Rapport TP

LOCALISATION ET CARTOGRAPHIE SIMULTANÉES EN ROBOTIQUE (SLAM)

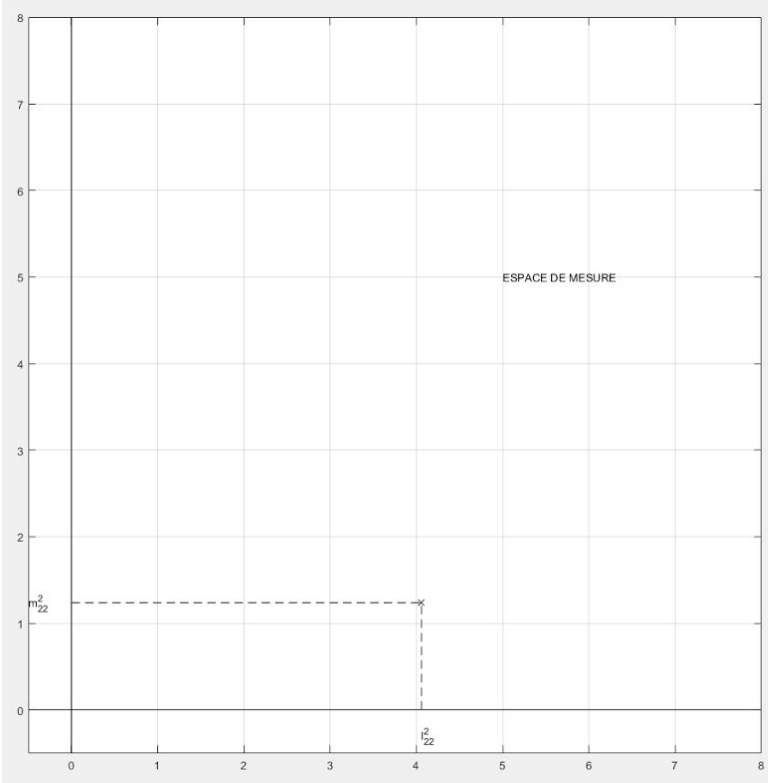
S9 -2017

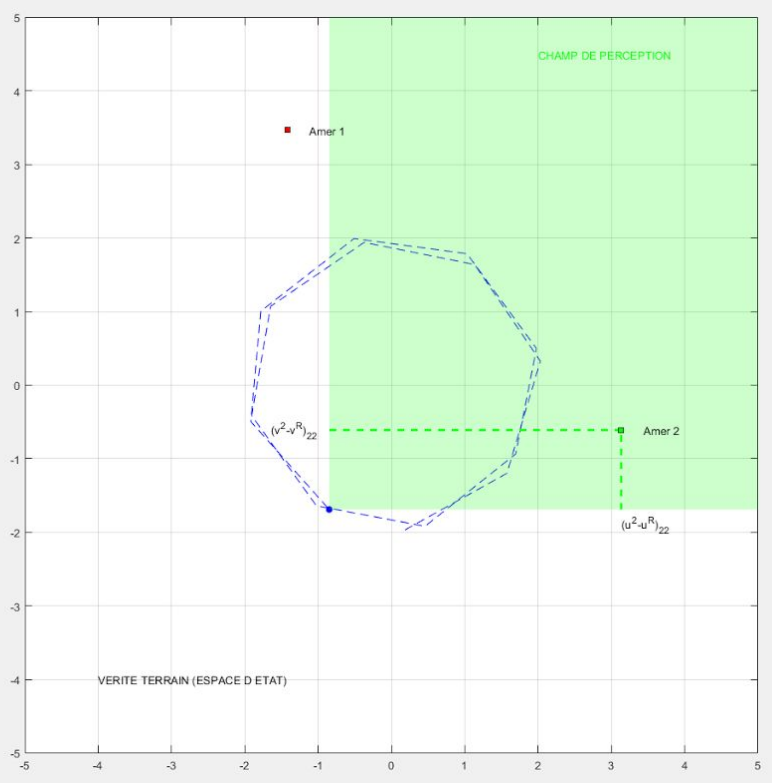
Yoann Fleytoux
Aurélien Bernier

Sommaire

Simulation – Ecriture du modèle

1. Exécuter un “run” de la fonction simulationDonnees .





(a) Expliquer les informations fournies par l’animation graphique.

La fonction simulationDonnees affiche deux animations de graphes:

- L’évolution de la position du robot, son champ de perception, et la position des deux amers dans le repère monde.
- L’espace de mesure, c’est à dire, les différentes mesures observés par le robot (m1,k ; m2,k ; l1,k ; l2,k), càd, les distance en x et y entre les amers et le robot.

