Fenòmens col.lectius i transicions de fase

Pràctica 1

Sistema operatiu, editor, FORTRAN i generador de nombres pseudo-aleatoris

EV 02/11/2021

Objectius

- Recordar:

  Sistema operatiu (windows o Linux),

  Editor (emacs, gedit, ...)

  Llenguatge FORTRAN

  Generadors de nombres pseudo-aleatoris

- Fer tres programes obligatoris:

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

`P1-exercici-2.f`: genera una matriu d'spins a l'atzar `LxL (L=64)`

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

- Fer un programa optatiu

`P1-exercici-4.f` : genera nombres a l'atzar discrets

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=50000` 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:

```
implicit none
integer*4 SEED, i,NRAND
real*8 x, sum, sum2, sigma, genrand_real2
SEED=23456
NRAND=50000
call init_genrand(SEED)
sum=0.0d0
sum2=0.0d0
do i=1,NRAND
   x=genrand_real2()
   write(*,*) i,x
   sum = sum + x
   sum2=sum2+x*x
enddo
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:

```
       49965  0.16879353788681328
       49966   4.6839193208143115E-002
       49967  0.51385557721368968
       49968  0.65596234844997525
       49969   7.5051509775221348E-003
       49970  0.91305548371747136
       49971  0.76812729169614613
       49972  0.57909613312222064
       49973  0.71227697608992457
       49974  0.12420654296875000
       49975  0.44272379437461495
       49976  0.70456380909308791
       49977  0.73119275993667543
       49978  0.69285601028241217
       49979  0.89829628169536591
       49980  0.54506867914460599
       49981  0.96495994809083641
       49982  0.65750146983191371
       49983  0.82169914781115949
       49984  0.25021858466789126
       49985  0.99942613905295730
       49986   5.1577002508565784E-002
       49987  0.25995469652116299
       49988  0.20038200425915420
       49989  0.20477369148284197
       49990  0.58367910166271031
       49991  0.67256356659345329
       49992  0.97790627297945321
       49993  0.84726547147147357
       49994  0.47513004415668547
       49995  0.47479080571793020
       49996  0.26027263258583844
       49997  0.76137966034002602
       49998  0.88058630865998566
       49999   5.4939547320827842E-002
       50000  0.23969478602521122
  mean =   0.49918482651493978
  sigm =   0.28780135998291217
MacBook-MacBook-Pro-de-Eduard:P1-tardor2020 Eduard$
```

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=128`, encara que usualment treballarem amb `L`'s inferiors

```
INTEGER*4 L
PARAMETER(L=64)
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=64)
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=64x64` amb spins a l'atzar i escriure en un arxiu de sortida les coordinades `I,J` dels spins que son positius

Per escriure la sortida feu-ho amb una subrutina `WRITECONFIG(S,L)` que obri un arxiu `"P1-configuration.conf"` i escrigui sequencialment en una columna les parelles `I,J` corresponents als spins `S(I,J)=+1`

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"`

Exemple P1-plotconfig.gnu

```
file="P1-configuration.conf"
L=64
symbsize=1.2
set size square
set xrange [0.5:L+0.5]
set yrange [0.5:L+0.5]
plot file using 1:2 with points pt 10 ps symbsize t ""
pause -1
set terminal jpeg
set output "P1-configuration.jpeg"
plot file using 1:2 with points pt 5 ps symbsize t ""
```

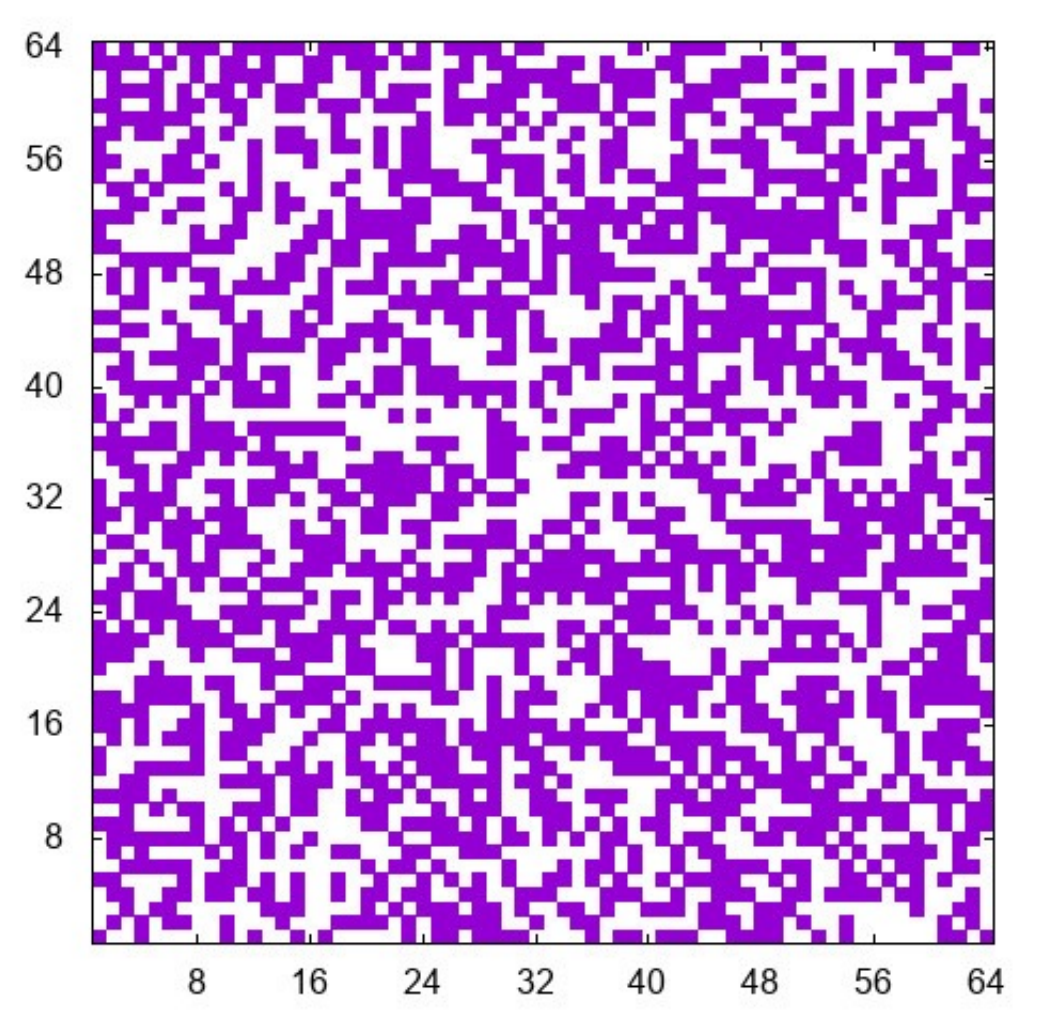
Depenent de la terminal gràfica que feu servir caldra retocar `pt` i `ps` en el dibuix a la pantalla. Aquest exemple esta ajustat al terminal "aqua".

