Complexity in social dynamics : from the micro to the macro
Laboratory 3

Franco Bagnoli

Namur 7-18/4/2008

1 Laboratory 3

Social Dynamics

1. Network structure and directed percolation.
2. Antoconformistic model.
3. Ising model.
4. Deffuant model.
5. Axelrod model.

Topology, disorder and mean field

- The topology of the network plays a fundamental role.
- In general, we can distinguish among regular and disordered networks.
- The *mean field* analysis, that hold in the absence of correlations, is in general a good approximation to the dynamics on random networks.
- With the program `DPdisordered.f90` we can explore what happens to the directed percolation problem using regular, random (annealed and quenched) networks, small worlds and the mean field approximation.

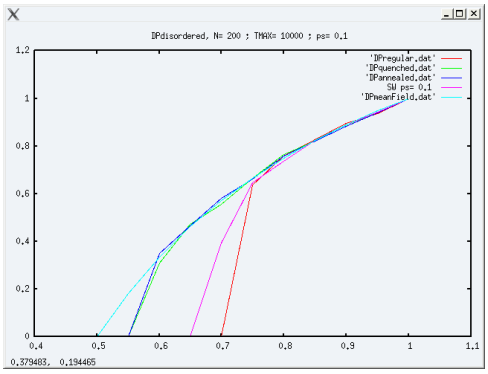
antiPhase.f90

```
program DKPhase
  implicit none
  integer, parameter :: N=200
  integer, parameter :: TMAX = 1000
  integer*1, target :: y(0:N+1)
  integer*1:: x1(1:N)
  integer*1, pointer :: xc(:),xl(:),xr(:)
  integer :: i, j, t
  real :: r(n), p1, p2
  real, allocatable :: d(:,:)
  character*200 :: str
  real :: p1min=0, p1max=1, p1step=0.05
  real :: p2min=0, p2max=1, p2step=0.05
  integer :: np1, np2 , ip1, ip2


  xc => y(1:N)
  xl => y(0:N-1)
  xr => y(2:N+1)
  call random_seed()

  np1 = floor((p1max-p1min)/p1step)
  np2 = floor((p2max-p2min)/p2step)

  allocate(d(0:np1, 0:np2))

  call gnuplotOpen("gnuplot")
  call gnuplotExecute("set term x11; set pm3d; set mouse; unset key")
  call gnuplotExecute("set title 'anticonformistic phase diagram'")
  call gnuplotExecute("set xlabel 'p1'")
  call gnuplotExecute("set ylabel 'p2'")
  call gnuplotExecute("set zlabel 'c'")
```

```
  write (str, *) "splot '-' w l  ", char(0)
  call gnuplotExecute(str)


  do ip1=0,np1
    do ip2=0,np2
      p1=p1min+ip1*p1step
      p2=p2min+ip2*p2step
      !initialization
      call random_number(r)
      xc = floor(r+0.5)
      do t=1, TMAX
        !boundary conditions
        y(0)=y(N)
        y(N+1) = y(1)
        call random_number(r)
        x1 = xc+xr+xl
        where ((x1 == 3) .or. (x1 == 1 .and. r < p1) &
          .or. (x1 == 2 .and. r < p2))
          x1 = 1
        elsewhere
          x1 = 0
        endwhere
        xc = x1
      end do
      d(ip1,ip2) = 0
      do i=1, N
        d(ip1,ip2) = d(ip1,ip2) + xc(i)
      end do
      d(ip1,ip2) = d(ip1,ip2) /N
      print *,  p1, p2, d(ip1,ip2)
      write (str, *) p1,p2,d(ip1,ip2), char(0)
      call gnuplotExecute(str)
      p2 = p2 + p2step
    end do
      call gnuplotExecute("") ! gnuplot wants an empty line
                              ! after each row
      p1 = p1 + p1step
  end do
  call gnuplotExecute("end")
  call gnuplotFlush()
  pause
end program
```

Disordered directed percolation

- One can see that away from the critical value of the probability, the mean field is a quite good approximation.
- The dirdered lattices are very similar to mean field. The annealed version more than the quenced one.
- Random rewiring (small world) is intermediate. By increasing N and $TMAX$, it approaches more the mean field.