(b) Expliquer le contenu des variables T , Z , X .

“ T : vecteur des instants d’echantillonnage (1xN)” : vecteur d’entier allant de 0 à N*deltaT par saut de deltaT (ici 1).

“ Z : réalisation du processus aléatoire de mesure (4xN, éventuellement avec composantes de type NaN)“ : vecteur contenant les observations (mesures) à chaque instant t; contient donc les distances (m1,k ; m2,k ; l1,k ; l2,k), si l’amer n’est pas dans le champ de perception, les mesure sont mises à NaN.

“X : réalisation du processus aléatoire d’ etat (6xN)” : vecteur contenant la position du robot et des amers à chaque instants t ;
X=[position en x du robot dans le repère monde;
position en y du robot dans le repère monde;
Position en x de l’amer 1 dans le repère du robot;
Position en y de l’amer 1 dans le repère du robot;
Position en x de l’amer 2 dans le repère du robot;
Position en y de l’amer 2 dans le repère du robot]

(c) Donner un sens physique à mX0 , P0 , Qw , Rv .

“mX0 : espérance du vecteur d’état à l’instant 0 (6x1)”
“PX0 : covariance du vecteur d’ état `a l’instant 0 (6x6)”
Permettent de générer la position du robot à l’instant t0.
“Qw : covariance du bruit de dynamique (supposé stationnaire) (6x6)” : représente le bruit sur les déplacement du robot et des amers (très faible dans le cas des amers).
“Rv : covariance du bruit de mesure (supposé stationnaire) (4x4)” : représente le bruit sur les mesures des capteurs.

2. Etablir l’expression de

(a) la distribution initiale p (x 0) ;

p(x0) ~N(x0;mX0,P0)
p(x0) = 1/((2*pi)^N * |P0|) * exp(-½*(x0 - mX0)^T * P0^-1 *(x0 - mX0))

(b) la loi de dynamique a priori p (x k | x k − 1) ;

p (x k | x k − 1) ~ N(xk; F*xk-1, Qk-1)
F la matrice du modèle liée à l’évolution de la dynamique du vecteur d’état.

(c) le modèle d’observation p (z k | x k) (selon la visibilité des amers).

p (z k | x k) ~ N(zk; H*xk,Rk)
H est la matrice du modèle liée à la mesure.

Implémentation d’un filtre particulaire SIS

3. Ecrire l’algorithme du filtre particulaire “Sequential Importance Sampling” (SIS) pour le problème considéré.

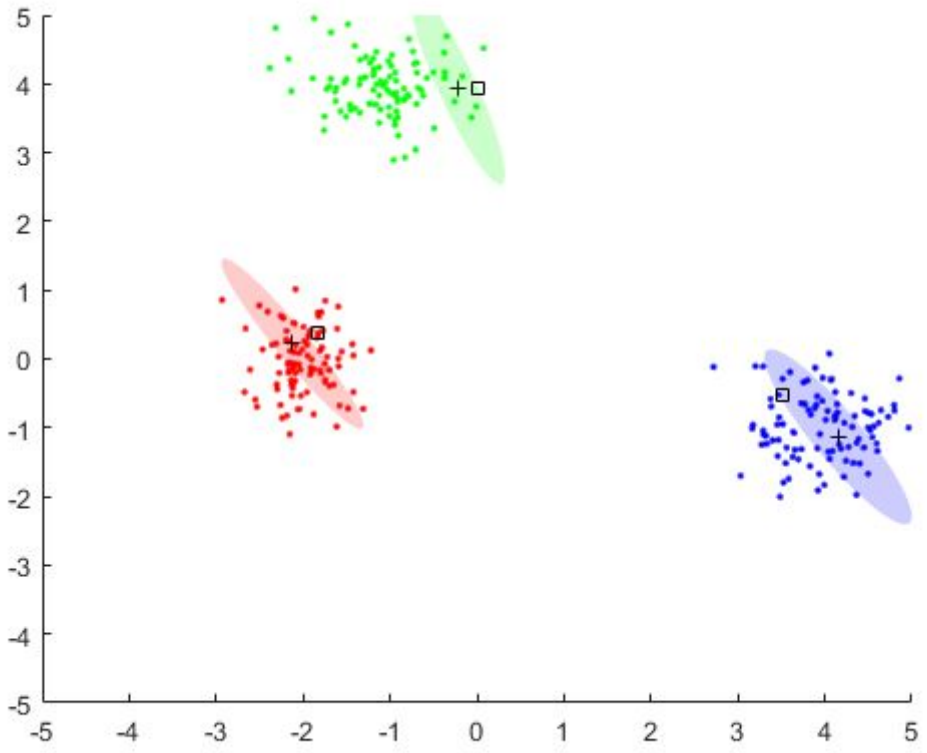


Figure 1: affichage du SIS

En rouge ce qui se rapporte au robot, en vert à l’amer 1 et en bleue à l’amer 2

Affichage des estimes des moyennes a posteriori (+ noirs)

