AI40 - Devoir Maison 1

FISA

Alexis LEBEL

P24

Table des Matières

1 Modèle Cinématique du robot

1.1 Démonstration de la généralisation

En partant du modèle cinématique définit dans le cours (Partie II cours 1 page 46) :

$$\begin{cases} \dot{x} = V\cos(\theta) \\ \dot{y} = V\sin(\theta) \\ \dot{\theta} = \omega \end{cases}$$

Figure 1: Modèle cinématique du robot

Ce modèle est défini dans le repère global, or le papier utilise un repère de Frenet.

Matriciellement, la cinématique du robot unicycle est donnée par :

$$\dot{\xi} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos\theta & -l_2\cos\theta - l_1\sin\theta \\ \sin\theta & -l_2\sin\theta + l_1\cos\theta \\ 0 & 1 \end{pmatrix} \begin{pmatrix} v \\ w \end{pmatrix} \quad (1)$$

Figure 2: Modèle cinématique matriciel

Ce qui donne, en développant les matrices :

$$\{\dot{x} = v\cos(\theta) - \omega(l_1\sin(\theta) + l_2\cos(\theta))\dot{y} = v\sin(\theta) + \omega(l_1\cos(\theta) - l_2\sin(\theta))\dot{\theta} = \omega$$

or, la deuxième partie des équations de x et y sont liées au décalage apporté par le repère de Frenet (qui dépends de l1, l'abscisse du point considéré, et l2, l'ordonnée du point considéré).

De ce fait, si on supperpose le point défini par (l1, l2) au point O', centre de l'entraxe de roue, on obtient :

$$\begin{cases} \dot{x} = v\cos(\theta) \\ \dot{y} = v\sin(\theta) \\ \dot{\theta} = \omega \end{cases}$$

Soit le modèle cinématique vu en cours.

1.2 Modèle Simulink

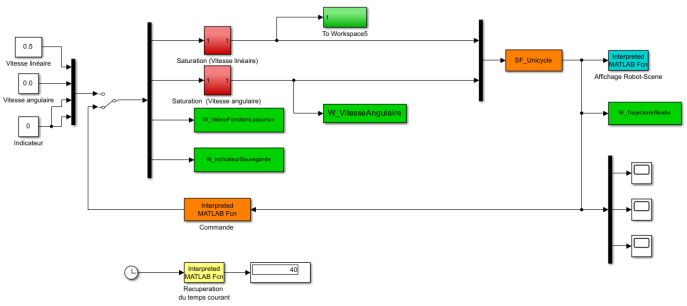

Figure 3: Modèle simulink de la simulation

Voici un algorithme en pseudo-code expliquant les différentes étapes de la commande du robot :

```
- Définition des commandes initiales (ultimement : perception)
- Définition du point cible
- Définition de la position des obstacles
- Définition du modèle du robot
- Instanciation des obstacles
- Simulation
- Affichage
```

Voici les étapes de la simulation :

```
- Saturation des vitesses linéaires et angulaires
- Application au modèle cinématique
- Calcul de la commande à appliquer
- Feedback de la commande
```

Détail de la commande :

```
- Calcul des obstacles potentiels
- Récupération de l'obstacle le plus proche
- S'il n'y a aucun obstacle potentiel, on se dirige vers le point cible (Commande d'attraction)
- Sinon, on calcule la trajectoire du robot pour éviter l'obstacle (Commande d'évitement)
```

2 Synthèse de commande en utilisant le théorème de Lyapunov

2.1 Preuve de la stabilité de la loi de commande

En partant du modèle cinématique suivant :

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} \cos\theta & -l_1 \sin\theta \\ \sin\theta & l_1 \cos\theta \end{pmatrix} \begin{pmatrix} v \\ w \end{pmatrix} = M \begin{pmatrix} v \\ w \end{pmatrix}$$

Figure 4: Modèle cinématique, page 81 article Robotica

Ce modèle suppose que le point considéré du repère de Frenet se situe sur l'axe longitudinal du robot (l2 = 0).

