UNIVERSITÉ DE GENÈVE — FACULTÉ DES SCIENCES
Sections d'Informatique et de Physique — Profs. B. Chopard et M. Droz

Outils, modélisation et simulation en calcul numérique – Corrigé série 2

22 mars 2005

Exercice 1.
Voir le corrigé de la série 1.

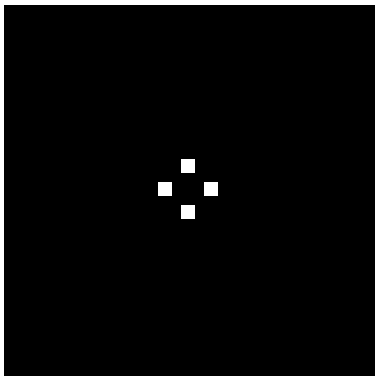
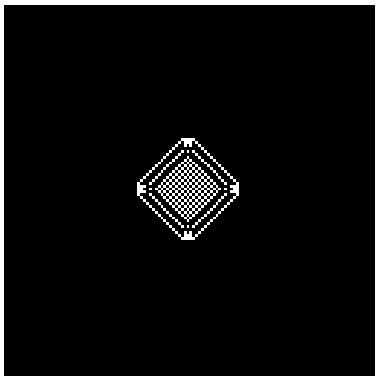
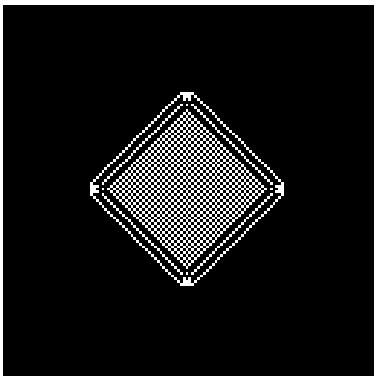
Exercice 2.

i) Constatons d'abord que la règle est additive, c'est-à-dire que toute condition initiale peut être décomposée en une superposition d'images de 1 pixel. Ces configurations évoluent alors indépendamment, et le résultat est superposé pour obtenir la figure finale complète. On peut ainsi considérer, sans restriction de généralité, la condition initiale $\psi(0,0;t=0)=1$ (un pixel).
Soit la décomposition binaire de T : comme T est impair il existe $n \in \mathbb{N}^*$ tel que

$$T = 2^n + \sum_{l=0}^{n-1} a_l 2^l = 2^n + T', \tag{1}$$

où $a_l \in \{0,1\}$ et par construction le nombre d'itérations T' satisfait $T' \leqslant 2^n - 1$. Ainsi, réaliser T itérations de l'automate cellulaire est équivalent à faire d'abord T' itérations, puis les dernières 2^n itérations. Par définition de la règle, et comme nous l'avons vu dans l'exercice 1 de la série 1, l'état $\psi(i,j;T')$ contient alors les termes $\psi(i-T',j;0)$, $\psi(i+T',j;0)$, $\psi(i,j-T';0)$, et $\psi(i,j+T';0)$, ainsi que d'autres termes contenant des translations d'une quantité plus petite que T'. Par conséquent, au temps T' l'extension spatiale d'une configuration est connue et donnée par T'.
Il reste à effectuer les 2^n itérations restantes. A nouveau, nous savons de l'exercice 1 de la série 1 que le résultat sera donné par la superposition de 4 translations par 2^n de la configuration au temps T'. Aucune de ces translations mènera à un effacement par superposition des structures. En effet, l'extension spatiale T' est plus petite que la distance de translation 2^n. Après la translation par 2^n, le bord droit sera à la position $T' - 2^n$ par rapport à l'image après T'. De façon similaire, le bord gauche sera en $-T' + 2^n$. Il n'y a donc en effet pas de recouvrement des structures car $T' < 2^n$ et donc $T' - 2^n < -T' + 2^n$.
Ainsi, comme T est quelconque (mais impair, s'il est pair on peut avoir recouvrement total des configurations ; par exemple pour $T = 2^m$, $m \in \mathbb{N}^*$, on a uniquement 4 translations de la condition initiale), pour chaque $a_l \neq 0$ dans l'Eq. (1), 4 translations sont produites et le résultat final est composé de 4^k translations sans recouvrement complet, où $k = \sum_{l \geqslant 0} a_l$. Lorsque la condition initiale n'est pas un pixel unique, on observe un recouvrement partiel avec interférences destructives (si nécessaire, s'aider de la représentation graphique de l'évolution temporelle obtenue dans l'exercice 1 de la série 1 pour comprendre la preuve).

ii) Le point i) indique que la dynamique engendre des structures géométriques riches (cf. Fig. ci-dessous).

8 itérations 15 itérations 31 itérations

De plus en plus de termes sont engendrés en fonction du nombre d'itérations, et par conséquent la complexité algorithmique doit augmenter. Pour évaluer la complexité algorithmique asymptotique, on constate de l'Eq. (1) que $T \sim 2^k$. Par conséquent $T^2 \sim 4^k$, ce qui est de l'ordre de la complexité algorithmique car comme vu au point i) 4^k correspond au nombre de translations de la configuration initiale. La complexité algorithmique asymptotique est donc de l'ordre du carré du nombre d'itérations.

