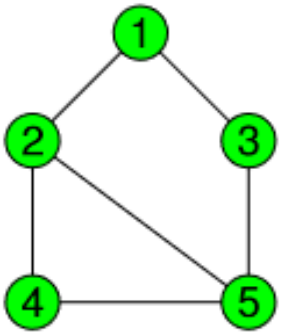
Nathan MARTIN - Alexandre SPRICH

TP Systèmes multi-agents

L'objectif de ce TP est de mettre en application des algorithmes de consensus sur des systèmes multi-agents.

Exemple 1 :

Dans ce premier exemple, très basique, nous considérons le réseau suivant, composé de 5 agents :



Chacun de ces gens possède un état x(t) qui évolue au cours du temps :

$$\dot{x}(t) = -L(t)x(t)$$

Avec L la matrice laplacienne correspondant au réseau en question :

$$L = \begin{pmatrix} 2 & -1 & -1 & 0 & 0 \\ -1 & 3 & 0 & -1 & -1 \\ -1 & 0 & 2 & 0 & -1 \\ 0 & -1 & 0 & 2 & -1 \\ 0 & -1 & -1 & -1 & 3 \end{pmatrix}$$

On dit que les agents parviennent à un consensus si :

$$\lim_{t \to +\infty} x_1(t) = \cdots = \lim_{t \to +\infty} x_n(t)$$

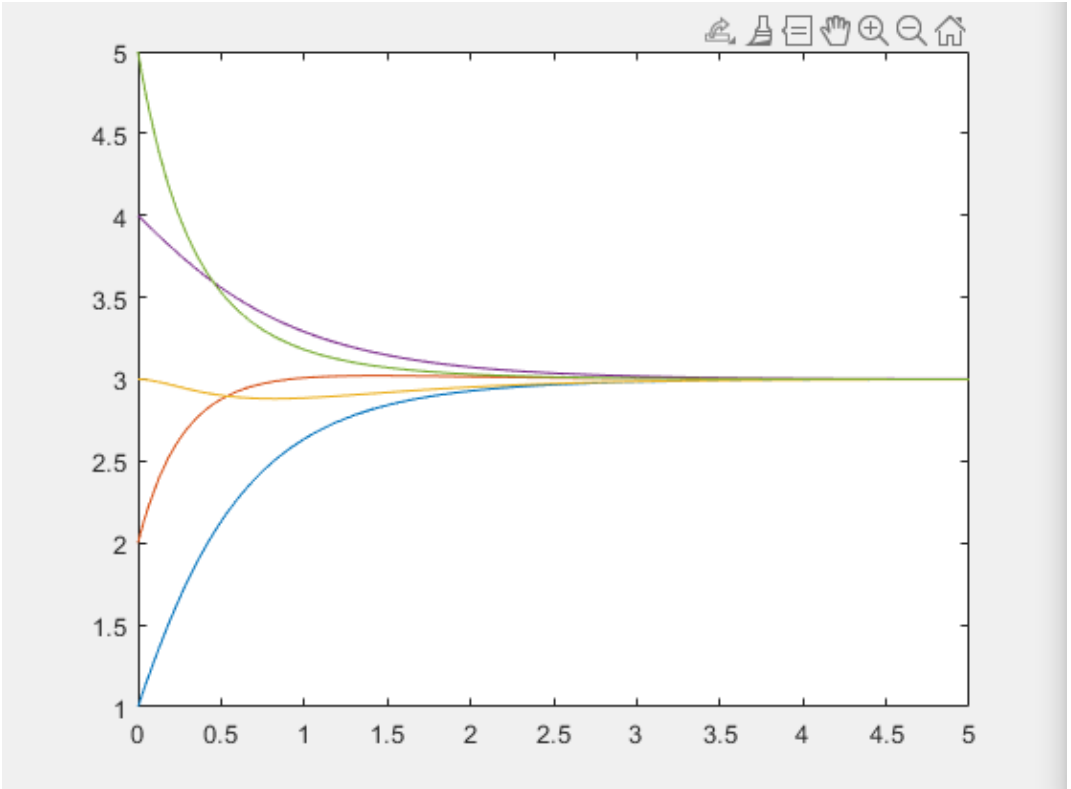
Autrement dit, un consensus est trouvé entre les agents si tous convergent vers la même valeur au bout d'un certain temps.