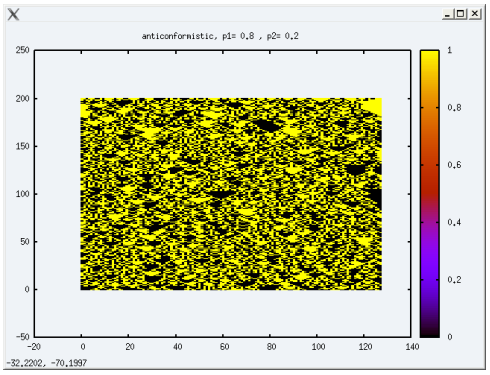


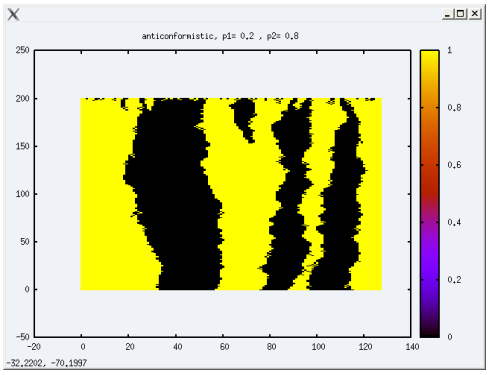
Antoconformistic model

- The anticonformistic model is a simple extension of the DK model, with two absorbing states.
- The basic idea is that one may take decision in opposition to a marginal majority, but cannot violate strong "social norms".
- It is modeled using a one-dimensional cellular automata whose transition probabilities $\tau(1|s)$ depends on the sum s of neighbors ($0 \leq s \leq 3$).
- The absorbing states (social norms) all-zero and all-ones are given by $tau(1|0) = 0$ and $tau(1|3) = 1$.
- The fcontrol parameters are $tau(1|1) = p_1$ and $tau(1|2) = p_2$

Simulations

- When you run `anti.f90`, and experiment with parameters, you find all-zero and all-ones regions, and also an "active state" with a lor of triangles.
- The absorbing states are reached after a transient phase in which the system is divided into homogeneos regions, whose dividing walls perform a sort of random motion.





anti.f90

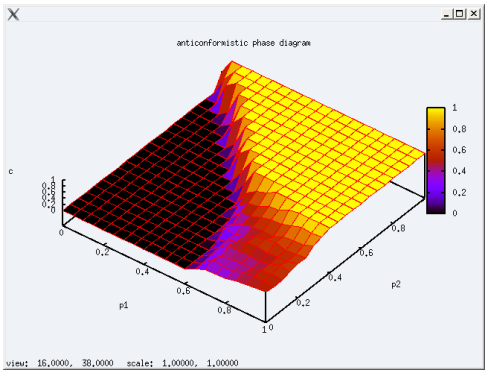
```
program anti ! anticonformistic / conformistic behavior
  implicit none
  integer, parameter :: N=128
  integer, parameter :: TMAX = 200
  integer*1, target :: y(0:N+1)
  integer*1 :: x1(1:N)
  integer*1, pointer :: xc(:),xl(:),xr(:)
  integer :: i, j, t
  real :: r(n), p1, p2
  real :: d
  character*200 :: str
```

```
if (iargc() < 1) then
  print *, "probability p1=p(1|1)?"
  read (*,*) p1
  print *, "probability p2=p(1|2)?"
  read (*,*) p2
else if (iargc() < 2) then
  print *, "probability p_2=p(1|2)?"
  read (*,*) p2
else
  call getarg(1, str)
  read (str,*), p1
  call getarg(2, str)
  read (str,*), p2
end if
print *, "p1=", p1, " p2=", p2

xc => y(1:N)
xl => y(0:N-1)
xr => y(2:N+1)

!initialization
call random_number(r)
xc = floor(r+0.5)

call gnuplotOpen("gnuplot")
call gnuplotExecute("set term x11; set mouse")
call gnuplotExecute("set cbrange [0:1]; unset key")
write(str, *) "set title 'anticonformistic, p1=",p1,", p2=", p2, "'"
call gnuplotExecute(str)
write (str, *) "plot '-' binary flipy array=",N,"x",TMAX+1, " format='%uchar' " //&
  " with image", char(0)
call gnuplotExecute(str)
call gnuplotWrite(xc, sizeof(xc))

do t=1, TMAX
  !boundary conditions
  y(0)=y(N)
  y(N+1) = y(N)
  call random_number(r)
  x1 = xr + xc + xl
  where ((x1 == 3)  .or. (x1 == 2 .and. r < p2) .or. &
    (x1 == 1 .and. r < p1))
    x1=1
  elsewhere
    x1=0
  endwhere
  xc = x1
  call gnuplotWrite(xc,sizeof(xc))
end do

call gnuplotFlush()

d=0
do i=1, N
  d = d + xc(i)
```

```
  end do
  d = d /N

  write(*, *) "d=", d


  pause
end program
```