Affichage des des ellipses de confiance (fonction de l’espérance et de la covariance)

Affichage du vecteur d'état réel (carrés noirs)

4. Procéder à son implémentation pour l'initialisation ainsi que les modèles de bruits identiques à ceux exploités dans simulationDonnees.

(a) Comment sélectionner le nombre de particules ? Plusieurs choix pourront être testés.

Le nombre de particules choisies influencera le temps de calcul nécessaire et la capacité du filtrage particulaire à converger. Plus le nombre de particules est grand, plus les calculs sont longs, mais plus on convergera vite.

(b) Comment définir la fonction d'importance ? Plusieurs choix pourront être testés.

On choisit la loi de dynamique à priori.

(c) Quelles difficultés observe-t-on ? Comment les résoudre ?

Au bout d'un certain nombre d'itérations, on rencontre un problème de dégénération (une particule avec un poids proche de 1, et le reste des particules avec un poids proche de zéro) qui sera réglé à l'implémentation du filtre particulaire SIR. On résoudra ce problème à l'aide d'un rééchantillonnage.

5. Exécuter le filtre sur la base d'une réalisation du processus de mesure.

(a) Déduire les estimés des moyennes et covariances a posteriori ^xk|k et Pk|k du vecteur d'´etat xk, conditionnellement à z1:k = z1:k. Observer l'évolution de ces quantités.

```
%Calcul de la moyenne a posteriori
esperance = zeros(6, 1);
for i = 1 : nbParticules
        esperance = esperance + wPoids(i)*ParticulesNew(:,i);
End

%Calcul de la covariance a posteriori
covariance = zeros(6,6);
```

```
for i = 1 : nbParticules
        covariance = covariance +
        wPoids(i)*(ParticulesNew(:,i)-esperance)*(ParticulesNew(:,i)-esperance)';
End
```

(b)

Voir la Figure 1: affichage du SIS

(c) Vérifier la dégénérescence du nuage par calcul de la taille efficace du N-échantillon.

```
seuil = nbParticules/3;

 %Reechantillonage pour eviter la dégénerescence
Neff = 1/sum(wPoids.^2);
if Neff<seuil
        'Resample'
        [ParticulesNew,wPoids] = Resample(nbParticules, ParticulesNew, wPoids);
End
```

Implémentation d’un filtre particulaire SIR

6. Compléter l’algorithme précédent par une étape de rééchantillonnage du nuage, lorsque le nombre de particules efficaces se situe en deçà d’un certain seuil à définir, de façon à obtenir un algorithme de “Sampling Importance Resampling” (SIR).

