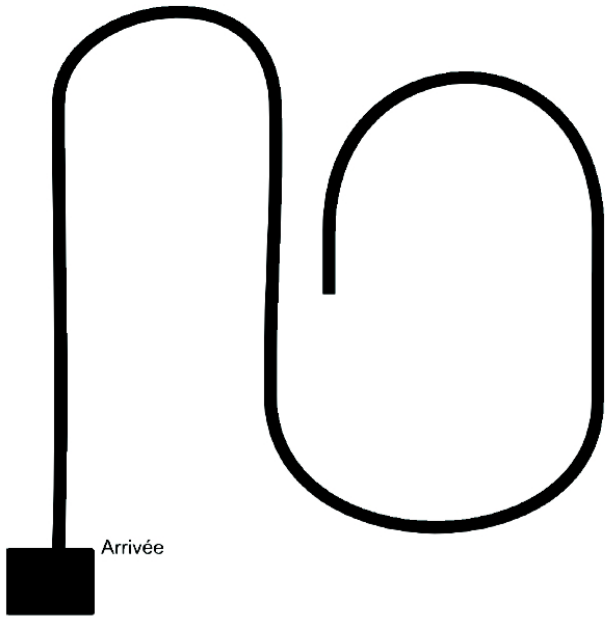
Modélisation et commande d'un robot mobile

- Problème
- Étape de conception
  - Modélisation
  - Étude du cahier de charges
  - Stratégie de commande
  - Test par simulation
  - Implantation numérique, génération de code (cours suivant)

Énoncé et cahier de charges

- Faire suivre au robot une ligne noire sur le sol
- Deux capteurs de lumière, positionnés sur le devant et qui pointent vers le sol afin de détecter la ligne noire.



Contrainte : le robot peut seulement, à partir des informations venant des capteurs, estimer sa déviation par rapport à la ligne.

Étapes à suivre

- Trouver le modèle du système (états, entrées et sorties)
- Proposer une loi de commande
  - Commande de direction (orientation) du robot
  - Commande de vitesse du robot
- Tester par simulation
- Implantation, validation, déboggage

Modèle du robot

$$\dot{x} = \frac{v_g + v_d}{2} cos(\theta) \quad (1)$$

$$\dot{y} = \frac{v_g + v_d}{2} sin(\theta) \quad (2)$$

$$\dot{\theta} = \frac{v_d - v_g}{l} \quad (3)$$

- l : distance entre deux roues, v_g et v_d sont les vitesses des deux roues gauche et droite. $\theta = -wt$.
- **Capteurs** : 2 capteurs de lumière.
- **Actionneurs** : 2 servomoteurs.
- **Consigne** : La ligne noire.

Modèle des capteurs

- **Hypothèse** : Supposons que les capteurs soient calibrés sur une échelle de 0 (noir) à 100 (blanc). **Il faut vériér cette hypothèse et ajuster le calibrage si besoin !**
- La zone est complètement noire, le capteur retourne 0.
- La zone est complètement blanche, le capteur retourne 100.
- Si un capteur retourne une valeur $c < 100$, le robot n'est plus aligné avec la ligne
  Idée pour la loi de commande : Faire tourner le robot d'un angle proportionnel à la différence $(C_d - C_g)$.

Commande de trajectoire

- Commande par retour d'états
- Découpler la dynamique du système
  - Système d'orientation (θ).

$$\dot{\theta} = \frac{u_\theta}{l}$$

  - Système de translation (x, y)

$$\dot{x} = u_\delta cos(\theta)$$

$$\dot{y} = u_\delta sin(\theta)$$

avec

$$u_\delta = \frac{v_d + v_g}{2}$$

$$u_\theta = v_d - v_g$$

Exercise : Commande de l'orientation du robot

1. Trouver la fonction de transfert entre θ et u_θ (en Laplace).
2. Boucler le système avec un correcteur (type PI) et calculer la fonction de transfert en boucle fermée.
3. Réglage du correcteur : trouver les paramètres de correcteur pour avoir un comportement désiré (stable, rapide, sans dépassement, avec une erreur statique nulle..).
4. Tester en simulation (en utilisant Matlab).