Simulations

- In order to have an idea of the phase diagram, let run `antiPhase`. The order parameter is the density c of ones in the asymptotic configuration.
- You can see that there is a region in which the transition between all-zeros to all-ones is sharp (a "first-order" phase transition), and a phase (active) whene the transition is smooth ("second-order").



antiPhase.f90

```
program DKPhase
  implicit none
  integer, parameter :: N=200
  integer, parameter :: TMAX = 1000
  integer*1, target :: y(0:N+1)
  integer*1:: x1(1:N)
  integer*1, pointer :: xc(:),xl(:),xr(:)
  integer :: i, j, t
  real :: r(n), p1, p2
  real, allocatable :: d(:,:)
  character*200 :: str
  real :: p1min=0, p1max=1, p1step=0.05
  real :: p2min=0, p2max=1, p2step=0.05
  integer :: np1, np2 , ip1, ip2


  xc => y(1:N)
  xl => y(0:N-1)
  xr => y(2:N+1)
  call random_seed()

  np1 = floor((p1max-p1min)/p1step)
  np2 = floor((p2max-p2min)/p2step)

  allocate(d(0:np1, 0:np2))
```

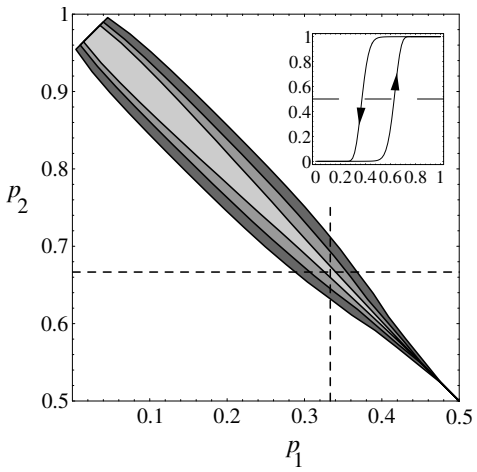
```
  call gnuplotOpen("gnuplot")
  call gnuplotExecute("set term x11; set pm3d; set mouse; unset key")
  call gnuplotExecute("set title 'anticonformistic phase diagram'")
  call gnuplotExecute("set xlabel 'p1'")
  call gnuplotExecute("set ylabel 'p2'")
  call gnuplotExecute("set zlabel 'c'")

  write (str, *) "splot '-' w l  ", char(0)
  call gnuplotExecute(str)


  do ip1=0,np1
    do ip2=0,np2
      p1=p1min+ip1*p1step
      p2=p2min+ip2*p2step
      !initialization
      call random_number(r)
      xc = floor(r+0.5)
      do t=1, TMAX
        !boundary conditions
        y(0)=y(N)
        y(N+1) = y(1)
        call random_number(r)
        x1 = xc+xr+xl
        where ((x1 == 3) .or. (x1 == 1 .and. r < p1) &
          .or. (x1 == 2 .and. r < p2))
          x1 = 1
        elsewhere
          x1 = 0
        endwhere
        xc = x1
      end do
      d(ip1,ip2) = 0
      do i=1, N
        d(ip1,ip2) = d(ip1,ip2) + xc(i)
      end do
      d(ip1,ip2) = d(ip1,ip2) /N
      print *,  p1, p2, d(ip1,ip2)
      write (str, *) p1,p2,d(ip1,ip2), char(0)
      call gnuplotExecute(str)
      p2 = p2 + p2step
    end do
      call gnuplotExecute("") ! gnuplot wants an empty line
                              ! after each row
      p1 = p1 + p1step
  end do
  call gnuplotExecute("end")
  call gnuplotFlush()
  pause
end program
```