7. Tester ce nouvel algorithme.

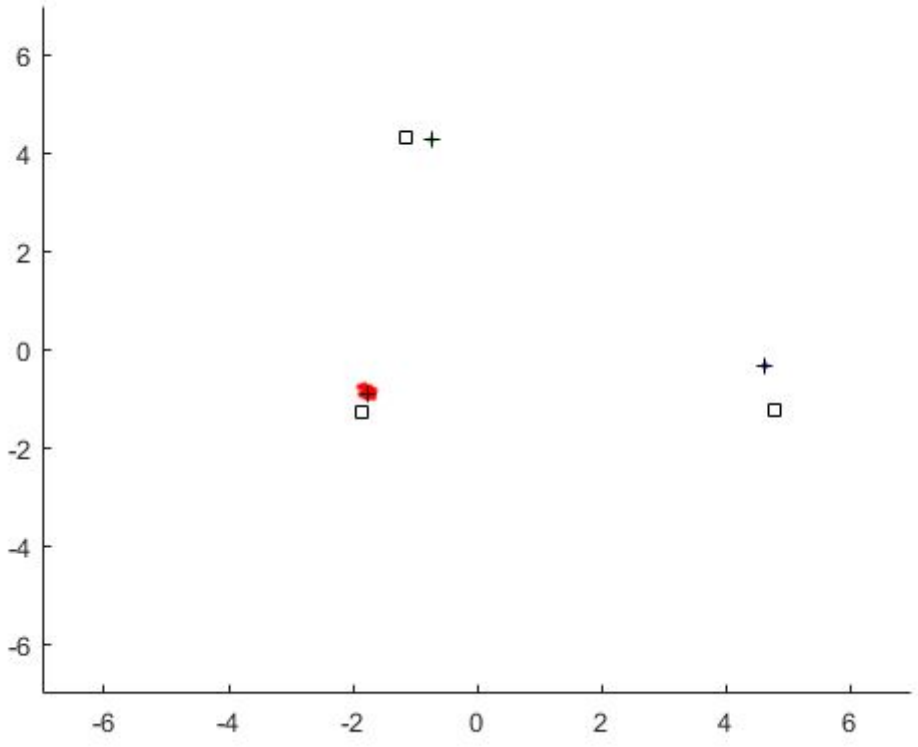


Figure 2: affichage du SIR

En rouge ce qui se rapporte au robot, en vert à l’amer 1 et en bleue à l’amer 2

Affichage des estimes des moyennes a posteriori (+ noirs)

Affichage des des ellipses de confiances (fonction de l’espérance et de la covariance)

Affichage du vecteur d'état réel (carrés noirs)

8. Conclure sur l'adéquation / la non-adéquation du filtrage particulaire pour le problème considéré.

On rééchantillonne très souvent ce qui laisse à penser que la fonction d'importance n'est pas adéquate. L'estimé du vecteur d'état ne converge pas vers leurs valeurs réels (ce qui peut notamment être dû au manque de dynamisme au niveau des amers). On en conclut qu'à moins de trouver une meilleur fonction d'importance, le filtrage particulaire n'est pas approprié au problème considéré.

Note: en introduisant du dynamisme au niveau des amers :

"Qw : covariance du bruit de dynamique (supposé stationnaire) (6x6)" :
Qw =

0.0025	0	0	0	0	0
0	0.0025	0	0	0	0
0	0	0.0000	0	0	0
0	0	0	0.0000	0	0
0	0	0	0	0.0000	0
0	0	0	0	0	0.0000

en :

Qw =

0.0025	0	0	0	0	0
0	0.0025	0	0	0	0
0	0	0.0025	0	0	0
0	0	0	0.0025	0	0
0	0	0	0	0.0025	0
0	0	0	0	0	0.0025

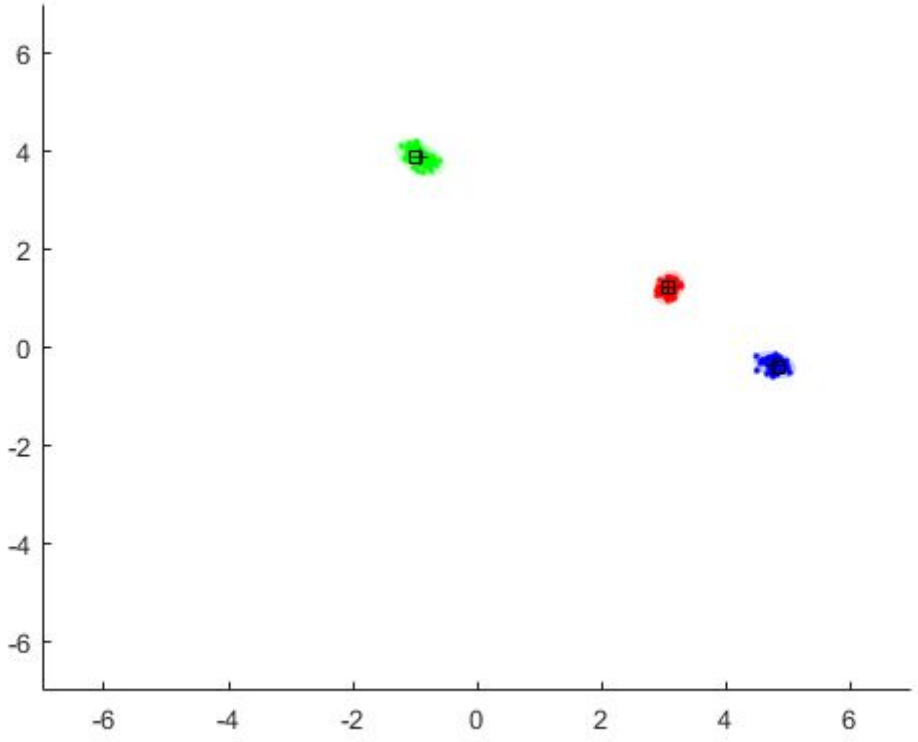


Figure 3: affichage du SIR avec dynamisme amers au bout de 50 itérations

En rouge ce qui se rapporte au robot, en vert à l'amer 1 et en bleue à l'amer 2

Affichage des estimes des moyennes a posteriori (+ noirs)

Affichage des des ellipses de confiances (fonction de l'espérance et de la covariance)

Affichage du vecteur d'état réel (carrés noirs)

On observe que l'estimé du vecteur d'état converge vers leurs valeurs réels.

