Annexes

I. Programme *Matlab* de suivi du robot au cours d'une expérience

```
%% Script suivi des images avec la webcam en temps

%cam = webcam('USB2.0');   % définit la camera à utiliser, USB2.0 est la webcam
utilisée au départ
%% Initialisation

nl = 480;
nc = 640;
nombre_robots = 1;
duree_simulation = 200;
rayon_robot = [15 25];

%grille = zeros(nl,nc); %taille des images obtenues par la caméra, grille vide

centers_abs = zeros(nombre_robots,duree_simulation);
centers_ord = zeros(nombre_robots,duree_simulation);

norme_vect = zeros(nombre_robots,1);
compteur_iteration = 1;

%% on prend une première photo pour capturer la position initiale des robots

img_rgb = snapshot(cam);
[centers, radii, metric] = imfindcircles(img_rgb,rayon_robot, ...
'ObjectPolarity','dark','Sensitivity',0.91,'Method','twostage');  %attention
rentrer le RAYON des cercles à detecter !!!!

for j = 1 : nombre_robots;  %on remplit initialement le tableau du point (0,0)
pour scaler l'image
    centers_abs(j,1) =  0;
    centers_ord(j,1) =  0;
end;


for j = 1 : nombre_robots;  %on remplit initialement le tableau des positions
pour ensuite faire la différence de position
    centers_abs(j,2) =  centers(j,1);
    centers_ord(j,2) =  centers(j,2);
end;

%% Détection en temps réel

for k = 2 : duree_simulation-1;
    compteur_iteration = k

    img_rgb = snapshot(cam);       % récupère une image de la webcam quand on le
veut
    [centers, radii1, metric] = imfindcircles(img_rgb,rayon_robot, ...
    'ObjectPolarity','dark','Sensitivity',0.91,'Method','twostage');

    [nligne,ncol] = size(centers);

    for robot = 1 : nombre_robots;

        if nligne ~= nombre_robots;         % si on n'a pas détecté autant de
cercles que robots on passe à la suite
            break;
        else;
```

```
            for test_vect = 1 : nombre_robots;  %on calcule la norme entre la
nouvelle position et les anciennes positions, on prend la minimum
                norme_vect(test_vect) = (centers(robot,1) -
centers_abs(test_vect,compteur_iteration))^2 + (centers(robot,2) -
centers_ord(test_vect,compteur_iteration))^2;
            end;
            [max,indice_min] = min(norme_vect);

            centers_abs(indice_min,compteur_iteration+1) = centers(robot,1);
            centers_ord(indice_min,compteur_iteration+1) = centers(robot,2);
        end;
    end;
    pause(3);

end;
```

II. Programme *Matlab* de calcul de l'écart moyen à l'origine du robot au cours d'une expérience

```
x = load('C:\Users\Chloé\Desktop\PSE robot\data\robot4_lum50_bis.txt'); %x
matrice des positions du robot
c = 0;
p= 10;   %temps d'expérience unitaire
RG = zeros(1,p);
TG = zeros(1,p);
X = zeros(1,floor(2000/p));
Y = zeros(1,floor(2000/p));
for k=3:p:1990;   %1 image toutes les 0,1s%
a = x(1,k:k+p)-x(1,k)
b = x(2,k:k+p)-x(2,k)
x2 = a.^2;
y2 = b.^2;
R = zeros(1,p);
T = zeros(1,p);
 for j=1:p;
     r2 = sum(x2(1,1:j))/length(x2(1,1:j)) + sum(y2(1,1:j))/length(y2(1,1:j));
     R(1,j) = r2;
     T(1,j) = j*0.1;
 end
X(1,k:k+p) = a;
Y(1,k:k+p) = b;
RG (1,1:end) = RG (1,1:end)+ R(1,1:end);
TG (1,1:end) = TG (1,1:end)+ T(1,1:end);
c = c+1;
end
TG=TG/c;
RG=RG/c;
plot(TG,RG,'o');
```