Extensions

- The first-order phase transitions are characterized by histeresys (moe than one stable state at the

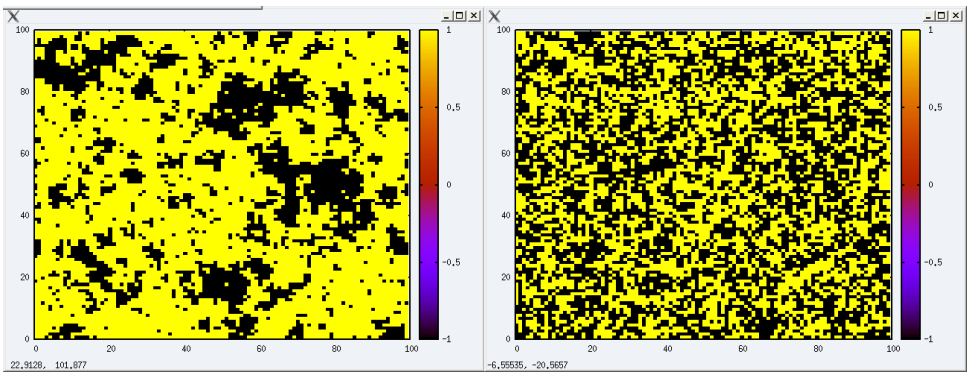
same time). If you change the parameters during the run, you may see that the all-zero phase does not looses stability when entering the all-one phase. If you want to see the transition, you have to add a small perturbation: $tau(1|0) = \varepsilon$ and $tau(1|3) = 1 - \varepsilon$.

- You may also experiment with different topologies. What about developing a 2d version?



Ising model

- The ising model is a prototype of phase transitions without absorbing states (in at least 2D: `Ising.f90`).
- It is also the base of many models (impact factor, Hopfield, etc.)
- At high temperature, the typical configuration is disordered, with small clusters and small correlation lenght.
- At low temperature, there is a majority of spin in an orientation. Clusters are large, but the correlation length is small.
- At the critical temperature, the correlation length diverges.



Ising.f90

```
program Ising
  implicit none
  integer, parameter :: TMAX=100
  integer, parameter :: L=100
  integer*1, target :: data (0:L+1,0:L+1)
  integer*1, pointer :: C(:,:)
  integer*1, pointer :: E(:,:)
  integer*1, pointer :: S(:,:)
  integer*1, pointer :: W(:,:)
```