Code:

SimulationDonnees.m

```
function [N,T,Z,F,Hfull,mX0,PX0,Qw,Rv,X] = simulationDonnees(plot_p);
%SIMULATIONDONNEES
%  Simulation d'une expérience : déplacement du robot, recueil des mesures, etc.
%  Syntaxe d'appel : [N,T,Z,F,Hfull,mX0,PX0,Qw,Rv,X] = simulationDonnees(plot_p);
%
%  Entrée :
%  . plot_p : si égal à 1, produit un affichage animé, sinon n'affiche rien
%
%  Sorties :
%  . N : nombre d'échantillons temporels
%  . T : vecteur des instants d'échantillonnage (1xN)
%  . Z : réalisation du processus aléatoire de mesure (4xN, éventuellement avec
composantes de type NaN)
%  -> et comme ceci est de la simulation, sont également accessibles
%        . F, Hfull : matrices du modèle (respectivement (6x6), (4x6))
%        . mX0 : espérance du vecteur d'état à l'instant 0 (6x1)
%        . PX0 : covariance du vecteur d'état à l'instant 0 (6x6)
%        . Qw  : covariance du bruit de dynamique (supposé stationnaire) (6x6)
%        . Rv  : covariance du bruit de mesure (supposé stationnaire) (4x4)
%        . X   : réalisation du processus aléatoire d'état (6xN)
%
%  Z et X admettent AUTANT DE COLONNES QUE D'INSTANTS

N = 50;
deltaT = 1;
T = [0:N-1]*deltaT;
w=pi/4;

mX0 = [2;0;-1;4;4;-1];
PX0 = diag([.1 .2 .2 .2 .2 .2]);
Qw = diag([.05^2 .05^2 (1e-10)^2 (1e-10)^2 (1e-10)^2 (1e-10)^2]);
Rv = diag([.1^2 .1^2 .1^2 .1^2]);

F = blkdiag([cos(w*deltaT) -sin(w*deltaT) ; sin(w*deltaT) cos(w*deltaT)], eye(2), eye(2));

H1=[-1 0 1 0 0 0; 0 -1 0 1 0 0];
H2=[-1 0 0 0 1 0 ; 0 -1 0 0 0 1];
H=[H1;H2]; Hfull=H;
X = nan*ones(6,N);
Z = nan*ones(4,N) ;
```

```
Source1Vue=zeros(1,N);
Source2Vue=zeros(1,N);

% Instant 0
W = chol(Qw)'*randn(6,N);
V = chol(Rv)'*randn(4,N);
X(:,1) = mX0+chol(PX0)'*randn(6,1);
% X = randn(n) returns an n-by-n matrix of normally distributed random numbers.

% R = chol(A) produces an upper triangular matrix R from the diagonal
% and upper triangle of matrix A, satisfying the equation R'*R=A.
% The chol function assumes that A is (complex Hermitian) symmetric.
% If it is not, chol uses the (complex conjugate) transpose of the upper triangle as the lower
triangle.
% Matrix A must be positive definite.

% Instants 1:(N-1);
for k=2:N
  X(:,k) = F*X(:,k-1) + W(:,k-1);

  if ((X(3,k)-X(1,k))>0)&&((X(4,k)-X(2,k))>0)
          Source1Vue(k)=1;
  end
  if ((X(5,k)-X(1,k))>0)&&((X(6,k)-X(2,k))>0)
          Source2Vue(k)=1;
  end

  if (Source1Vue(k)==1)&&(Source2Vue(k)==1)
          Z(:,k) = H*X(:,k) + V(:,k);
  elseif (Source1Vue(k)==1)&&(Source2Vue(k)==0)
          Z(:,k) = [H1*X(:,k) + V(1:2,k);nan(2,1)];
  elseif (Source1Vue(k)==0)&&(Source2Vue(k)==1)
          Z(:,k) = [nan(2,1);H2*X(:,k) + V(3:4,k)];
  else
          Z(:,k)=[nan(4,1)];
  end

  Z;
  X;
end;

if plot_p==1

scrsz=get(0,'ScreenSize');