Le code Matlab permettant de résoudre ce problème est le suivant :

```
X0=[1 ;2 ;3 ;4 ;5];
L=[2 -1 -1 0 0 ; -1 3 0 -1 -1 ; -1 0 2 0 -1 ; 0 -1 0 2 -1 ; 0 -1 -1 -1 3];
tspan = [0 5];
[t,x] = ode45(@(t,x)  -L*x, tspan,X0);
figure
plot(t,x)
```

Ainsi, on attribue à chaque agent un état initial x0, et l'on applique l'algorithme de consensus à l'aide de la fonction ode45.

Cela nous permet d'obtenir le résultat suivant :



On constate alors un consensus autour de la valeur x*=3.
Cette valeur de consensus correspond à la moyenne des valeurs initiales x0.

Par ailleurs, si un consensus est trouvé (uniquement possible si le graphe étudié est connexe), on doit pouvoir vérifier l'inégalité suivante :

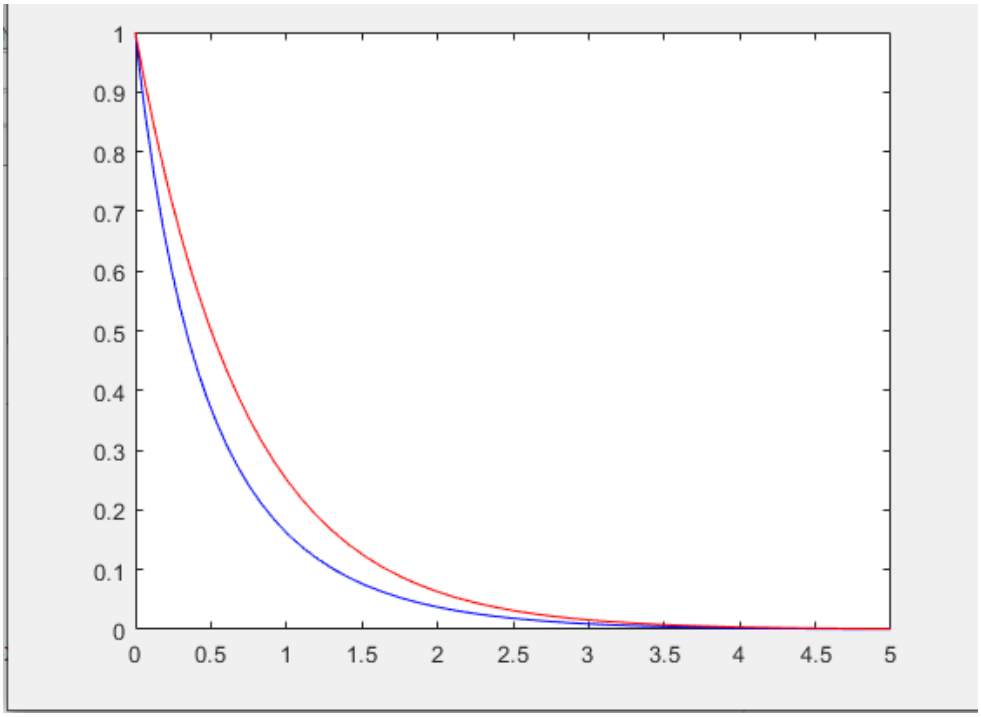
$$\forall t \geq 0,\ \|x(t) - x^*\mathbf{1}_n\| \leq e^{-\lambda_2 t}\,\|x(0) - x^*\mathbf{1}_n\|$$

Cela signifie que le consensus est atteint à vitesse exponentielle : la vitesse de convergence est déterminée par la 2è plus petite valeur propre λ2.

Le code Matlab pour vérifier cette propriété est le suivant :

```
l=eig(L);
l2=l(2);
Xet=mean(X0);
Y=zeros(size(t));
for i=1:length(t)
    Y(i)=(norm(x(i,:)-Xet*ones(1,5)))/(norm(X0-Xet*ones(5,1)));
end
figure
plot(t,Y,'b')
hold on
plot(t,exp(-t*l2),'r')
```

Et on obtient le résultat suivant, qui représente la convergence au cours du temps :



Exemple 2 :