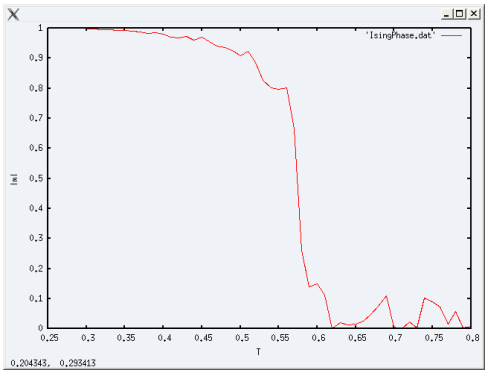
```
integer*1, pointer :: N(:,:)
integer*1, pointer :: SE(:,:)
integer*1, pointer :: SW(:,:)
integer*1, pointer :: NE(:,:)
integer*1, pointer :: NW(:,:)
real :: h, deltaE, Temp
integer :: i, j, t,t1, tt
real :: r, rr, p
character*200 :: str
real :: d(TMAX)

if (iargc() < 1) then
  print *, "temperature?"
  read (*,*) Temp
else
  call getarg(1, str)
  read (str,*), Temp
end if
print *, "Temp=", Temp


C => data(1:L,1:L)
E => data(1:L, 2:L+1)
W => data(1:L, 0:L-1)
N => data(0:L-1, 1:L)
S => data(2:L+1, 1:L)
NE => data(0:L-1, 2:L+1)
SE => data(2:L+1, 2:L+1)
NW => data(0:L-1, 0:L-1)
SW => data(2:L+1, 0:L-1)

call random_seed

d=0

do i=1, L
  do j=1, L
    call random_number(r)
    data(i,j) = 2*floor(r+0.5)-1
  end do
end do

C=1

! boundary conditions
data(:,0) = data(:,L)
data(:,L+1) = data(:,1)
data(0,:) = data(L,:)
data(L+1,:) = data(1,:)


call gnuplotSelect(0)
call gnuplotOpen("gnuplot")
call gnuplotExecute("set mouse")
call gnuplotExecute("set term x11")
call gnuplotExecute("set cbrange [-1:1]")
```

```
  write (str,*) "set xrange [0:",L,"]; set yrange [0:",L,"]", char(0)
  call gnuplotExecute(str)
  call gnuplotSelect(1)
  call gnuplotOpen("gnuplot")
  call gnuplotExecute("set term x11")


  do t=1,TMAX ! Monte-Carlo steps
    do t1=1,L**2 ! Monte-Carlo steps
      call random_number(rr)
      i = floor(rr*L)+1
      call random_number(rr)
      j = floor(rr*L)+1
      h = (N(i,j)+S(i,j)+W(i,j)+E(i,j))/4. !local field
      deltaE = 2*C(i,j)*h
      p = exp(-deltaE/Temp)
      call random_number(rr)
      if (rr < p) then
        C(i,j)=-C(i,j)
        if (i == 1) data(L+1, j) = data(i,j)
        if (i == L) data(0, j) = data(i,j)
        if (j == 1) data(i, L+1) = data(i,j)
        if (j == L) data(i, 0) = data(i,j)
      end if
    end do
    call gnuplotSelect(0)
    write (str, *) "plot '-' "// &
    " binary array=",L,"x",L," format='%char'"  // &
    " title 't=",t,"'  with image ", char(0)
    call gnuplotExecute(str)
    call gnuplotWrite(C, sizeof(C))
    call gnuplotFlush()
    d(t)=0
    do i=1,L
      do j=1,L
        d(t) = d(t) + C(i,j)
      end do
    end do
    print *, t, d(t)

    d(t) = d(t) /L**2
    call gnuplotSelect(1)
    write(str, *) "plot '-' binary array=",TMAX," w l"
    call gnuplotExecute(str)
    call gnuplotwrite(d,sizeof(d))
    call gnuplotFlush()
  end do
  pause
end program
```

Ising model

- In order to put into evidence the phase transition, try `IsingPhase.f90`.
- In the y axes, there is the absolute value of the magnetization m, since starting from $m = 0.5$ in the magnetized phase one can obtain either $m > 0$ or $m < 0$.

- Try to add a magnetic field, and measure the size of a cluster that forms around a spin forced to stay in the opposite direction.