currentfigure=figure('Position',[1 1 scrsz(3) scrsz(4)]);
nfig=1;
```

```
set(nfig,'PaperPositionMode','auto');

for k=1:N
        h=subplot(1,2,2);
        set(h,'position',[0.53 0.05 .45 .92]);
        plot(X(1,k),X(2,k),'o','markeredgecolor','none','markerfacecolor','b');
        hold on
        grid on
        text(-4,-4,'VERITE TERRAIN (ESPACE D ETAT)')
        plot(X(1,max(k-15,1):k),X(2,max(k-15,1):k),'--b');
        visib_domain=[X(1,k) 5 5 X(1,k); X(2,k) X(2,k) 5 5];
        patch(visib_domain(1,:),visib_domain(2,:),'g','edgecolor','none','facealpha',.2);
        text(X(3,k)+.3,X(4,k),'Amer 1')
        text(X(5,k)+.3,X(6,k),'Amer 2')
        text(2,4.5,'CHAMP DE PERCEPTION','color','g')
        if Source1Vue(k)==1
        color='g';
        plot([X(3,k) X(3,k)],[X(2,k) X(4,k)],'--g','linewidth',2);
        text(X(3,k),X(2,k)-.2,sprintf('%s%s%s%s','(u^1-u^R)_','{',int2str(k),'}'));
        plot([X(3,k) X(1,k)],[X(4,k) X(4,k)],'--g','linewidth',2);
        text(X(1,k)-.8,X(4,k),sprintf('%s%s%s%s','(v^1-v^R)_','{',int2str(k),'}'));
        else
        color='r';
        end
        plot(X(3,k),X(4,k),'s','markeredgecolor','k','markerfacecolor',color);
        if Source2Vue(k)==1
        color='g';
        plot([X(5,k) X(5,k)],[X(2,k) X(6,k)],'--g','linewidth',2);
        text(X(5,k),X(2,k)-.2,sprintf('%s%s%s%s','(u^2-u^R)_','{',int2str(k),'}'));
        plot([X(5,k) X(1,k)],[X(6,k) X(6,k)],'--g','linewidth',2);
        text(X(1,k)-.8,X(6,k),sprintf('%s%s%s%s','(v^2-v^R)_','{',int2str(k),'}'));
        else
        color='r';
        end
        plot(X(5,k),X(6,k),'s','markeredgecolor','k','markerfacecolor',color);
        axis equal
        xlim([-5 5]);
        ylim([-5 5]);
        hold off

        g=subplot(1,2,1);
        set(g,'position',[0.03 0.05 .45 .92]);
        plot(Z(1,k),Z(2,k),'xk','markersize',7)
        hold on
        grid on
        text(5,5,'ESPACE DE MESURE')
```

```
        plot([0 0],[-.5 8],'k')
        plot([-.5 8],[0 0],'k')
        plot([0 Z(1,k)],[Z(2,k) Z(2,k)],'--k');
        plot([Z(1,k) Z(1,k)],[0 Z(2,k)],'--k');
        plot(Z(3,k),Z(4,k),'xk','markersize',7)
        plot([0 Z(3,k)],[Z(4,k) Z(4,k)],'--k');
        plot([Z(3,k) Z(3,k)],[0 Z(4,k)],'--k');
        if Source1Vue(k)==1
        text(Z(1,k),-.3,sprintf('%s%s%s%s','l^1_','{',int2str(k),'}'));
        text(-.5,Z(2,k),sprintf('%s%s%s%s','m^1_','{',int2str(k),'}'));
        end
        if Source2Vue(k)==1
        text(Z(3,k),-.3,sprintf('%s%s%s%s','l^2_','{',int2str(k),'}'));
        text(-.5,Z(4,k),sprintf('%s%s%s%s','m^2_','{',int2str(k),'}'));
        end
        axis equal
        xlim([-.5 8]);
        ylim([-.5 8]);
        hold off

        drawnow;
        pause(deltaT)
end

end
```