Tambe caldrà retocar la mida `symbsize` quan feu xarxes mes grans.

Tambe podeu fer servir una terminal png

```
set terminal png
set output "P1-configuration.png"
```

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`

Preparar propera sessió

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

Recordeu que l'energía és

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

Exercicis optatius

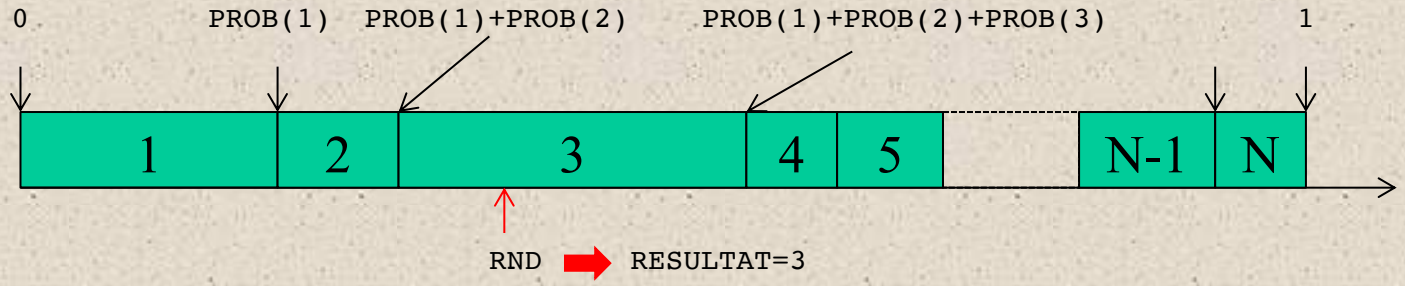
Generar nombres a l'atzar discrets

Volem transformar els nombres uniformes U(0,1) i obtenir nombres sencers a l'atzar entre 1 i N de manera que

1 tingui PROB(1) de sortir
2 tingui PROB(2) de sortir
.....
N tingui PROB(N) de sortir

i evidentment, PROB(1)+PROB(2)+... PROB(K)+...+PROB(N)=1

Per fer-ho cal comparar el nombre a l'atzar uniforme RND amb els límits següents:



Function DICE

A partir del programa `P1-exercici-1.f`, el modificarem i inclourem una function `DICE(RND,N,PROB)` que, a partir d'un nombre uniforme `RND`, retorni un nombre a l'atzar `DICE` entre `1` i `N`, d'acord amb un vector de probabilitats `PROB(1:N)` qualsevol

Esquemàticament ha de fer una cosa així:

```
xlim=0.0D0
do k=1,N
    xlim=xlim+prob(k)
    if rnd<xlim then
       DICE=K  (DICE es el nom de la function)
       return  (surt de la function)
    endif
enddo
```

Noteu que `xlim` va augmentant cada vegada i indica quin es el valor de la suma parcial `prob(1)+prob(2)+...+prob(K)`

P1-exercici-4.f

Construiu un programa `P1-exercici-4.f` que contingui la subrutina `DICE`.

Definiu `PROB` al inici del programa principal

genereu `NRAND=1000` nombres uniformes amb `rgnd()` i transformeu-los a sencers d'acord amb la llei `PROB`, utilitzant `DICE`

Per exemple simuleu 1000 tirades d'un dau trucat amb `N=6`

```
K={1,2,3,4,5,6}
PROB(1)=PROB(2)=PROB(3)=PROB(4)=PROB(5) = 1/12
PROB(6)=1-(5/12)
```

Feu un histograma `hist(1:6)` per comptar quantes vegades ha sortit cada numero, guardeu el resultat en un arxiu `P1-histogram.dat` i representeu-lo amb gnuplot.