, i,j, L
PARAMETER (L=96)
integer*2 S(1:L,1:L)
real*8 genrand_real2
SEED=23456
open (12, file="P1-configuration.conf")
call init_genrand(SEED)
do i=1,L
   do j=1,L
      if (genrand_real2().lt.0.5D0) then
         S(i,j)=1
      else
         S(i,j)=-1
      endif
   enddo
   write(12,*) S(I,:)
enddo
```

Exemple resultats



Mesurar la imantació

Entre d'altres mesures que haurem de fer, caldrà mesurar en algun moment de la simulació, la imantació instantània de la matriu d'spins `S(I,J)`.

Per a fer-ho construirem una `FUNCTION` que anomenarem `MAGNE`, a la qual passarem la matriu `S` i la mida `L` i ens retornarà la imantació.

```
real*8 FUNCTION MAGNE(S,L)
```

Aquesta `FUNCTION` podria ser perfectament un `INTEGER`, però donat que el valor de la magnetització el mesurarem moltes vegades i el promitjarem (dividint per un nombre gran) , i també el dividirem per la mida del sistema per obtenir la imantació per partícula, es convenient definir ja la funció `MAGNE` com un `REAL*8`

Exemple MAGNE

```
REAL*8 FUNCTION MAGNE(S,L)
INTEGER*2 S(1:L,1:L)
INTEGER*4 I,J,L
REAL*8 MAG
MAG=0.0D0
DO I =1,L
   DO J=1,L
        MAG=MAG+S(i,j)
   ENDDO
ENDDO
MAGNE=MAG
RETURN
END
```

Aquesta function la crideu al programa principal, per exemple, fent:

```
MAG=MAGNE(S,L)
```

on `MAG` és una variable `REAL*8` que heu definit abans

P1-exercici-3.f

Prepareu un programa que es digui `P1-exercici-3.f`, afegint la funció `MAGNE` al programa anterior.

Un cop generada la matriu inicial `S` amb spins a l'atzar, que el programa principal cridi la funció `MAGNE` i ens retorni per pantalla la imantació de la matriu generada a l'atzar.

Hauria de ser un valor petit comparat amb `N=LxL`

Unitats reduïdes

$$H\ (S_1, S_2, \cdots S_N) = -J \sum_{ij} S_i S_j - B \sum_i S_i$$

Definim: H* = H/J, $B^*=B/J$ de forma que:

$$H^*(S_1, S_2, \cdots S_N) = -\sum_{ij} S_i S_j - B^* \sum_i S_i$$

La probabilitat d'acceptació que apareix en l'algorisme serà:

$$e^{-\frac{\Delta H}{k_B T}} = e^{-\frac{\Delta H\ J}{k_B T\ J}} = e^{-\frac{\Delta H^*}{T^*}}$$

- Temperatura reduïda $T^*=k_BT/J$

Exercici de preparació a P2

Prepareu una subrutina ENERG que mesuri l'energia de la configuració d'spins

Recordeu que l'energía és

$$\mathcal{H} = -\sum_{i,j}^{n.n.} S_i S_j$$

Quin problema trobeu? El resoldrem a P2.