Exercise : Commande de l'orientation du robot

Question 1 :

De l'équation (3) nous avons

$$\dot{\theta} = \frac{u_\theta}{l},$$

ce qui donne en transformée de Laplace

$$s\theta(s) = \frac{u_\theta(s)}{l}$$

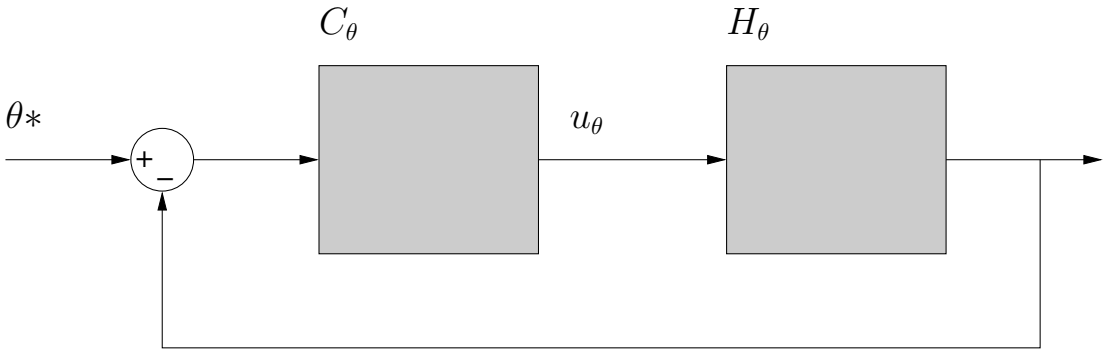
(en supposant que les conditions initiales sont 0). La fonction de transfert de θ par rapport à l'entrée u_θ est alors :

$$H_\theta(s) = \frac{\theta(s)}{u_\theta(s)} = \frac{1}{ls}$$

Exercise : Commande de l'orientation du robot

Question 2 : Le schéma de commande

La sortie u_θ du correcteur $C_\theta(s)$ est connectée à l' entrée du bloc $H_\theta(s)$ (correspondant à la dynamique de l'orientation θ du robot). L'entrée du correcteur $C_\theta(s)$ est l'écart entre la sortie θ de $H_\theta(s)$ et l'orientation désirée θ^*.



Pour un correcteur de type PI nous avons : $C_\theta(s) = k_{p\theta} + \frac{k_{p\theta}}{s}$

La fonction de transfert en boucle fermée

$$H(s) = \frac{C_\theta(s)H_\theta(s)}{1 + C_\theta(s)H_\theta(s)} = \frac{\frac{k_{p\theta}}{l}s + \frac{k_{i\theta}}{l}}{s^2 + \frac{k_{p\theta}}{l}s + \frac{k_{i\theta}}{l}}$$

Exercise : Commande de l'orientation du robot

Question 3 : Réglage du correcteur de l'orientation du robot

Notons $\alpha = 1/l$.

Si l'on choisit $k_{i\theta} = \omega^2/\alpha$ et $k_{p\theta} = 2\xi\omega/\alpha$ (avec $\omega > 0$, $\xi > 0$), le système en boucle fermée est du deuxième ordre avec un gain statique de 1, une pulsation ω et un amortissement ξ.

Notons que les pôles du système en boucle fermée sont

$$p_{1,2} = -\omega\xi \pm i\sqrt{\omega^2(1-\xi^2)}$$

avec les parties réelles $Re(p_{1,2}) < 0$, ce qui garantit la stabilité de θ.

Il nous reste la liberté de choisir des valeurs de ω et de ξ afin d'obtenir des performances désirées (temps de réponse, dépassement, etc).