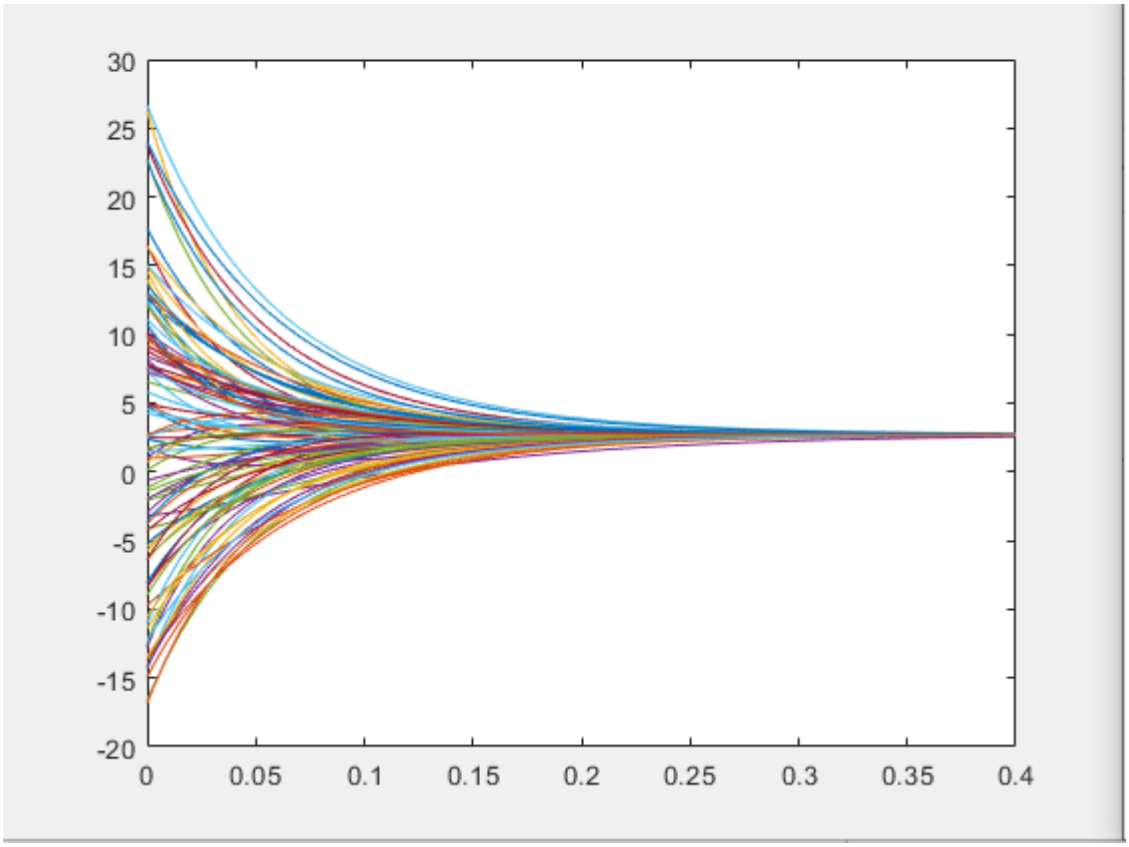
Dans ce deuxième exemple, nous considérons un réseau plus complexe, avec un nombre beaucoup plus élevé d'agents (n=100). On détermine alors les états initiaux x0 de manière aléatoire.

Le principe de l'algorithme de consensus reste le même, simplement ici la matrice laplacienne est calculée à partir des matrices A et D :

```
n=100;
X0=30*rand(n,1)-20*rand(n,1);
A=rand(n,n);
S=0.89;
for i=1:n
    for j=1:n
        if A(i,j)<S
            A(i,j)=0;
        else A(i,j)=1;
        end
    end
    A(i,i)=0;
end
A=A+A';
for i=1:n
    for j=1:n
      if A(i,j)==2
          A(i,j)=1;
      end
    end
end
D=zeros(n);
for i=1:n
    D(i,i)=sum(A(i,:));
end
L=D-A;

tspan = [0 0.4];
[t,x] = ode23(@(t,x)  -L*x, tspan, X0);
figure
plot(t,x)
```

Le résultat obtenu est le suivant :