III. Programme *Java* de simulation du mouvement de 100000 particules dans un champ d'intensités de vibration variables

- Fonction « class paint » pour positionner les particules sur la fenêtre d'affichage à un instant donné

```
//Bibliothèques importées
import javax.swing.JPanel;
import java.awt.Color;
import java.awt.Graphics;
```

```
public class paint extends JPanel {
      public float[][]Evolution;
      public int hauteur;
      public int largeur;
      public int a;
public void paintComponent(Graphics g){
      super.paintComponent(g);
      for (int i=0;i<largeur;i++){
            for (int j=0;j<hauteur;j++){
                  if (Evolution[i][j]==1){    //la valeur 1 implique la présence
d'une particule
                        g.setColor(Color.RED);
                        g.fillRect(i, j, 3, 3);
                  }
            }
      }



}
}
```

- Fonction principale calculant le déplacement des particules au cours du temps selon l'environnement d'intensités de vibration défini

```
//Import des bibliothèques utiles
import java.util.Observable;
import javax.imageio.ImageIO;
import javax.swing.JFrame;
import javax.swing.JPanel;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Color;
import java.util.concurrent.TimeUnit;
import java.awt.Image;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.awt.Robot;

public class dynamique {
       public static void main(String[] args) throws IOException{
            JFrame fenetre = new JFrame();
          fenetre.setTitle("Dynamique");
            int [][] Pattern;   //Matrice du champ d'intensités de vibration
            float [][] Evolution; //Matrice des particules évoluant selon
Pattern
            //Dimensions de l'environnement
            int largeur;
            int hauteur;
            int t;
            int Nombre_particule;
            int d_h;
            int d_l;
            int c;
            int a;
            int C;
            largeur = 700;
            hauteur = 700;
            c = 0;
            a=0;
            C=0;
```

```
      //Création de la fenêtre d'affichage de l'environnement des particules
          fenetre.setSize(700,700);
          fenetre.setLocationRelativeTo(null);
          fenetre.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
          fenetre.setResizable(false);
          JPanel pan = new JPanel();
          pan.setBackground(Color.WHITE);
          fenetre.setContentPane(pan);

            Pattern = new int [largeur][hauteur];
            Evolution= new float [largeur][hauteur];

      // définition du type de champ d'intensités de vibration
            for (int i = 0; i < largeur; i++) {
                for (int j = 0; j < hauteur; j++) {
                  ////Type fonction triangulaire
                  //if (c%2==0){
                        //Pattern[i][j] = j%100 ;
                  //}
                  //if (c%2==1){
                        //Pattern[i][j] = 100 - j%100;
                  //}
                  //if (j%100==99){
                  //c = c +1;
                  //}

                  ////Type gradient
                  //Pattern[i][j]=   j%700;

                  Evolution[i][j]=0; //valeur d'un élément determine présence ou
non d'une particule (0 ou 1)


                }
              }

            ////Champ de type 4puits de potentiel nul
            //for (int i=0;i<=largeur/4;i++){
                  //for(int j=0;j<=hauteur/4;j++){
                        //Pattern[i+largeur/4][j+hauteur/4] = i+j;
                        //Pattern[i+largeur/4][hauteur/4-j] = i+j;
                        //Pattern[largeur/4-i][j+hauteur/4] = i+j;
                        //Pattern[largeur/4-i][hauteur/4-j] = i+j;
                  //}
            //}
            //for (int i=0;i<largeur/4;i++){
                  //for(int j=0;j<=hauteur/4;j++){
                        //Pattern[i+3*largeur/4][j+hauteur/4] = i+j;
                        //Pattern[i+3*largeur/4][hauteur/4-j] = i+j;
                        //Pattern[3*largeur/4-i][j+hauteur/4] = i+j;
                        //Pattern[3*largeur/4-i][hauteur/4-j] = i+j;
                  //}
            //}
            //for (int i=0;i<=largeur/4;i++){
                  //for(int j=0;j<hauteur/4;j++){
                        //Pattern[i+largeur/4][j+3*hauteur/4] = i+j;
                        //Pattern[i+largeur/4][3*hauteur/4-j] = i+j;
                        //Pattern[largeur/4-i][j+3*hauteur/4] = i+j;
                        //Pattern[largeur/4-i][3*hauteur/4-j] = i+j;
                  //}
            //}
            //for (int i=0;i<largeur/4;i++){
                  //for(int j=0;j<hauteur/4;j++){
                        //Pattern[i+3*largeur/4][j+3*hauteur/4] = i+j;
```

```
                        //Pattern[i+3*largeur/4][3*hauteur/4-j] = i+j;
                        //Pattern[3*largeur/4-i][j+3*hauteur/4] = i+j;
                        //Pattern[3*largeur/4-i][3*hauteur/4-j] = i+j;
                  //}
            //}