Exercise : Commande de l'orientation du robot

Question 3 : Réglage du correcteur de l'orientation du robot (suite)

Rappel : la réponse d'un système du deuxième ordre ayant le polynôme caractéristique $\pi(s) = s^2 + 2\xi\omega s + \omega^2$ avec les racines $p_{1,2} = -\omega\xi \pm i\sqrt{\omega^2(1-\xi^2)}$

– Temps de montée (rise time) $10 - 90\%$:

$$t_r = \frac{1 + 1.1\xi + 1.4\xi^2}{\omega}$$

– Temps de stabilisation (settling time)

$$t_s = \frac{3}{\xi\omega}$$

– Temps de pic (time to peak amplitude)

$$t_p = \frac{\pi}{\omega\sqrt{1-\xi^2}}$$

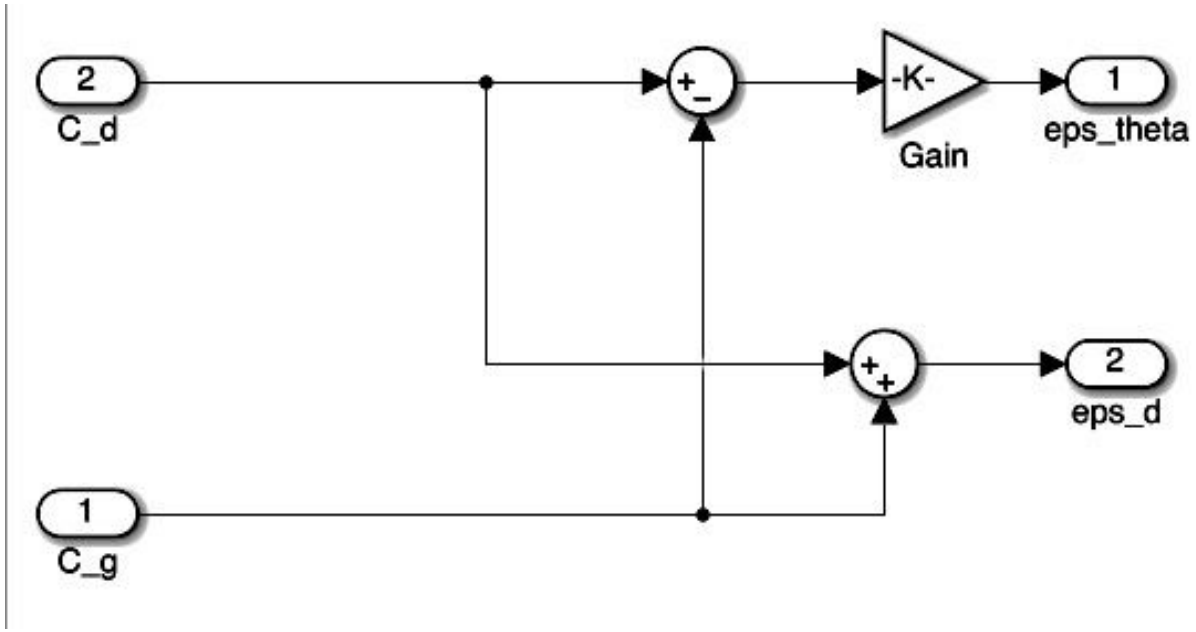
– Dépassement (peak overshoot) :

$$M_p = e^{-\xi\omega t_p}$$

Commande de la vitesse du robot

- En ligne droite : Vitesse du robot constante. Donc un écart de vitesse constant à l'entrée du correcteur $\Rightarrow$ Idée : utiliser un correcteur de type P et de garder l'entrée proportionelle à la somme $(C_d + C_g)$.
- Au virage : Vitesse réduite pour éviter des dépassements à la courbure. Réduire l'erreur à l'entrée du correcteur de vitesse.
- Détecter un virage : c'est quand la commande u_θ a une grande valeur absolue. Idée : soutraire de la sortie u_δ (du correcteur de vitesse) une quantité proportionnelle à la valeur absolue $|u_\theta|$

