11 octobre 2016	**TP - Modèles de Mobilité Chaotiques**	M2 SMAC

L'objectif de ce TP est d'implémenter des modèles de mobilité en utilisant des systèmes dynamiques en régime chaotique. L'intérêt de ces modèles est le suivant : ils sont déterministes mais imprévisibles à long terme tout en ayant une structure dynamique ordonnée. Pour un observateur le comportement d'un mobile semblera équivalent à un comportement aléatoire. Pour la personne ayant construit le modèle la trajectoire du mobile est connue à l'avance et respecte certains motifs liés à la dynamique du système. Des systèmes dynamiques discrets et continus sont utilisés pour obtenir une solution en régime chaotique avec différentes dynamiques.

L'implémentation se fera en `java` en utilisant la librairie `JBotSim` (alpha 18) `http://jbotsim.sourceforge.net/` pour réaliser la visualisation des déplacements des agents mobiles.

Exercice 1.1 Résolution de systèmes dynamiques

1. Implémenter l'application logistique
$$x_{n+1} = r\ x_n\ (1 - x_n)$$
et tracer son diagramme de bifurcations pour $r \in [2,4; 4]$.
2. Implémenter le système de Rössler
$$\begin{cases} \dot{x} = -y - z \\ \dot{y} = x + ay \\ \dot{z} = b + z(x - c) \end{cases}$$
et le résoudre avec une méthode de Runge-Kutta 4. Tracer l'attracteur solution de ce système pour $a = 2$, $b = 2$ et $c = 5,7$. Les classes `Solver`, `Derivative` et `Rossler` sont mises à disposition.
3. Implémenter le système de Lorenz
$$\begin{cases} \dot{x} = s(y - x) \\ \dot{y} = rx - y - xz \\ \dot{z} = xy - bz \end{cases}$$
et tracer l'attracteur solution pour $s = 10$, $b = 8/3$ et $r = 28$.

Exercice 1.2 Modèles de mobilité chaotiques

Dans ce TP, nous allons implémenter des modèles de mobilité chaotiques dans le but de couvrir une zone. Le taux de couverture est le nombre de cases de l'environnement visitées depuis le début de la simulation divisé par le nombre total de cases. Nous comparerons les modèles obtenus avec des modèles de marches aléatoires. L'objectif est de tracer le graphique représentant le taux de couverture d'une zone en fonction du nombre d'itérations.

1. Implémentation de la marche aléatoire
   (a) Implémenter un modèle de marche aléatoire : à chaque instant, un robot se déplace dans une direction aléatoire.
   (b) Pour 10 robots positionnés aléatoirement sur un espace de taille 500×500, tracer le taux de couverture en fonction du nombre d'itérations.
2. Implémentation d'une marche à partir de l'application logistique
   (a) Implémenter un modèle de marche chaotique : à chaque instant, un robot se déplace dans une direction obtenue à partir du résultat d'une itération de l'application logistique. L'application donne une valeur comprise en 0 et 1, l'angle est alors $\theta_{n+1} = 2\pi x_{n+1}$.
   (b) Pour 10 robots positionnés aléatoirement sur un espace de taille 500×500, tracer le taux de couverture en fonction du nombre d'itérations pour $r = 3.75$ et $r = 4$.
   ($\star$) Soit d le nombre de décimales de x_{n+1} utilisées pour calculer l'angle θ_{n+1}. Quels sont les motifs obtenus lorsque d varie de 1 à 7 ?
3. Implémentation d'une marche à partir d'un système d'EDO.
   (a) Implémenter un modèle de marche chaotique : à chaque instant, un robot se déplace dans une direction obtenue à partir de la variable y du système de Lorenz.
   (b) Pour 10 robots positionnés aléatoirement sur un espace de taille 500×500, tracer le taux de couverture en fonction du nombre d'itérations pour $s = 10$, $b = 8/3$ et $r = 28$.
   ($\star$) Implémenter d'autres modèles à partir d'autres systèmes d'équations ayant des attracteurs chaotiques comme solution (e.g. systèmes de Chua, de Chen, ...).
4. Conclure sur l'utilisation d'un modèle de mobilité chaotique par rapport à un modèle aléatoire.

Exercice 1.3 Marche de Manhattan

Le modèle de marche de Manhattan autorise quatre déplacement : en avant, en arrière, à droite et à gauche. Les robots sont positionnés initialement sur une grille qu'ils ne peuvent pas quitter.

1. Implémenter une marche aléatoire où les choix sont équiprobables dans un environnement carré.
2. En utilisant le système de Lorenz, proposer une méthode pour réaliser une marche chaotique de Manhattan. La difficulté réside dans le fait qu'il faut passer du modèle continu à quelque chose de discret à l'aide d'une *section de Poincaré*.
3. Comparer les deux systèmes des questions précédentes à l'aide du taux de couverture en fonction du nombre d'itérations.

(⋆) On considère que la marche arrière est impossible. Dans le modèle de marche aléatoire, la probabilité de se déplacer vers l'avant est alors de 50%, les autres probabilités restent à 25%. Adapter le modèle de marche de Manhattan chaotique et comparer le avec la marche aléatoire à l'aide du taux de couverture en fonction du nombre d'itérations.

Extraits de code pour tracer les trajectoires

```java
package robot;
import jbotsim.Topology;
import jbotsim.ui.JViewer;

public class MainClass{
    public static void main(String[] args) {
        Topology tp = new Topology(500, 500);
        tp.setDefaultNodeModel(Robot.class);
        int[][] environnement = new int[500][500];
        tp.setProperty("env",environnement);
        JViewer jv = new JViewer(tp);
        jv.getJTopology().addBackgroundPainter(new BackgroundPainter());
    }
}
```

Fichier 1 – MainClass.java

```java
package robot;
import jbotsim.Topology;
import java.awt.*;

public class BackgroundPainter implements jbotsim.ui.painting.BackgroundPainter
{
    @Override
    public void paintBackground(Graphics2D g, Topology tp) {
        int[][] env = (int[][]) tp.getProperty("env");
        // Paints a background image
        for (int i=0; i<500 ;i++ ) {
            for (int j=0; j<500 ;j++ ) {
                if (env[i][j]==1)
                    g.drawOval(i,j,1,1);
            }
        }
    }
}
```

Fichier 2 – BackgroundPainter.java

```java
@Override
public void onPreClock(){
    int [][] env = (int[][]) getTopology().getProperty("env");
    int x = (int) this.getLocation().getX();
    int y = (int) this.getLocation().getY();
    if (!(x>=500 || y>=500 || x<0 || y<0))
        env[x][y] = 1;
}
```

Fichier 3 – Robot.java