La loi de commande est donc (algèbre matricielle linéaire) :

$$\begin{pmatrix} v \\ w \end{pmatrix} = -KM^{-1} \begin{pmatrix} e_x \\ e_y \end{pmatrix}$$

Figure 5: Loi de commande

Avec :

- K : un gain de commande
- M^{-1} : l'inverse de la matrice du modèle cinématique
- e_x : l'erreur sur l'axe x, $x - x_T$
- e_y : l'erreur sur l'axe y, $y - y_T$

En considérant la fonction de Lyapunov suivante :

$$V_1 = \frac{1}{2}d^2$$

$$V_1 = \frac{1}{2}((x - x_T)^2 + (y - y_T)^2)$$

Démontrons que cette fonction est positive :

- Les termes $(x - x_T)^2$ et $(y - y_T)^2$ sont toujours positifs (carré d'un nombre réel)
- La racine de la somme de ces termes est donc positive
- La multiplication par 1/2 ne change pas le signe de la fonction

De ce fait, la fonction de Lyapunov est positive.

En dérivant cette fonction, on obtient :

$$V_1' = \frac{1}{2}\frac{d}{dt}e_x^2 + e_y^2$$

$$V_1' = \frac{1}{2}(2e_x\dot{e_x} + 2e_y\dot{e_y})$$

Or, on connait les valeurs de $\dot{e_x}$ et $\dot{e_y}$:

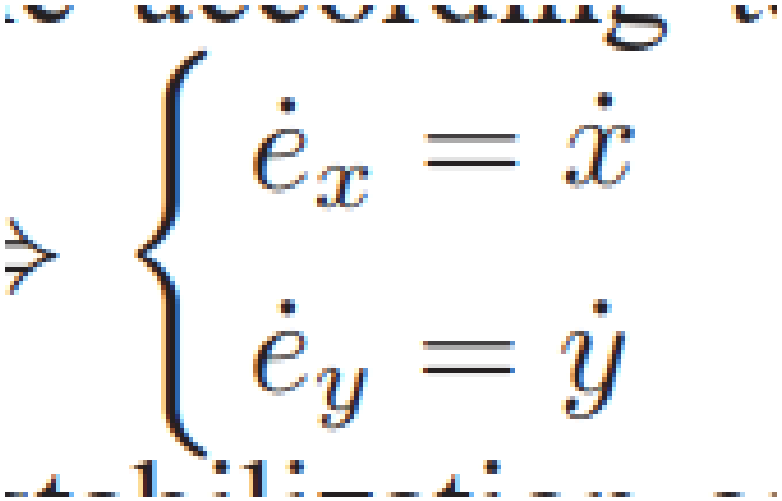

Figure 6: Dérivées des erreurs

En remplaçant ces valeurs dans l'équation de V_1', on obtient :

$$V_1' = e_x\dot{x} + e_y\dot{y}$$

Or, on connait aussi les valeurs de $\dot{x}$ et $\dot{y}$:

$$\begin{cases} \dot{x} = v\cos(\theta) \\ \dot{y} = v\sin(\theta) \\ \dot{\theta} = \omega \end{cases}$$

En sachant que v et $\dot{\theta} = \omega$ sont définies par le régulateur linéaire $-K.M^{-1}.\vec{e}$:

$$M^{-1} = \frac{1}{\cos\theta.l1.\cos\theta - l1.\sin\theta.\sin\theta}\begin{pmatrix} l1.\cos\theta & l1.\sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$$

$$M^{-1} = \frac{1}{l1.\cos^2\theta - l1.\sin^2\theta}\begin{pmatrix} l1.\cos\theta & l1.\sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$$

$$M^{-1} = \begin{pmatrix} \frac{\cos\theta}{\cos^2\theta - \sin^2\theta} & \frac{\sin\theta}{\cos^2\theta - \sin^2\theta} \\ \frac{-\sin\theta}{l1.\cos^2\theta - l1.\sin^2\theta} & \frac{\cos\theta}{l1.\cos^2\theta - l1.\sin^2\theta} \end{pmatrix}$$

On trouve donc :

$$\begin{cases} v = -K.(\frac{\cos\theta}{\cos^2\theta - \sin^2\theta}.e_x + \frac{\sin\theta}{\cos^2\theta - \sin^2\theta}.e_y) \\ \omega = -K.(\frac{-\sin\theta}{l1.\cos^2\theta - l1.\sin^2\theta}.e_x + \frac{\cos\theta}{l1.\cos^2\theta - l1.\sin^2\theta}.e_y) \end{cases}$$

$$\begin{cases} v = -K.(\frac{\cos\theta}{\cos 2\theta}.e_x + \frac{\sin\theta}{\cos 2\theta}.e_y) \\ \omega = -K.(\frac{-\sin\theta}{l1.\cos 2\theta}.e_x + \frac{\cos\theta}{l1.\cos 2\theta}.e_y) \end{cases}$$