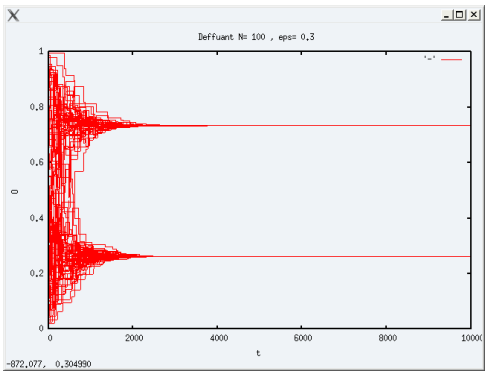
IsingPhase.f90

```
program IsingPhase
  implicit none
  integer, parameter :: TMAX=1000
  integer, parameter :: L=80
  integer*1, target :: data (0:L+1,0:L+1)
  integer*1, pointer :: C(:,:)
  integer*1, pointer :: E(:,:)
  integer*1, pointer :: S(:,:)
  integer*1, pointer :: W(:,:)
  integer*1, pointer :: N(:,:)
  integer*1, pointer :: SE(:,:)
  integer*1, pointer :: SW(:,:)
  integer*1, pointer :: NE(:,:)
  integer*1, pointer :: NW(:,:)
  real :: h, deltaE, Temp
  real :: Tstart=0.3, Tend=0.8, Tstep=.01
  integer :: i, j, t,t1, tt
  real :: r, rr, p
  character*200 :: str
  real :: d

  C => data(1:L,1:L)
  E => data(1:L, 2:L+1)
  W => data(1:L, 0:L-1)
  N => data(0:L-1, 1:L)
  S => data(2:L+1, 1:L)
  NE => data(0:L-1, 2:L+1)
  SE => data(2:L+1, 2:L+1)
  NW => data(0:L-1, 0:L-1)
  SW => data(2:L+1, 0:L-1)

  call random_seed
  open(unit=12, file="IsingPhase.dat", status="replace", action="write")

  Temp = Tstart
  do while (Temp < Tend)

    d=0
    do i=1, L
      do j=1, L
        call random_number(r)
```

```
        data(i,j) = 2*floor(r+0.5)-1
      end do
    end do

    C=1

    ! boundary conditions
    data(:,0) = data(:,L)
    data(:,L+1) = data(:,1)
    data(0,:) = data(L,:)
    data(L+1,:) = data(1,:)

    do t=1,TMAX ! Monte-Carlo steps
      do t1=1,L**2 ! Monte-Carlo steps
        call random_number(rr)
        i = floor(rr*L)+1
        call random_number(rr)
        j = floor(rr*L)+1
        h = (N(i,j)+S(i,j)+W(i,j)+E(i,j))/4. !local field
        deltaE = 2*C(i,j)*h
        p = exp(-deltaE/Temp)
        call random_number(rr)
        if (rr < p) then
          C(i,j)=-C(i,j)
          if (i == 1) data(L+1, j) = data(i,j)
          if (i == L) data(0, j) = data(i,j)
          if (j == 1) data(i, L+1) = data(i,j)
          if (j == L) data(i, 0) = data(i,j)
        end if
      end do
    end do
    d=0
    do i=1,L
      do j=1,L
        d = d + C(i,j)
      end do
    end do

    d = d /L**2
    print *, Temp, abs(d)
    write(12,*) Temp, abs(d)
    Temp = Temp + Tstep
  end do
  close(12)
  call gnuplotOpen("gnuplot")
  call gnuplotExecute("set term x1; unset key")
  call gnuplotExecute("set xlabel 'T'; set ylabel '|m|'")
  call gnuplotExecute("plot 'IsingPhase.dat' w l")
  call gnuplotFlush()
  pause
  call gnuplotClose()
end program
```

Deffuant model

- The deffuant model `deffuant.f90` uses continuous opinions from 0 to one.
- when two individuals meet, if the distance between their opinion is smaller than a certain threshold ε, their opinion converges.
- The final state is given by one or more clusters.
- It is an example of an irreversible process to a fixed point. In this case the phase transition corresponds to a change in stability of different attractors.



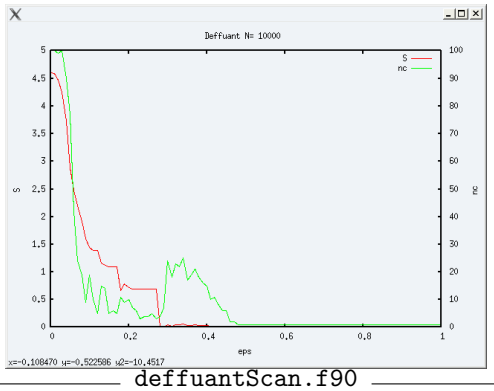
deffuant.f90

```
program deffuant
  implicit none
  integer :: i,j,k, t
  integer, parameter :: N = 100
  integer, parameter :: TMAX=10000
  integer, parameter :: HISTMAX=1000
  real :: O(N)
  real :: hist(N,0:HISTMAX,2)
  integer :: histn(N)
  real :: mu = 0.5
  real :: eps
  real :: r
  character*200:: str

  if (iargc() < 1) then
    print *, "input eps"
    read (*,*) eps
  else
    call getarg(1,str)
    read(str, *) eps
  end if

  call random_seed()
  !initialization
  call random_number(O)
  hist(:,0,1) = O
  hist(:,0,2) = O
  histn=0

  do t=1, TMAX
    call random_number(r)
    i = floor(r*N)+1
    call random_number(r)
    j = floor(r*N)+1
    if (abs(O(i)-O(j))<eps) then
```

```
      O(i) = O(i) + mu*(O(j)-O(i))
      O(j) = O(j) + mu*(O(i)-O(j))

      !recording history
      if (histn(i) < HISTMAX) then
        histn(i) = histn(i)+1
        hist(i,histn(i),1)=t
        hist(i,histn(i),2)=O(i)
      end if
      if (histn(j) < HISTMAX) then
        histn(j) = histn(j)+1
        hist(j,histn(j),1)=t
        hist(j,histn(j),2)=O(j)
      end if

    end if
  end do

  !plot
  call gnuplotOpen("gnuplot")
  call gnuplotExecute("set term x11; set mouse")
  call gnuplotExecute("set xlabel 't'; set ylabel 'O'")
  write(str,*) "set title 'Deffuant N=",N,", eps=", eps,"'"
  call gnuplotExecute(str)
  call gnuplotExecute("set yrange [0:1]")
  write(str, *) "plot '-' w l "
  call gnuplotExecute(str)
  do i=1, N
    write(str, *) hist(i,0,1), hist(i,0,2)
    call gnuplotExecute(str)
    do j=1,histn(i)
      write(str, *) hist(i,j,1), hist(i,j-1,2) ! steps
      call gnuplotExecute(str)
      write(str, *) hist(i,j,1), hist(i,j,2)
      call gnuplotExecute(str)
    end do
    write(str, *) TMAX, hist(i,histn(i),2)
    call gnuplotExecute(str)
    call gnuplotExecute("") ! an empty line makes gnuplot not join lines
  end do
  call gnuplotExecute("end")
  call gnuplotFlush()
  pause

end program
```