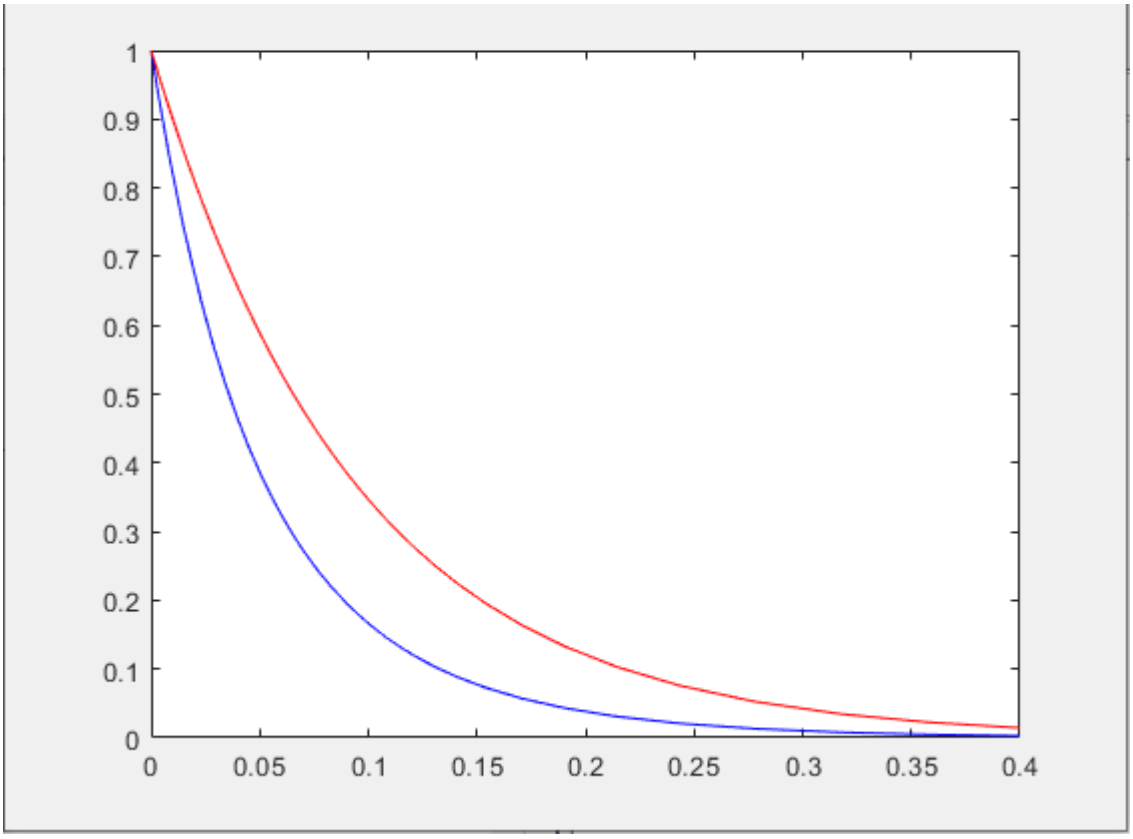


Ainsi, malgré un nombre élevé d'agents, un consensus est rapidement trouvé autour de la valeur moyenne des états initiaux x0. On peut alors tester plusieurs fois l'algorithme et constater que plus le nombre d'agents est élevé, plus le consensus est trouvé rapidement.
De la même façon, plus S est faible, plus cette vitesse de convergence sera importante.

Toutefois, il faut faire attention à ce que le graphique créé soit toujours connexe, sinon il ne sera pas possible de trouver un consensus. Pour cela, il faut éviter d'avoir une valeur de S trop élevée et une valeur de n trop faible.

Enfin, on peut vérifier la validité de la solution trouvée en vérifiant le théorème de convergence énoncé précédemment :

```
l=eig(L); l2=l(2); Xet=mean(X0); Y=zeros(size(t));
for i=1:length(t)
    Y(i)=(norm(x(i,:)-Xet*ones(1,n)))/(norm(X0-Xet*ones(n,1)));
end
figure
plot(t,Y,'b')
hold on
plot(t,exp(-t*l2),'r')
```