En remplaçant ces valeurs dans l'équation de V_1', on obtient :

$$V_1' = e_x.v.\cos\theta + e_y.v.\sin\theta$$

$$V_1' = v.(e_x.\cos\theta + e_y.\sin\theta)$$

$$V_1' = -K.(\frac{\cos\theta}{\cos 2\theta}.e_x + \frac{\sin\theta}{\cos 2\theta}.e_y).(e_x.\cos\theta + e_y.\sin\theta)$$

$$V_1' = -K.\frac{\cos\theta}{\cos 2\theta}.e_x(e_x.\cos\theta + e_y.\sin\theta) - K.\frac{\sin\theta}{\cos 2\theta}.e_y(e_x.\cos\theta + e_y.\sin\theta)$$

$$V_1' = -K.(\frac{\cos^2\theta.e_x + \cos\theta\sin\theta.e_y}{\cos 2\theta}e_x + \frac{\sin\theta\cos\theta.e_x + \sin^2\theta.e_y}{\cos 2\theta}e_y)$$

$$= -K\frac{e_x^2.\cos^2\theta + e_x.e_y.\cos\theta\sin\theta + e_x.e_y.\cos\theta\sin\theta + e_y^2.\sin^2\theta}{\cos 2\theta}$$

$$= -K\frac{e_x^2.\cos^2\theta + 2.e_x.e_y.\cos\theta\sin\theta + e_y^2.\sin^2\theta}{\cos 2\theta}$$

$$= -K\frac{e_x^2.\cos^2\theta + e_x.e_y.\sin 2\theta + e_y^2.\sin^2\theta}{\cos 2\theta}$$

Preuve de la négativité de V_1' :

- Les termes $e_x^2.\cos^2\theta$ et $e_y^2.\sin^2\theta$ sont toujours positifs
- Le signe dépends donc de $\frac{e_x.e_y.\sin 2\theta}{\cos 2\theta}$ = $e_x.e_y.\tan 2\theta$

2.2 Implémentation Matlab

Changements dans le code du fichier "CommandeAttraction.m" :

```
K = [K1 0; 0 K2];
M = [cos(ThetaReel) -l1*sin(ThetaReel); sin(ThetaReel) l1*cos(ThetaReel)];
Commande = K * inv(M) * [Ex; Ey];
```

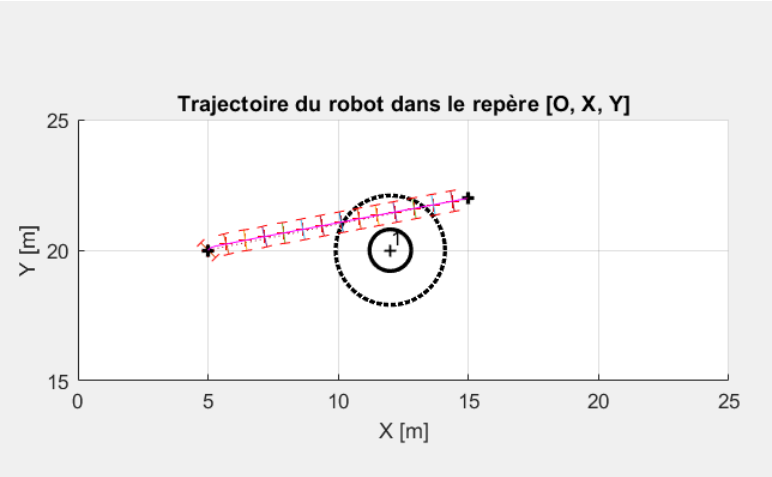

Figure 7: Trajectoire du robot

Le robot adapte bien sa trajectoire pour aller sur le point cible.