Modèle Simulink pour le calcul d'écarts



Détermination des vitesses de deux roues

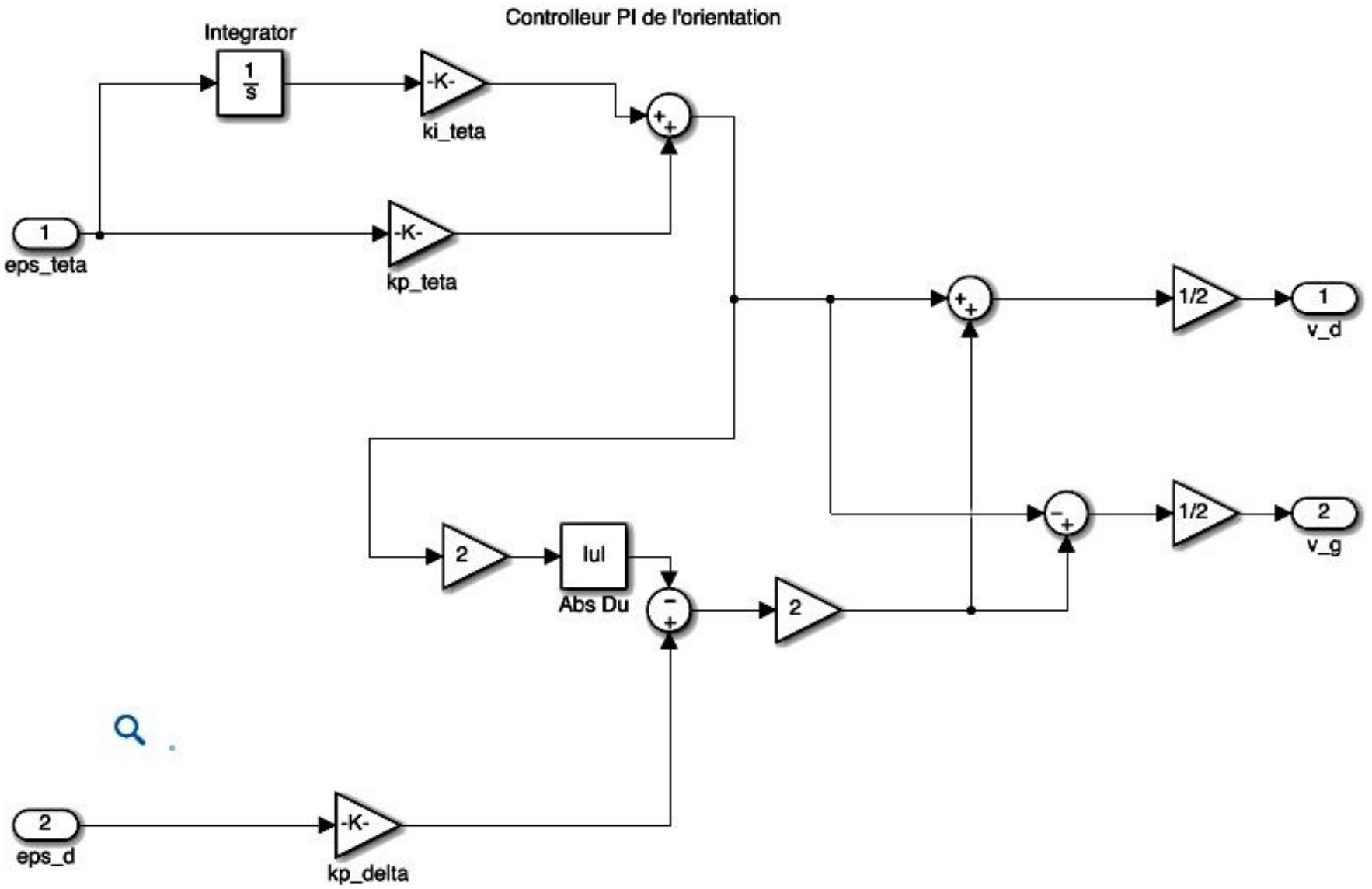
À partir de

$$u_\delta = \frac{v_d + v_g}{2}$$
