6 TP, partie 3 : Implémentation du retour d'état pour le Segway Lego EV3

Dans cette section, on se propose d'abord d'implémenter le retour d'état sous Simulink autour d'un simulateur du Lego Ev3.
Une fois le schéma validé, on utilisera la capacité de Simulink à transformer un schéma blocs en code C pour programmer le Lego.

6.1 Analyse du simulateur Ev3

Le Lego EV3 (modèle linéaire) est simulé dans le sous-système du fichier `simulation.slx` avec comme sur la maquette :

— en entrée, une tension en pour cent de la batterie U_{pwm}
(Pulse Width Modulation (pwm) : signal de rapport cyclique variable entre -100 et 100)
— en sortie, la mesure du gyroscope (deg/s) et les angles moteurs en (deg) tels que délivrés par les capteurs du lego et comme préciser dans la section 1.2.

1. Ouvrir le sous-système présent dans la simulation. Au centre, on a le modèle d'état continu de la dynamique du système. Vérifier dans ce bloc que les paramètres correspondent bien aux variables que vous avez définies. Noter la condition initiale non nulle, initialisée à $[0\ \psi_0\ 0\ 0]^T$.
On trouve en entrée un gain $\frac{1}{K_{pwm}}$, avec K_{pwm} le gain de conversion d'une tension (volt) en un rapport cyclique du signal PWM ($-100\% < \ < 100\%$). En sortie, un bloc fonction (que l'on peut ouvrir) modélise les mesures fournies par les capteurs Lego avec l'unité d'angle propre aux capteurs Lego : le degré. Ces mesures sont alors échantillonnées à T_s par le bloc suivant.
Quelle propriété des capteurs modélisent les blocs `round` présents (voir section 1.2) ?

6.2 Implémentation du retour d'état

Le simulateur, comme la maquette du Segway Ev3, ne donne pas accès à l'ensemble des états du vecteur d'état X, dont on a besoin pour la commande via le retour d'état.

Il faut donc commencer par reconstruire le vecteur d'état $\left[\theta\ \psi\ \dot{\theta}\ \dot{\psi}\right]^T$ dans les bonnes unités, à partir des données fournies par les 2 seuls capteurs disponibles (figure 5).

1. En dehors du bloc `EV3(Simulateur)`, commencer par reconstruire le signal $\dot{\psi}$ dans les unités attendues en connectant un bloc `Gain` à la sortie `gyro` du simulateur.
2. Pour reconstruire l'angle ψ, on n'utilisera pas directement la mesure de, $\dot{\psi}$ car celle-ci peut contenir un faible biais constant (offset) dû à la technologie du capteur. L'intégration directe du signal avec un biais constant ne peut que diverger. On se propose donc d'estimer ce biais par un filtrage et de le soustraire à la mesure de $\dot{\psi}$ issue du gyroscope pour estimer une valeur non biaisée (voir figure 5).
Le filtre retenu pour estimer le biais est un filtre moyenneur à fenêtre glissante et facteur d'oubli exponentiel ("exponential smoothing" en anglais) qui se traduit par l'équation aux différences suivante (équation utilisée quand il faut implémenter le filtre sur un microprocesseur) :

$$\text{biais}(k) = \gamma\,\text{biais}(k-1) + (1-\gamma)u(k) \tag{18}$$

avec biais(k) le biais à estimer, $u(k)$ le signal en entrée et un facteur d'oubli $0 < \gamma < 1$.

Donner l'expression de la fonction de transfert discrète en z ou z^{-1} correspondant à ce filtre.
Rappel : $\mathbf{Z}\{y(k-1)\} = z^{-1}Y(z)$ avec $Y(z) = \mathbf{Z}(y(k))$.
3. Le facteur d'oubli choisi est $\gamma = 0.999$. Afin de valider ce choix et votre résultat précédent, créer la fonction de transfert échantillonnée à T_s ms pour ce filtre (`help tf`). Tracer son diagramme de Bode (fonction `bode`). Est-ce un filtre passe-bas ou passe-haut ? Quels sont sa fréquence de coupure et le gain en basse fréquence ? Est-ce un filtre adéquat pour identifier un biais constant ?
4. Avec les blocs Simulink de base (gain, sommateur, mux, demux,..) et blocs de fonctions de transfert discrètes [3], implémentez alors l'estimation de l'état ψ.