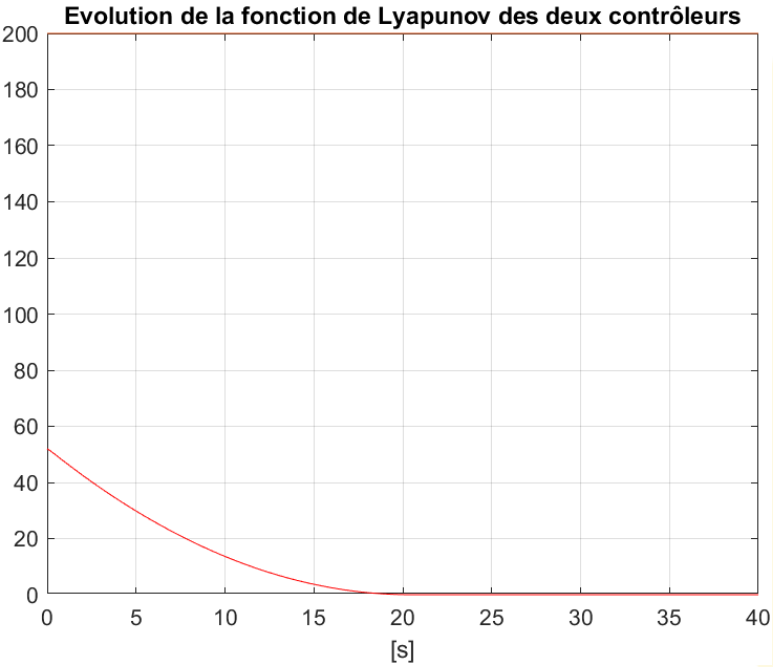

Figure 8: Evolution de la Lyapunov

3 Evitement d'obstacles avec la méthode des cycles limites

3.1 Changements dans Matlab

Matrice de changement de repère :

```
T_O_A = [cos(Alpha) -sin(Alpha) 0 X_D_O
         sin(Alpha) cos(Alpha)  0 Y_D_O
         0          0           1 0
         0          0           0 1    ];
```

Trajectoire du robot :

```
if Y_Prime < 0 %%clock-wise
    X_dot = Yrelatif + Xrelatif*((RayonCycleLimite^2) - (Xrelatif^2) - (Yrelatif^2));
    Y_dot = -Xrelatif + Yrelatif*((RayonCycleLimite^2) - (Xrelatif^2) - (Yrelatif^2));
else %%counter-clockwise
    X_dot = -Yrelatif + Xrelatif*((RayonCycleLimite^2) - (Xrelatif^2) - (Yrelatif^2));
    Y_dot = Xrelatif + Yrelatif*((RayonCycleLimite^2) - (Xrelatif^2) - (Yrelatif^2));
end
```

(La condition permet de gérer les sens de rotation)

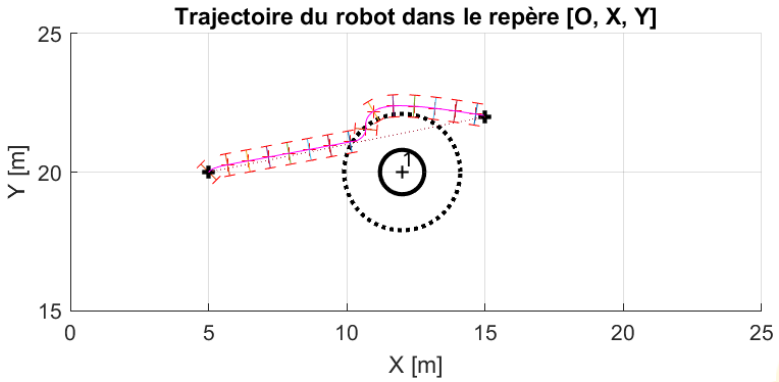

Figure 9: Trajectoire du Robot

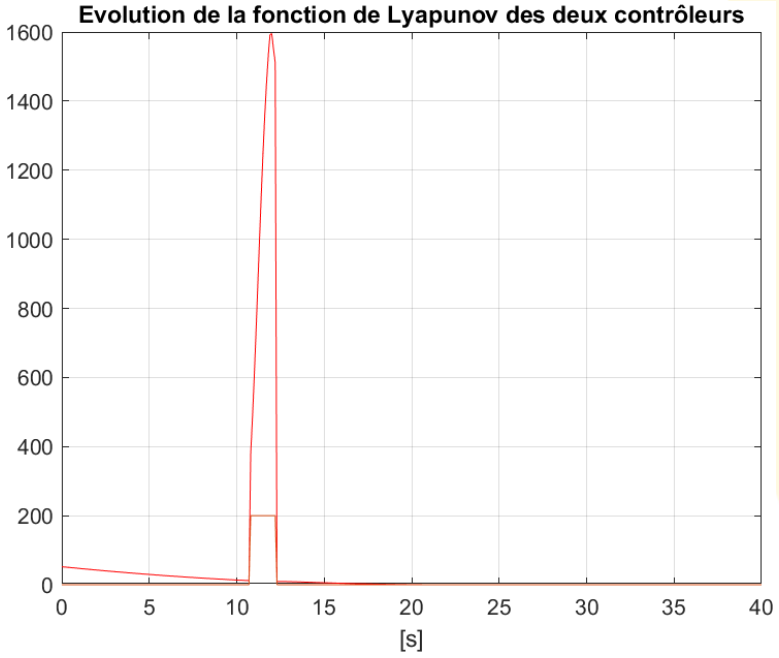

Figure 10: Fonction de Lyapunov

3.2 Preuve de la stabilité de la loi de commande

On utilise la fonction de Lyapunov candidate suivante :

$$V_2 = \frac{1}{2}\theta_e^2 \tag{11}$$

Cette fonction de Lyapunov est choisie parce qu'elle est positive et quadratique, ce qui simplifie l'analyse de la stabilité.