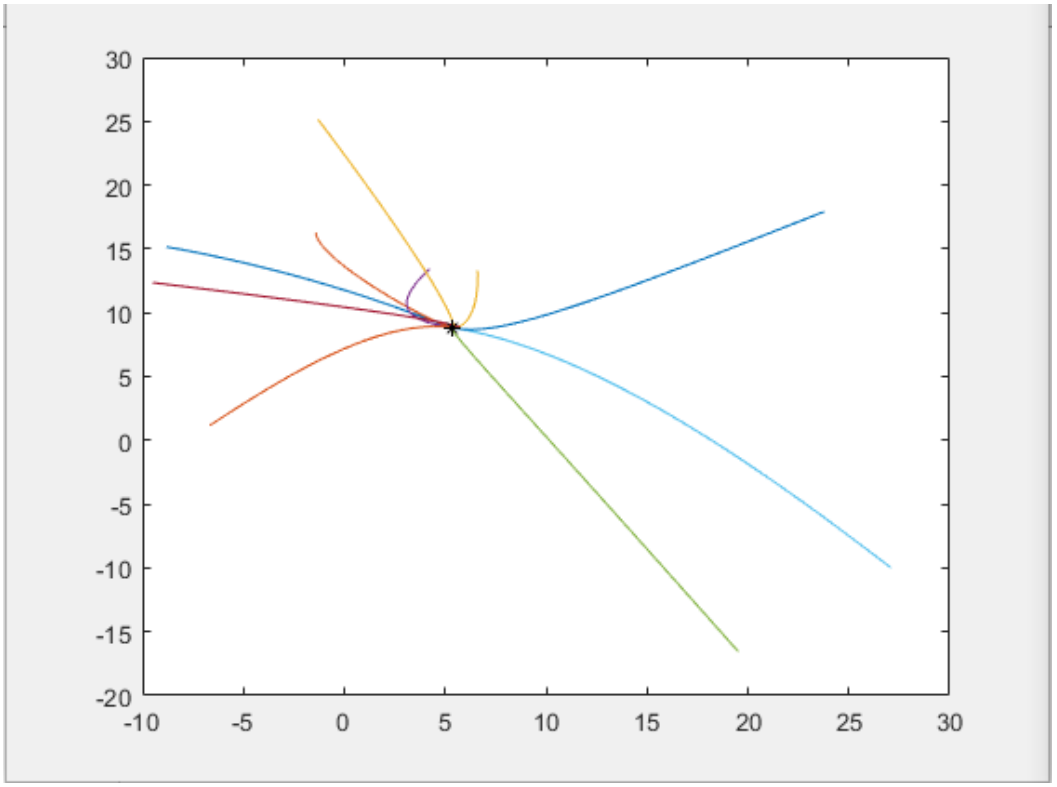


Exemple 3 :

Dans ce troisième exemple, les agents ne sont plus seulement définis par leur état x, mais également par un état y :

```
n=10; X0=30*rand(n,1)-20*rand(n,1); Y0=30*rand(n,1)-20'
A=rand(n,n); S=0.6;
for i=1:n
    for j=1:n
        if A(i,j)<S
            A(i,j)=0;
        else A(i,j)=1; end
    end
    A(i,i)=0;
end
A=A+A';
for i=1:n
    for j=1:n
      if A(i,j)==2
          A(i,j)=1;
      end
    end
end
D=zeros(n);
for i=1:n
    D(i,i)=sum(A(i,:));
end
L=D-A; tspan = [0 5];
[t,x] = ode23(@(t,x)  -L*x, tspan, X0);
[t,y] = ode23(@(t,y)  -L*y, t, Y0);
figure
plot(x,y)
hold on
xet=mean(X0);
yet=mean(Y0);
plot(xet,yet,'k*')
```

Ainsi, la convergence ne se fera plus selon l'état xi de chaque agent, mais en fonction des états xi et yi. De cette manière, le consensus trouvé ne se fera autour d'une valeur x* au cours du temps, mais autour d'un point, défini dans un espace par ses coordonnées x* et y*, qui correspondent respectivement aux moyennes des états initiaux x0 et y0 :