$$u_\theta = v_d - v_g$$

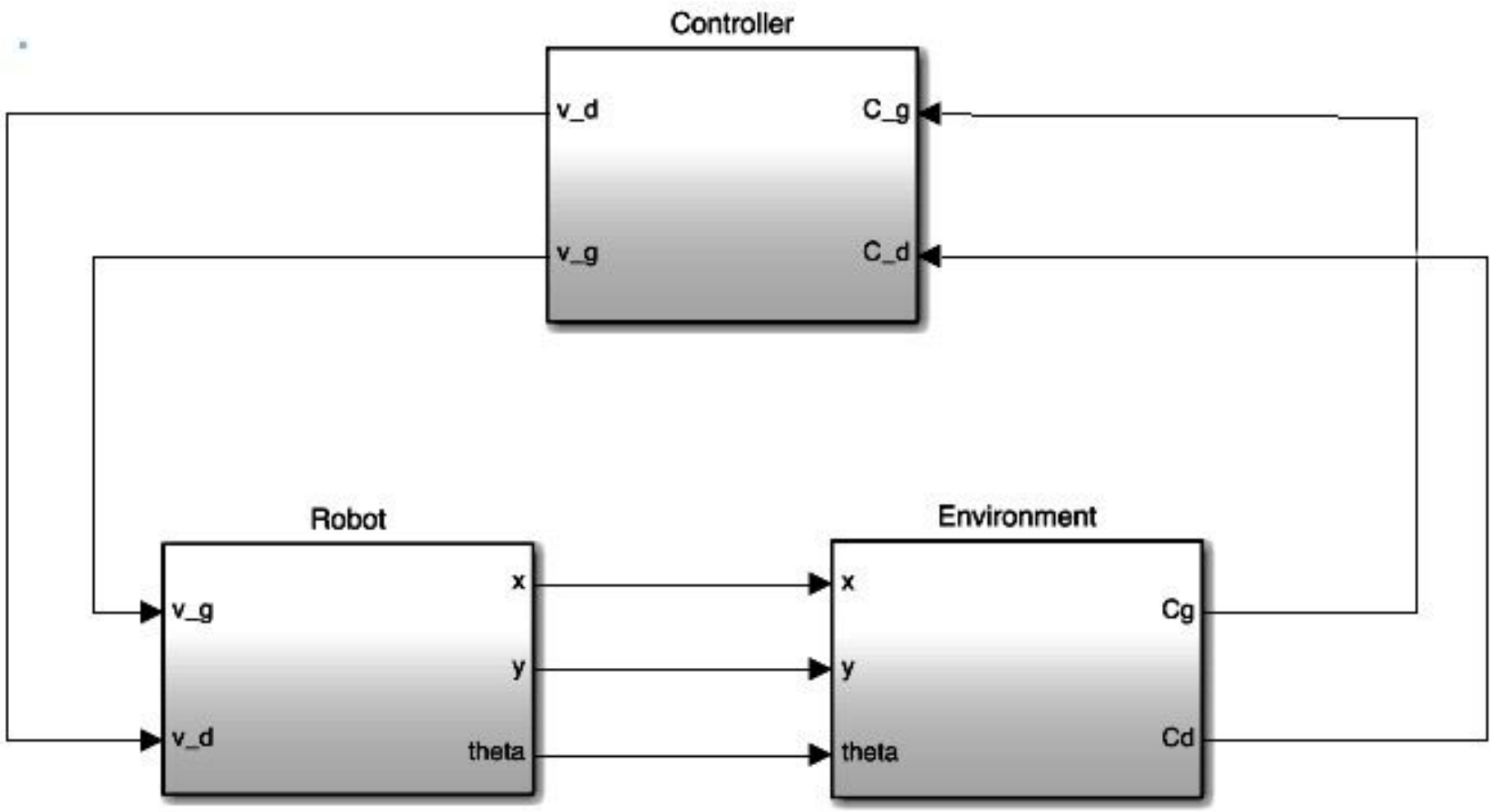
on obtient les vitesses des deux roues en fonction des valeurs de u_θ et u_δ