Deffuant model 1

- If one want to have an idea of the phase transition, as usual one has to "scan" for different values of ε (`deffuantScam.f90`).
- Two possible order parameters are the number of clusters n_c or the entropy S of the final distribution. As can be noticed on the graph, the entropy is a much more robust quantity, since it "weights" the clusters with their size. There can be situations with a lot of tiny clusters and a big one: the entropy almost neglects the small ones.



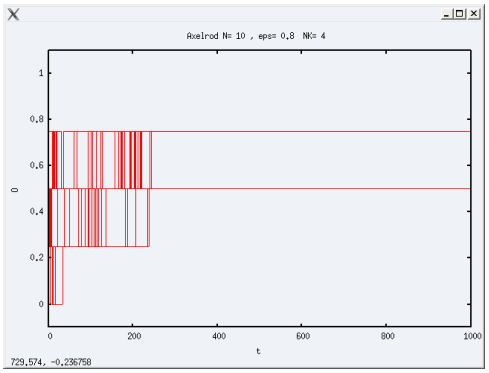
deffuantScan.f90

```
program deffuantScan
  implicit none
  integer :: i,j,k, t
  integer, parameter :: N = 10000
  integer, parameter :: TMAX=1000000
  real :: O(N)
  real :: mu = 0.5
  real :: eps
  real :: r
  character*200:: str
  integer, parameter :: neps=100
  integer, parameter :: resc=100 ! histogram resolution
  real :: h(0:resc) !histogram
  real :: S ! entropy
  integer :: nc

  call random_seed()
  !initialization

  open(unit=12, file="deffuantScan.dat", action="write", status="replace")

  do k=0,neps
    eps = real(k)/neps
    print *, eps
    call random_number(O)
    do t=1, TMAX
      call random_number(r)
      i = floor(r*N)+1
      call random_number(r)
      j = floor(r*N)+1
      if (abs(O(i)-O(j))<eps) then
        O(i) = O(i) + mu*(O(j)-O(i))
        O(j) = O(j) + mu*(O(i)-O(j))
      end if
    end do
    ! histogram
    h=0
    do i=1, N
      j = floor(O(i)*resc)
      h(j)=h(j)+1
    end do
    h = h / N
    ! entropy and number of clusters
    S = 0
```

```
    nc = 0
    do i=0, resc
      if (h(i)>0) then
        nc = nc +1
        S = S - log(h(i))*h(i)
      end if
    end do
    write(12, *) eps, S, nc
  end do
  close(12)

  !plot
  call gnuplotOpen("gnuplot")
  call gnuplotExecute("set term x11; set mouse")
  call gnuplotExecute("set xlabel 'eps'; set ylabel 'S'; set y2label 'nc'; set y2tics")
  write(str,*) "set title 'Deffuant N=",N,"'"
  call gnuplotExecute(str)
  write(str, *) "plot 'deffuantScan.dat' t 'S' w l, ", &
    "'' u 1:3 t 'nc' axis x1y2 w l"
  call gnuplotExecute(str)
  call gnuplotFlush()
  pause

end program
```