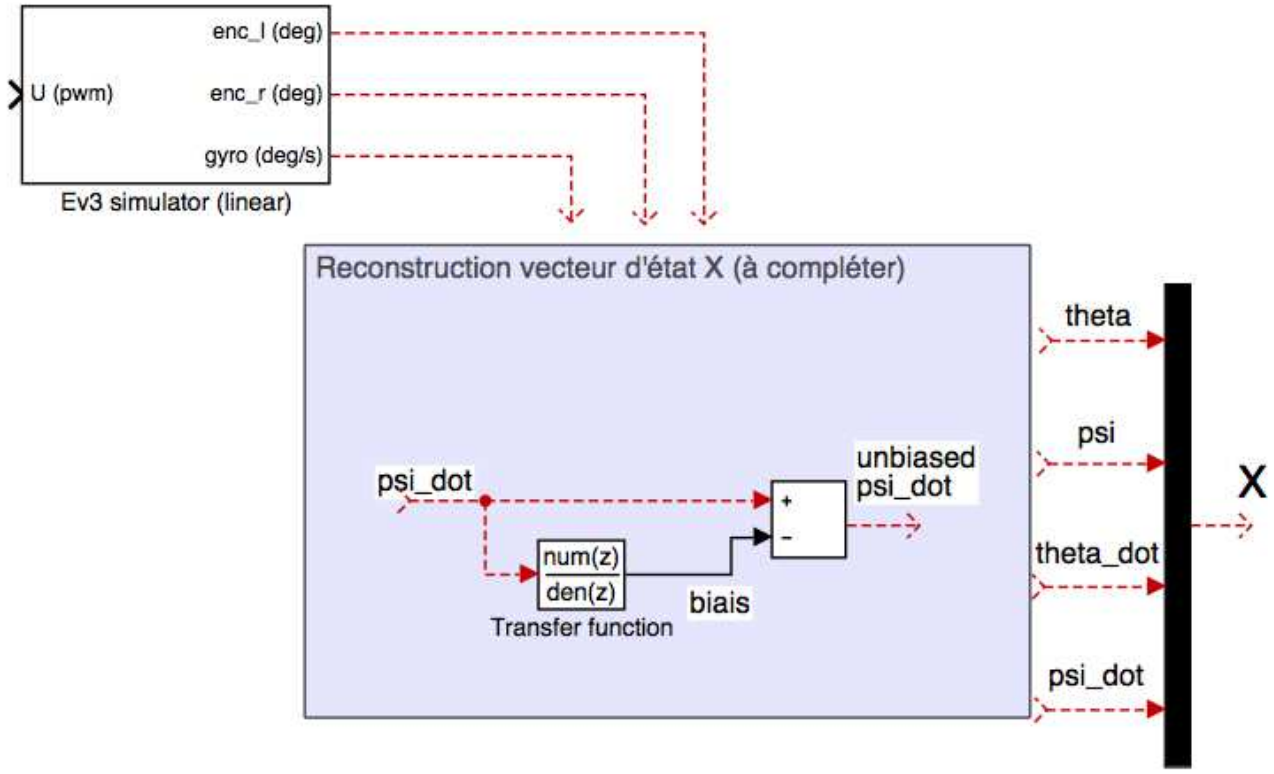

FIGURE 5 – Schéma bloc de la reconstruction de l'état X et estimation d'une vitesse angulaire $\dot{\psi}$ non biaisée pour l'intégration

5. Implémenter de même le calcul de la position angulaire moyenne θ du Segway et sa dérivée $\dot{\theta}$ à partir des signaux issus du codeur droit et gauche du simulateur du Segway Ev3 (voir détail section 1.2).
6. Ayant désormais reconstruit à partir des capteurs disponibles le vecteur d'état X (figure 5), stabiliser le système en ajoutant le gain du retour d'état. Valider par la simulation et l'observation de l'évolution des états.
   En cas de problème, vérifier que la commande en entrée du simulateur Ev3 est bien dans la bonne unité (Quelle est la grandeur en sortie du gain du retour d'état, figure 4 ? Regarder également à l'intérieur du bloc simulateur).
   Rappel : Le vecteur d'état initial dans le simulateur est non nul, sinon il ne se passe rien, car on est déjà à l'équilibre en 0.