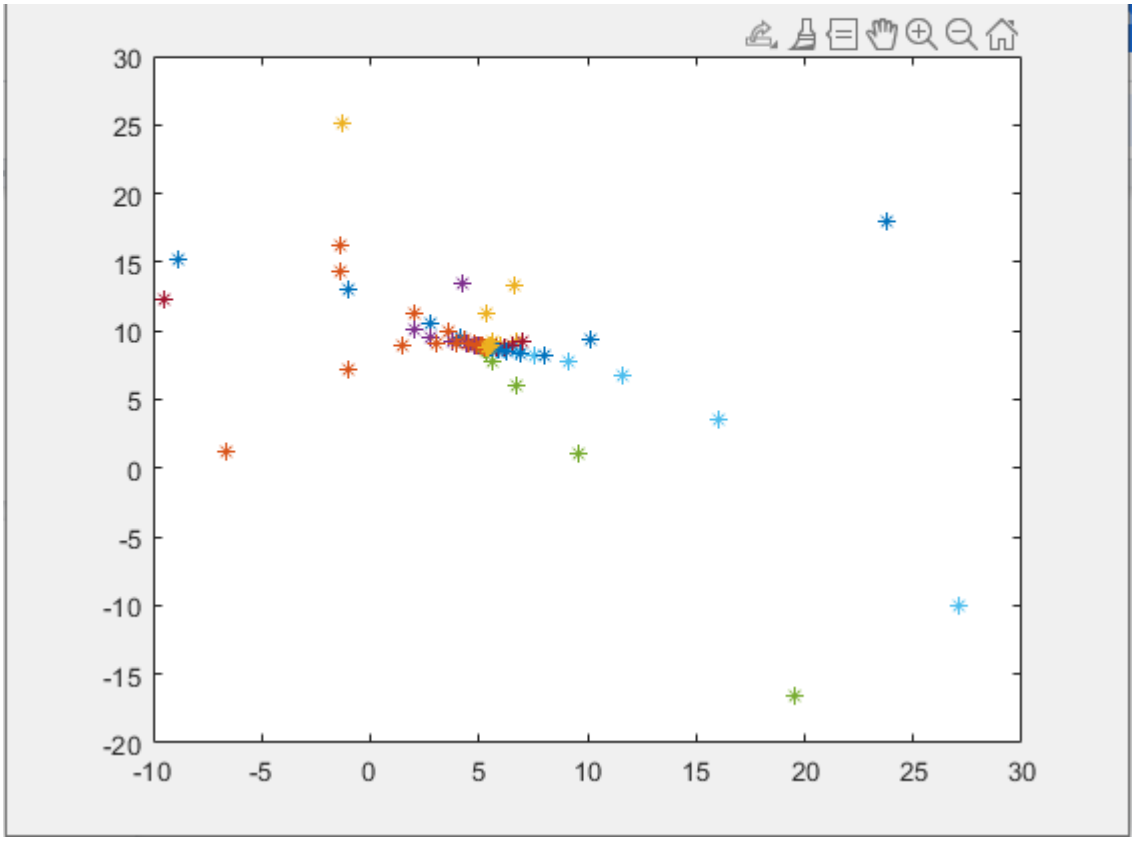
On peut également décider de tester l'algorithme de consensus en temps discret. Celui-ci sera alors dirigé par la loi suivante :

$$x(t+1) = (I - \varepsilon L(t))x(t)$$

On aura donc besoin d'introduire une variable ε, correspondant à la fréquence d'échantillonnage :

```
Eps=1/max(max(D));
P=eye(n)-Eps*L;
X=[X0];
Y=[Y0];
iter=1;
while norm([X0;Y0]-[xet*ones(n,1);yet*ones(n,1)])>0.01
    X0=P*X0;
    Y0=P*Y0;
    X=[X X0];
    Y=[Y Y0];
end
figure
plot (X',Y','*')
```

Les résultats obtenus sont similaires à ceux obtenus en temps continu :



Exemple 4 :

Dans ce dernier exemple, nous voulons nous intéresser à un exemple d'une masse d'agents (comme un groupe d'oiseaux par exemple), qui se déplacent à une certaine vitesse, en gardant un alignement et une cohésion entre eux.

Ici, ces agents sont caractérisés par leurs coordonnées dans le plan x et y, ainsi que leur vitesse v :

```
n=20;
X0=30*rand(n,1)-20*rand(n,1);
Y0=30*rand(n,1)-20*rand(n,1);
VX0=30*rand(n,1)-20*rand(n,1);
VY0=30*rand(n,1)-20*rand(n,1);
Z0=[X0;VX0];
W0=[Y0;VY0];
A=rand(n,n);
S=0.7;
for i=1:n
    for j=1:n
        if A(i,j)<S
            A(i,j)=0;
        else A(i,j)=1;
        end
    end
    A(i,i)=0;
end
A=A+A';
for i=1:n
    for j=1:n
      if A(i,j)==2
          A(i,j)=1;
      end
    end
end
D=zeros(n);
for i=1:n
    D(i,i)=sum(A(i,:));
end
L=D-A;
I=eye(n);
l=[zeros(n) eye(n) ;zeros(n) -L];
tspan = [0 5];
[t,z] = ode23(@(t,z) l*z, tspan, Z0);
[t,w] = ode23(@(t,w) l*w, t, W0);
x=z(:,1:n);
y=w(:,1:n);
figure
plot(x,y)
```

Dans ce cas, le consensus consiste ici à ce que chaque agent calque sa vitesse sur celle de ses voisins, afin que le groupe se déplace de manière unie, où chaque agent maintient toujours les mêmes distances avec ses voisins :