            Nombre_particule = 100000; //choix du nombre de particules
            //Placement aléatoire des particules à t=0
            for (int i=1;i<=Nombre_particule;i++){
                  Evolution[(int)(largeur*Math.random ())][(int)
(hauteur*Math.random ())]=1;
            }

            t = 0;
            //Evolution temporelle en coninu
            while (t==0){
                  C= 1 +C;
                  for (int i = 0; i < largeur; i++){
                        for (int j = 0; j < hauteur; j++) {
                              if (Evolution[i][j]==1){

                        //Calcul du déplacement (direction aléatoire, longueur
du saut selon la valeur de l'intensité de vibration à la position de la
particule
                              d_l = i + (int)Pattern[i][j]/10*(1-
2*(int)Math.round(Math.random ()));
                              d_h = j + (int)Pattern[i][j]/10*(1-
2*(int)Math.round(Math.random()));

                        //Traitement des conditions au bord : type Rebond
                        if (d_l<largeur && d_l>0 && d_h<hauteur && d_h>=0){
                                    Evolution[i][j]=0;
                                    Evolution[d_l][d_h]=1;
                              }
                        if (d_h>=hauteur){
                              if (d_l<largeur && d_l>=0){
                                    Evolution[i][j]=0;
                                    Evolution[d_l][hauteur-1-(d_h-hauteur)]=1;
                              }
                              if (d_l>=largeur){
                                    Evolution[i][j]=0;
                                    Evolution[largeur-1-(d_l-largeur)][hauteur-
1-(d_h-hauteur)]=1;
                              }
                              if (d_l<0){
                                    Evolution[i][j]=0;
                                    Evolution[-d_l][hauteur-1-(d_h-hauteur)]=1;
                              }
                        }
                        if (d_h<0){
                              if (d_l<largeur && d_l>=0){
                                    Evolution[i][j]=0;
                                    Evolution[d_l][-d_h]=1;
                              }
                              if (d_l>=largeur){
                                    Evolution[i][j]=0;
                                    Evolution[largeur-1-(d_l-largeur)][-d_h]=1;
                              }
                              if (d_l<0){
                                    Evolution[i][j]=0;
                                    Evolution[-d_l][-d_h]=1;
                              }
                        }
                        if (d_l>=largeur){
```

```
                              if (d_h<hauteur && d_h>=0){
                                    Evolution[i][j]=0;
                                    Evolution[largeur-1-(d_l-largeur)][d_h]=1;
                              }
                              if (d_h>=hauteur){
                                    Evolution[i][j]=0;
                                    Evolution[largeur-1-(d_l-largeur)][hauteur-
1-(d_h-hauteur)]=1;
                              }
                              if (d_h<0){
                                    Evolution[i][j]=0;
                                    Evolution[largeur-1-(d_l-largeur)][-d_h]=1;
                              }
                        }
                        if (d_l<0){
                              if (d_h<hauteur && d_h>=0){
                                    Evolution[i][j]=0;
                                    Evolution[-d_l][d_h]=1;
                              }
                              if (d_h>=hauteur){
                                    Evolution[i][j]=0;
                                    Evolution[-d_l][hauteur-1-(d_h-hauteur)]=1;
                              }
                              if (d_h<0){
                                    Evolution[i][j]=0;
                                    Evolution[-d_l][-d_h]=1;
                              }
                        }
                    }
                }
            }

            paint P = new paint();
            P.largeur = largeur;
            P.hauteur = hauteur;
            P.Evolution = Evolution;
            P.setBackground(Color.WHITE);
            fenetre.setContentPane(P);
            fenetre.setVisible(true);

            ////Attention enregistrement des différentes fenêtres
            //BufferedImage image = new BufferedImage
(P.getWidth(),P.getHeight(), BufferedImage.TYPE_INT_RGB);
            //Graphics2D graphics2D = image.createGraphics();
            //P.paint(graphics2D);
            //ImageIO.write(image,"png", new
File(String.format("C:/Users/Chloé/Desktop/PSE robot/puits noirs2/image
%d.png",C)));
            try {
                  TimeUnit.MILLISECONDS.sleep(100); //pause entre les fenetres
en ms
            }
            catch(InterruptedException e) {
                //Handle exception

            }
}
}
```