Exercice 3.

i) L'écriture de la règle est immédiate en considérant les cas $n_i(t) = 1$ et $n_i(t) = 0$ séparément.

ii) La décomposition binaire de 184 est

$$184 = 2^0 \times 0 + 2^1 \times 0 + 2^2 \times 0 + 2^3 \times 1 + 2^4 \times 1 + 2^5 \times 1 + 2^6 \times 0 + 2^7 \times 1 = \sum_{i=0}^{7} 2^i \alpha_i,$$

c'est-à-dire selon la classification de Wolfram

$$\underbrace{111}_{\alpha_7=1} \quad \underbrace{110}_{\alpha_6=0} \quad \underbrace{101}_{\alpha_5=1} \quad \underbrace{100}_{\alpha_4=1} \quad \underbrace{011}_{\alpha_3=1} \quad \underbrace{010}_{\alpha_2=0} \quad \underbrace{001}_{\alpha_1=0} \quad \underbrace{000}_{\alpha_0=0}.$$

Ainsi en associant la valeur 1 à l'existence d'un véhicule, on constate bien que la règle 184 correspond au modèle considéré pour le trafic.

iii) Par conservation, le flux de véhicules peut être défini de façon équivalente comme étant le nombre de véhicules –par unité de temps– entrant dans une cellule donnée (i.e., si $\alpha_4 = 1$ ou $\alpha_5 = 1$), ou en sortant (i.e., si $\alpha_2 = 1$ ou $\alpha_6 = 1$).

Programme Matlab :

```
clear;
RULE = 184; %rule \in [1,256]
N = 500; %# of cells
ITMAX = 1000; %# time steps for the average for 1 point
POINTS = 10; %# points in the graphic
%decimal -> binary
rb = zeros(1,8); %array for binary rule
d1 = RULE;
for (n=[0:7])
    nn=7-n;
```

```
    d2 = 2^nn;
    if (d1-d2 >= 0)
        d1 = d1-d2;
        rb(nn+1) = 1;
    end
end
%loop for the points of the graphic
for j=0:POINTS
    %initial conditions
    ca = ones(1,N);
    r=j/POINTS;
    for i=1:N %on *average*, density*N cars (exact density not needed!)
        randomnumber=rand(1,1);
        if (randomnumber > r)
            ca(1,i)=0;
        end
    end
    density=0;
    for i=1:N
        density=density+ca(1,i);
    end
    density=density/N;
    flux=0;
    for i=1:ITMAX
        tmp=zeros(1,N);
        for (n=[2:N-1])
            x=4*ca(1,n-1)+2*ca(1,n)+ca(1,n+1);
            tmp(1,n)=rb(1,x+1);
        end
        %periodic boundary conditions:
        x=4*ca(1,N)+2*ca(1,1)+ca(1,2);
        tmp(1,1)=rb(1,x+1);
        x=4*ca(1,N-1)+2*ca(1,N)+ca(1,1);
        tmp(1,N)=rb(1,x+1);
        %flow
        x=4*ca(1,N/2-1)+2*ca(1,N/2)+ca(1,N/2+1);
        if (x == 2 | x == 6) %car leaving the cell
            flux=flux+1;
        end
        %update the cellular automata
        ca(1,1:N)=tmp(1,1:N);
    end
    densityplot(j+1,1)=density;
    densityplot(j+1,2)=flux/ITMAX;
end
plot(densityplot(:,1),densityplot(:,2),'kx-');
```