On calcule la dérivée par rapport au temps de V_2 :

$$\dot{V}_2 = \theta_e \cdot \dot{\theta}_e$$

Ensuite, on remplace $\dot{\theta}_e$ par l'expression issue de la dynamique du système, qui est donnée par :

$$\dot{\theta}_e = -K_p\theta_e \tag{10}$$

En substituant cette expression dans l'équation de $\dot{V}_2$, on obtient :

$$\dot{V}_2 = \theta_e \cdot (-K_p\theta_e)$$

Ce qui se simplifie en :

$$\dot{V}_2 = -K_p\theta_e^2$$

Ici, K_p est une constante positive. Par conséquent, $\dot{V}_2$ est toujours négatif ou nul car θ_e^2 est toujours positif ou nul. Cela signifie que V_2 est une fonction décroissante du temps.

Parce que V_2 est bornée inférieurement par zéro et décroissante, selon le critère de stabilité de Lyapunov, cela implique que V_2 converge vers zéro à mesure que le temps tend vers l'infini. En d'autres termes, l'erreur θ_e converge vers 0, ce qui signifie que le système est stable et l'erreur est éliminée à long terme.

Ainsi, l'utilisation de cette fonction de Lyapunov démontre que l'erreur θ_e décroît et converge vers zéro, assurant la stabilité du système.

3.3 Cas multi-obstacles

Pour faire fonctionner le cas multi-obstacles, il faut rajouter les lignes suivantes dans le code Matlab :

```
if(isempty(RayonCycleLimite))
    RayonCycleLimite=GetRv(Obstacle(IndiceObstaclePlusProche))-0.3;
end
```

Cela permet de corriger un bug qui empêche le correcteur de récupérer le rayon de cycle limite de l'obstacle le plus proche.

Une fois cette correction effectuée, on peut activer l'exemple 2 dans le programme principal, ce qui donne les figures suivantes :