$$v_d = \frac{u_\delta + u_\theta}{2}$$
$$v_g = \frac{u_\delta - u_\theta}{2}$$

Modèle Simulink des correcteurs

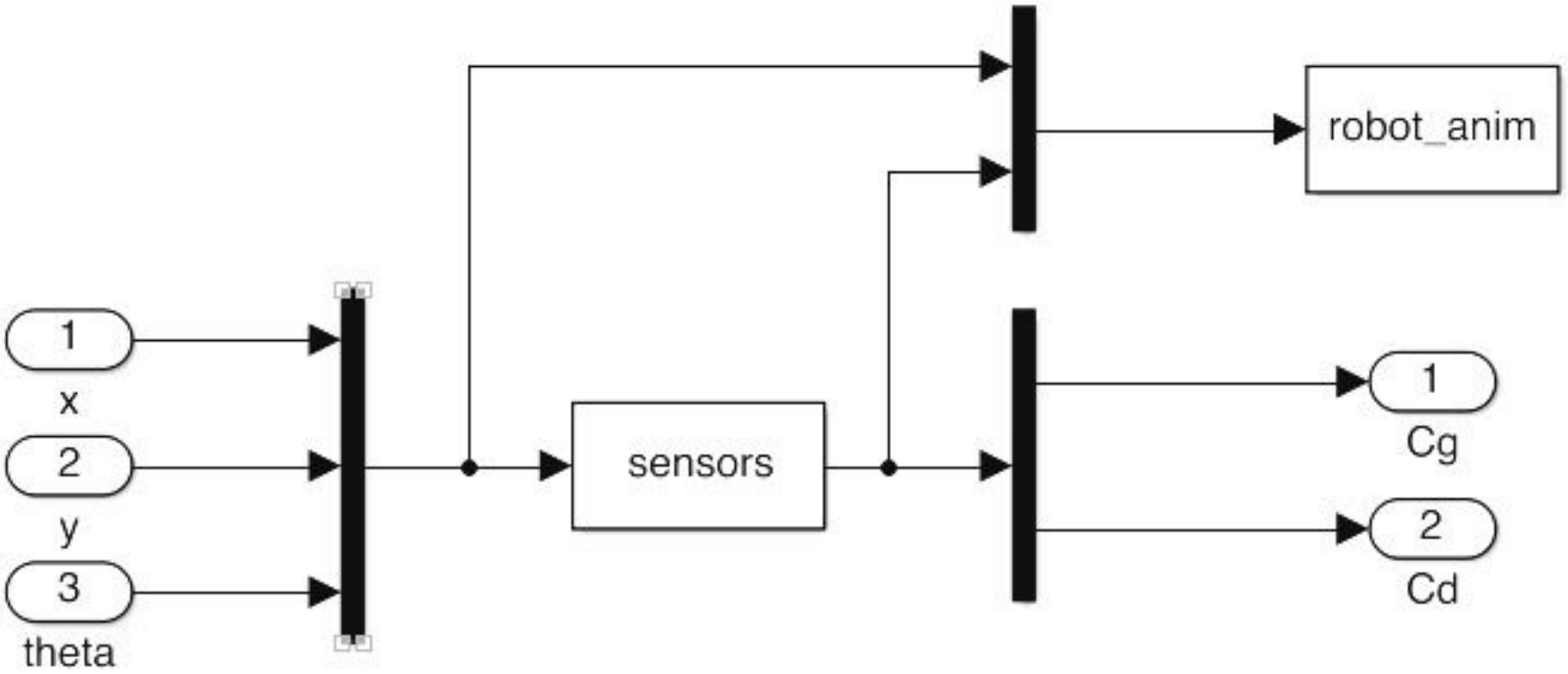


Modèle du robot et l'environnement



Le bloc "Environment" a pour l'objectif d'émuler les comportements des capteurs de lumière d'un robot sur une ligne donnée.

Modèle de l'environnement



Exercise : Modélisation et simulation avec Simulink

- 1. Télécharger le modèle SIMULINK (robot + contrôleur + environnement) (`ExRobotAndEnvironmentControllerCont.mdl`). L'archive contient également :
  - `InitRobotAndEnvironment.m` : Ce fichier Matlab sert à initialiser les paramètres du modèle (tels que la distance entre deux roue, les paramètres du contrôleur, la position et l'orientation initiales du robot, le nom de l'image de circuit à tester)
  - quelques fichiers d'image de circuits
  - `robot_anim.m` et `sensors.m` : fonctions Matlab pour l'émulation de l'environnment

Exercise : Modélisation et simulation avec Simulink

- 2. Lancer Matlab/Simulink et ouvrir le modèle.
- 3. Exécuter le script "`InitRobotandEnvironment.m`" (taper simplement "`InitRobotandEnvironment`" dans la "Command Window" de Matlab)
- 4.Utiliser le menu "Simulation" pour simuler le modèle
- 5. Régler les correcteurs pour améliorer la performance