Programme Fortran 90 (plus de 100 fois plus rapide que le programme Matlab, mais nécessite un autre logiciel pour l'affichage graphique, par exemple *GnuPlot*) :

```
module fonctions
implicit none
integer, parameter :: rule = 184 !Wolfram rule
integer, parameter :: N = 100000 !size of the system
integer, parameter :: itmax = 1000000 !maximum number of iterations
integer, parameter :: points = 50 !number of points in the graphic
character*(*), parameter :: filename = 's2.txt' !name of the output file
contains

FUNCTION ran2(idum) !Random generator. Initialize idum with negative integer
INTEGER idum,IM1,IM2,IMM1,IA1,IA2,IQ1,IQ2,IR1,IR2,NTAB,NDIV
REAL :: ran2,AM,EPS,RNMX
PARAMETER (IM1=2147483563,IM2=2147483399,AM=1./IM1,IMM1=IM1-1,IA1=40014,&
          &IA2=40692,IQ1=53668,IQ2=52774,IR1=12211,IR2=3791,NTAB=32,&
          &NDIV=1+IMM1/NTAB,EPS=1.2e-7,RNMX=1.-EPS)
INTEGER :: idum2,j,k,iv(NTAB),iy
SAVE iv,iy,idum2
DATA idum2/123456789/, iv/NTAB*0/, iy/0/
if(idum.le.0)then
  idum=max(-idum,1)
  idum2=idum
  do j=NTAB+8,1,-1
    k=idum/IQ1
    idum=IA1*(idum-k*IQ1)-k*IR1
    if(idum.lt.0)idum=idum+IM1
    if(j.le.NTAB)iv(j)=idum
  enddo
  iy=iv(1)
endif
k=idum/IQ1
idum=IA1*(idum-k*IQ1)-k*IR1
if(idum.lt.0)idum=idum+IM1
k=idum2/IQ2
idum2=IA2*(idum2-k*IQ2)-k*IR2
if(idum2.lt.0)idum2=idum2+IM2
j=1+iy/NDIV
iy=iv(j)-idum2
iv(j)=idum
if(iy.lt.1)iy=iy+IMM1
ran2=min(AM*iy,RNMX)
return
end function ran2
end module fonctions
!---------------------------------------------------------------------
program s2
```

```
use fonctions
implicit none
integer :: seed,i,j,k,l,flux
real :: density,x
integer, dimension(1:8) :: rb
integer, dimension(1:N) :: ca,tmp
real, dimension(1:2,1:points+1) :: densityplot
seed = -14731
rb=0
k=rule
do i=0,7
  j=7-i
  l=2**j
  if (k-l >= 0) then
    k = k-l
    rb(j+1)=1
  end if
end do
do j=0,points
  ca = 1
  x=real(j)/real(points)
  do i=1,N
    if (ran2(seed) > x) ca(i) = 0
  end do
  density=real(sum(ca))/real(N)
  flux=0
  do i=1,itmax
    do k=2,N-1
      l=4*ca(k-1)+2*ca(k)+ca(k+1)
      tmp(k)=rb(l+1)
    end do
    l=4*ca(N)+2*ca(1)+ca(2)
    tmp(1)=rb(l+1)
    l=4*ca(N-1)+2*ca(N)+ca(1)
    tmp(N)=rb(l+1)
    l=4*ca(N/2-1)+2*ca(N/2)+ca(N/2+1)
    if ((l == 2) .or. (l == 6)) flux=flux+1
    ca=tmp
  end do
  densityplot(1,j+1)=density
  densityplot(2,j+1)=real(flux)/real(itmax)
end do
open(unit=1,file=filename)
do i=1,points+1
  write(1,*) densityplot(1,i),densityplot(2,i)
end do
close(1)
end program s2
```

Le diagramme obtenu est représenté sur la Fig. 1.

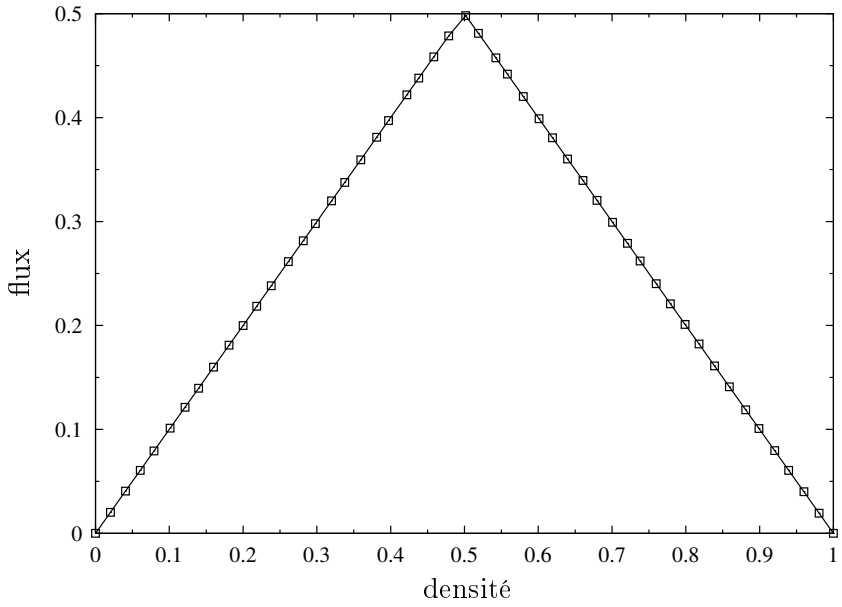


FIG. 1 – Diagramme du flux de véhicules en fonction de leur densité. Paramètres : $N = 10^5$, 50 points avec pour chacun d'eux 10^6 itérations.

iv) Le flux est une fonction croissante linéairement, jusque pour la densité critique $\rho_c = 1/2$ au-delà de laquelle le flux diminue linéairement. L'existence de ce maximum se comprend facilement. En effet, pour $\rho \leqslant 1/2$ un véhicule trouvera toujours, en moyenne, une case vide devant lui. Les véhicules se déplacent en moyenne librement. Par conséquent le flux augmente linéairement jusque pour $\rho = \rho_c$. Par contre, pour $\rho > \rho_c$, il y a plus que $N/2$ véhicules et donc en moyenne plus de 1 véhicule pour 2 cellules d'où un effet d'engorgement et un flux qui diminue linéairement. Le flux ne peux pas dépasser la valeur 0.5 correspondant à la densité critique ρ_c.