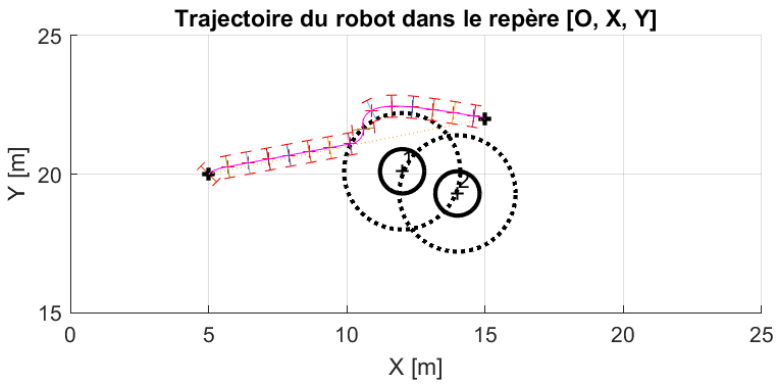

Figure 11: Déplacement du robot en multi-obstacles

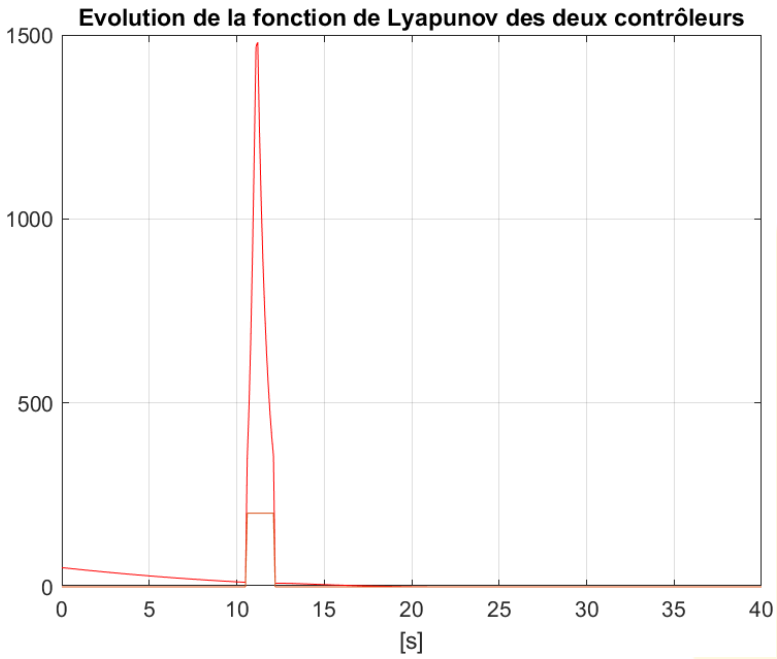

Figure 12: Lyapunov multi-obstacles

Cependant, si l'on déplace le point de destination, le robot ne réagit pas correctement :

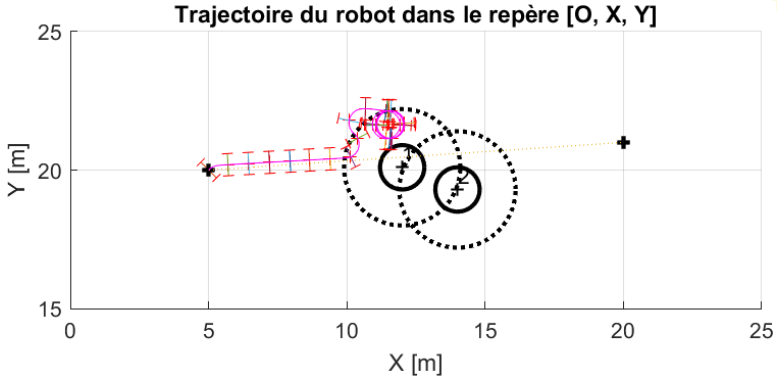

Figure 13: Robot en multi-obstacles coincé

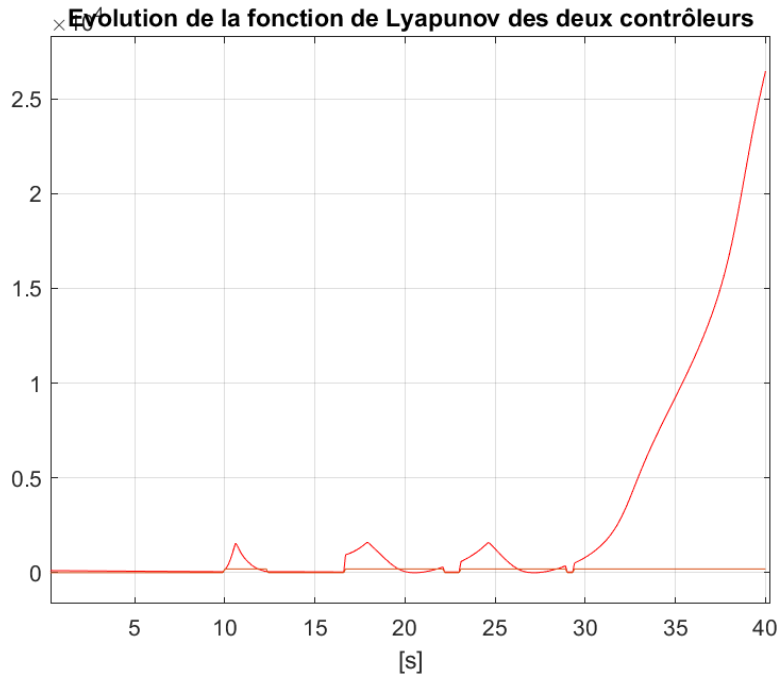

Figure 14: Lyapunov du robot coincé

Le robot est coincé dans le disque de cycle limite du premier obstacle, et en dehors du disque du deuxième obstacle. Il ne peut donc pas se déplacer.

C'est un cas non géré par la simulation bien que prévu dans le papier de 2009 (*V. CONFLICTING SITUATIONS MANAGEMENT*, Cas (a))

4 Conclusion

En conclusion, la méthode des cycles limites est une méthode efficace pour éviter les obstacles. Le robot est capable de se déplacer en évitant les obstacles, même en cas de multi-obstacles. Cependant, il reste des mécanismes à ajouter pour gére les situations de conflit, comme le montre le cas du robot coincé entre deux obstacles.

Les preuves de stabilité de la loi de commande montrent que le système est stable et que l'erreur converge vers zéro à long terme. Cela assure le bon fonctionnement du robot et sa capacité à atteindre sa destination.