SIS.m

```
clear all; close all;

%on recupere les donnees
[N,T,Z,F,Hfull,mX0,PX0,Qw,Rv,X] = simulationDonnees(0);

% nombres de particules
nbParticules = 100;

% Initialisation des poids des particules
wPoids = ones(1, nbParticules) .* (1 / nbParticules);

% Generation des particules au premier instant

% R = chol(A) produces an upper triangular matrix R from the diagonal
% and upper triangle of matrix A, satisfying the equation R'*R=A.
% The chol function assumes that A is (complex Hermitian) symmetric.
% If it is not, chol uses the (complex conjugate) transpose of
% the upper triangle as the lower triangle. Matrix A must be positive definite.

% B = repmat(A,r1,...,rN) specifies a list of scalars, r1,..,rN, that describes
% how copies of A are arranged in each dimension.
% When A has N dimensions, the size of B is size(A).*[r1...rN].
% For example, repmat([1 2; 3 4],2,3) returns a 4-by-6 matrix.

Particules = chol(PX0)'*randn(6, nbParticules) + repmat(mX0, 1, nbParticules)
ParticulesNew = zeros(6,nbParticules);

hold on;

% Pour chaque instant de l'echantillonnage
for k = 2:N
        % Pour chaque particule
 % Mise a  jour des poids
        amers=1%
        if(isnan(Z(1,k)) && isnan(Z(3,k)))%on ne voit ni l'amer 1 ni la 2
        %si amers==0, pas de mise a jour des poids car pas de mesure
        amers=0
        elseif(isnan(Z(1,k)))%on ne voit pas l'amer 1
        H=Hfull(3:4, :);
        R=Rv(3:4, 3:4);
        ZCurrent=Z(3:4,k);
        elseif(isnan(Z(3,k)))%on ne voit pas l'amer 2
```

```
        H=Hfull(1:2, :);
        R=Rv(1:2, 1:2);
        ZCurrent=Z(1:2,k);
        else %on voit les deux amers
        H=Hfull;
        R=Rv;
        ZCurrent=Z(:,k);
        end
        if(amers == 1)
        for i=1:nbParticules
        % Propagation de la particule xk-1 en xk
        ParticulesNew(:,i) = chol(Qw)'*randn(6, 1)+(F*Particules(:,i));
        % Calcul du poids si au moins un amer est percu
        wPoids(:,i) = wPoids(:,i) * (1/sqrt(det(2*pi*R))) * exp(-(1/2) *
(ZCurrent-H*ParticulesNew(:,i))' * inv(R) * (ZCurrent-H*ParticulesNew(:,i)));
        end
        else
        for i=1:nbParticules
        % Propagation de la particule xk-1 en xk
        ParticulesNew(:,i) = chol(Qw)'*randn(6, 1)+(F*Particules(:,i));
        end
        end

        %Normalisation des poids
        wPoids(:,:) = wPoids ./ sum(wPoids);

        %Calcul de la moyenne a posteriori
        esperance = zeros(6, 1);
        for i = 1 : nbParticules
        esperance = esperance + wPoids(i)*ParticulesNew(:,i);
        end

        %Calcul de la covariance a posteriori
        covariance = zeros(6,6);
        for i = 1 : nbParticules
        covariance = covariance +
wPoids(i)*(ParticulesNew(:,i)-esperance)*(ParticulesNew(:,i)-esperance)';
        end

        %Affichage
        axis([-7,7,-7,7]);

        %Affichage des particules (points)
        particulesPlots = [];
        for j=1:nbParticules
```

```
        particulesPlots = [particulesPlots,
plot(ParticulesNew(1,j),ParticulesNew(2,j),'.r'),plot(ParticulesNew(3,j),ParticulesNew(4,j),'.g'),
plot(ParticulesNew(5,j),ParticulesNew(6,j),'.b')];
        end

        %Affichage des estimes des moyennes a posteriori (+ noirs)
        P1 = plot(esperance(1),esperance(2),'+k');
        P2 = plot(esperance(3),esperance(4),'+k');
        P3 = plot(esperance(5),esperance(6),'+k');

        %Affichage des des ellipses de confiances
        E1 = ellipse(esperance(1:2), covariance(1:2,1:2), 'r');
        E2 = ellipse(esperance(3:4), covariance(3:4,3:4), 'g');
        E3 = ellipse(esperance(5:6), covariance(5:6,5:6), 'b');

        %Affichage du vecteur d'etat reel (carres noirs)
        P1r = plot(X(1,k),X(2,k),'sk');%robot
        P2r = plot(X(3,k),X(4,k),'sk');%amer1
        P3r = plot(X(5,k),X(6,k),'sk');%amer2

        pause(0.2);

        if(k~=N)
        delete([E1, P1, E2, P2, E3, P3,P1r,P2r,P3r]);
        delete(particulesPlots);
        end
        k
        Particules=ParticulesNew;
end
```