Axelrod model

- The Axelrod model (`axelrod.f90`) is a sort of "microscopic" version of the Deffuant one.
- In this case individuals are represented as vectors of discrete variables (Boolean in the example). We visualize the sum O of Boolean variables.
- Since the phase space is high dimensional, more configurations may correspond to the same value.
- It may happen that two individuals with similar values of the "macroscopic" variable O are actually quite distant and do not merge



axelrod.f90

```
program axelrod
  implicit none
  integer :: i,j,k, t
  integer, parameter :: N = 10
  integer, parameter :: TMAX=1000
  integer, parameter :: HISTMAX=1000
  integer, parameter :: NK = 4 ! number of factors
```

```
integer*1 :: F(N,NK) ! factors (characteristics)
real :: O(N) ! opinion (for plotting..)
real :: hist(N,0:HISTMAX,2)
integer :: histn(N)
real :: mu = 0.5
real :: eps
integer ::  w
integer :: diff
real :: r
character*200:: str

if (iargc() < 1) then
  print *, "input eps"
  read (*,*) eps
else
  call getarg(str,1)
  read(str, *) eps
end if
diff = floor(eps*NK)
print *, "eps=", eps," corresponding to diff=", diff," for ", NK, &
  " factors."


call random_seed()
!initialization
do i=1, N
  do j =1, NK
    call random_number(r)
    F(i,j) = 2*floor(r+0.5)-1
  end do
  O(i) = (sum(F(i,:))+NK)/(2.*NK)
end do
hist(:,0,1) = 0
hist(:,0,2) = 0
histn=0

do t=1, TMAX
  call random_number(r)
  i = floor(r*N)+1
  call random_number(r)
  j = floor(r*N)+1
  w = 0
  do k=1,NK
    w = w + F(i,k)*F(j,k)
  end do
  w = (w+NK)/2
  if (w<diff .and. w < NK) then
    ! synchronize one more component
    do k=1, NK
      if (F(i,k) .ne. F(j,k)) exit
    end do
    call random_number(r)
    F(i,k) = 2*floor(r+0.5) -1
    F(j,k) = 2*floor(r+0.5) -1
    O(i) = (sum(F(i,:))+NK)/(2.*NK)
    O(j) = (sum(F(j,:))+NK)/(2.*NK)
```

```
      !recording history
      if (histn(i) < HISTMAX) then
        histn(i) = histn(i)+1
        hist(i,histn(i),1)=t
        hist(i,histn(i),2)=O(i)
      end if
      if (histn(j) < HISTMAX) then
        histn(j) = histn(j)+1
        hist(j,histn(j),1)=t
        hist(j,histn(j),2)=O(j)
      end if
    end if
  end do
  print *, "plotting"
  !plot
  call gnuplotOpen("gnuplot")
  call gnuplotExecute("set term x11; set mouse; unset key")
  call gnuplotExecute("set xlabel 't'; set ylabel 'O'")
  write(str,*) "set title 'Axelrod N=",N,", eps=", eps," NK=",NK,"'"
  call gnuplotExecute(str)
  call gnuplotExecute("set yrange [-0.1:1.1]")
  write(str, *) "plot '-' w l "
  call gnuplotExecute(str)
  do i=1, N
    write(str, *) hist(i,0,1), hist(i,0,2)
    call gnuplotExecute(str)
    do j=1,histn(i)
      write(str, *) hist(i,j,1), hist(i,j-1,2) ! steps
      call gnuplotExecute(str)
      write(str, *) hist(i,j,1), hist(i,j,2)
      call gnuplotExecute(str)
    end do
    write(str, *) TMAX, hist(i,histn(i),2)
    call gnuplotExecute(str)
    call gnuplotExecute("") ! an empty line makes gnuplot not join lines
  end do
  call gnuplotExecute("end")
  call gnuplotFlush()
  pause

end program
```