SIR_v1.m

```
clear all; close all;

%on recupere les donnees
[N,T,Z,F,Hfull,mX0,PX0,Qw,Rv,X] = simulationDonnees(0);

% nombres de particules
nbParticules = 100;

seuil = nbParticules/3;
% Initialisation des poids des particules
wPoids = ones(1, nbParticules) .* (1 / nbParticules);

% Generation des particules au premier instant

% R = chol(A) produces an upper triangular matrix R from the diagonal
% and upper triangle of matrix A, satisfying the equation R'*R=A.
% The chol function assumes that A is (complex Hermitian) symmetric.
% If it is not, chol uses the (complex conjugate) transpose of
% the upper triangle as the lower triangle. Matrix A must be positive definite.

% B = repmat(A,r1,...,rN) specifies a list of scalars, r1,..,rN, that describes
% how copies of A are arranged in each dimension.
% When A has N dimensions, the size of B is size(A).*[r1...rN].
% For example, repmat([1 2; 3 4],2,3) returns a 4-by-6 matrix.

Particules = chol(PX0)'*randn(6, nbParticules) + repmat(mX0, 1, nbParticules);
ParticulesNew = zeros(6,nbParticules);

hold on;

% Pour chaque instant de l'echantillonnage
for k = 2:N
        % Pour chaque particule
 % Mise a  jour des poids
        amers=1;
        if(isnan(Z(1,k)) && isnan(Z(3,k)))%on ne voit ni l'amer 1 ni la 2
        %si amers==0, pas de mise a jour des poids car pas de mesure
        amers=0;
        elseif(isnan(Z(1,k)))%on ne voit pas l'amer 1
        H=Hfull(3:4, :);
        R=Rv(3:4, 3:4);
        ZCurrent=Z(3:4,k);
```

```
        elseif(isnan(Z(3,k)))%on ne voit pas l'amer 2
        H=Hfull(1:2, :);
        R=Rv(1:2, 1:2);
        ZCurrent=Z(1:2,k);
        else %on voit les deux amers
        H=Hfull;
        R=Rv;
        ZCurrent=Z(:,k);
        end
        if(amers == 1)
        for i=1:nbParticules
        % Propagation de la particule xk-1 en xk
        ParticulesNew(:,i) = chol(Qw)'*randn(6, 1)+(F*Particules(:,i));
        % Calcul du poids si au moins un amer est percu
        wPoids(:,i) = wPoids(:,i) * (1/sqrt(det(2*pi*R))) * exp(-(1/2) *
(ZCurrent-H*ParticulesNew(:,i))' * inv(R) * (ZCurrent-H*ParticulesNew(:,i)));
        end
        else
        for i=1:nbParticules
        % Propagation de la particule xk-1 en xk
        ParticulesNew(:,i) = chol(Qw)'*randn(6, 1)+(F*Particules(:,i));
        end
        end

        %Normalisation des poids
        wPoids(:,:) = wPoids ./ sum(wPoids);

        %Calcul de la moyenne a posteriori
        esperance = zeros(6, 1);
        for i = 1 : nbParticules
        esperance = esperance + wPoids(i)*ParticulesNew(:,i);
        end

        %Calcul de la covariance a posteriori
        covariance = zeros(6,6);
        for i = 1 : nbParticules
        covariance = covariance +
wPoids(i)*(ParticulesNew(:,i)-esperance)*(ParticulesNew(:,i)-esperance)';
        end

        %Reechantillonage pour eviter la degenerescence
        Neff = 1/sum(wPoids.^2);
        if Neff<seuil
        'Resample'
        [ParticulesNew,wPoids] = Resample(nbParticules, ParticulesNew, wPoids);
        end
```

```
        %Affichage
        axis([-7,7,-7,7]);

        %Affichage des particules (points)
        particulesPlots = [];
        for j=1:nbParticules
        particulesPlots = [particulesPlots,
plot(ParticulesNew(1,j),ParticulesNew(2,j),'.r'),plot(ParticulesNew(3,j),ParticulesNew(4,j),'.g'),
plot(ParticulesNew(5,j),ParticulesNew(6,j),'.b')];
        end

        %Affichage des estimes des moyennes a posteriori (+ noirs)
        P1 = plot(esperance(1),esperance(2),'+k');
        P2 = plot(esperance(3),esperance(4),'+k');
        P3 = plot(esperance(5),esperance(6),'+k');

        %Affichage des des ellipses de confiances
        E1 = ellipse(esperance(1:2), covariance(1:2,1:2), 'r');
        E2 = ellipse(esperance(3:4), covariance(3:4,3:4), 'g');
        E3 = ellipse(esperance(5:6), covariance(5:6,5:6), 'b');

        %Affichage du vecteur d'etat reel (carres noirs)
        P1r = plot(X(1,k),X(2,k),'sk');%robot
        P2r = plot(X(3,k),X(4,k),'sk');%amer1
        P3r = plot(X(5,k),X(6,k),'sk');%amer2

        pause(0.2);

        if(k~=N)
        delete([E1, P1, E2, P2, E3, P3,P1r,P2r,P3r]);
        delete(particulesPlots);
        end
        k
        Particules=ParticulesNew;
end
```