[Point de validation : Faire valider par un encadrant]

6.3 Premiers pas avec la programmation via Simulink du Lego Ev3

Dans Simulink, créer une nouvelle simulation vierge. Aller dans la bibliothèque pour trouver les divers blocs permettant de reproduire la simulation de la figure 6.
Cette simulation simple envoie une consigne constante de 10% au moteur connecté au port A, lit la valeur entière retournée par les capteurs (degrés), convertie cette valeur en double et affiche le résultat sur une ligne de l'écran LCD. Les blocs `conversion` (disponibles dans la section Signal Attributes de la bibliothèque) sont nécessaires, car les autres blocs Simulink classiques travaillent uniquement en précision double. Penser à :

— configurer le port de chaque bloc Ev3 et indiquer T_s comme fréquence d'échantillonnage dans les propriétés des capteurs ;

3. On s'interdit d'utiliser des blocs contenant des fonctions de transfert continues (en s), car :
— un calculateur numérique travaille à temps échantillonné et donc l'état sera reconstruit à temps échantillonné.
— toutes les fonctions de transfert doivent être échantillonnées et de période commune T_s afin que Simulink puisse générer le code C correspondant à la simulation.

— choisir une ligne d'affichage différente pour le codeur et le gyroscope.

1. Allumer et connecter la brique Ev3 au PC avec le câble USB fourni. Pour compiler et exécuter ce schéma-blocs sur la brique, il vous faut la toute première fois aller dans l'onglet `Modeling`, puis sélectionner `Model Settings` et `Hardware implementation`. Dans le panneau de configuration qui s'ouvre, choisir `LEGO EV3` pour la plateforme `Hardware` et `build, load and run` pour `build options` dans `hardware ressource`, ainsi que l'usb pour la connection `host to hardware connection`. Appliquer les changements et fermer le panneau.
2. Cliquez sur l'icône "Build, deploy & start" dans le bandeau Simulink, onglet Hardware. Le programme compile puis se lance automatiquement sur la brique après la fin de la compilation.
   Note 1 : Si on modifie la simulation, un clic sur l'icône suffit pour lancer la nouvelle simulation.
   Note 2 : Après une première exécution, on peut lancer le programme sans être connecté au PC en allant dans l'explorateur de fichiers de la brique, répertoire `mw` et sélectionner le nom de la simulation.
3. Grâce à cette simulation, noter le sens positif défini par Simulink pour la rotation des moteurs, les codeurs et le capteur gyroscopique. Correspondent-ils au choix fait lors de la modélisation figure 2 ? Si non, mettre des gains -1 sur les entrées et sorties qui le nécessitent afin d'avoir les mêmes conventions de signe que dans la modélisation (et donc le simulateur Ev3 précédent).

.

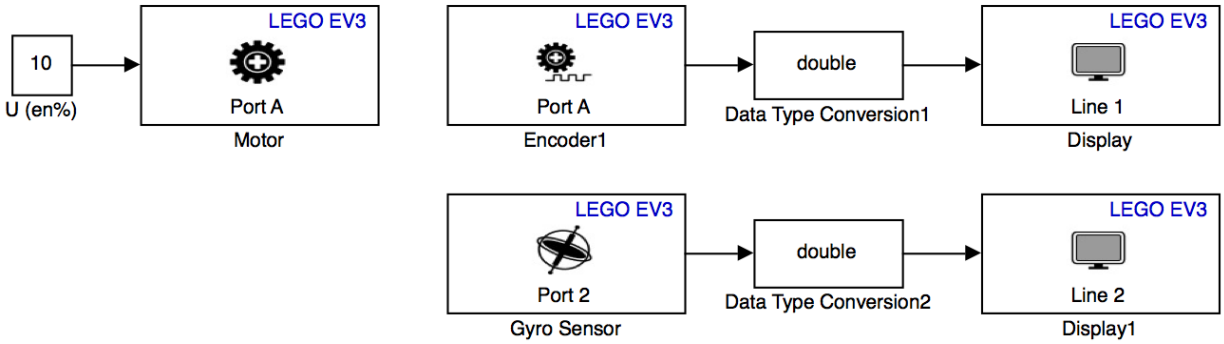

FIGURE 6 – Schéma bloc pour tester les conventions de sens de rotation positif des blocs Ev3

6.4 Implémentation du retour d'état sur le matériel Lego Ev3

L'heure de tester sur la maquette est venue. Le segway va-t-il rester en équilibre ?
Si votre retour d'état est effectif avec le simulateur du lego (`simulation.slx`), il suffit alors de remplacer le contenu du bloc `Ev3(simulateur)` par les blocs Ev3 adéquats de connexions aux moteurs et capteurs de la brique Lego Ev3. Ensuite, Simulink se chargera de générer le code C correspondant à la simulation et de l'uploader sur la brique.

1. Dupliquer votre fichier `simulation.slx` sous un autre nom. Dans cette copie, renommer le sous-système `Lego EV3 (Simulateur)` en `Lego Ev3 (hardware)`. Ouvrir le sous-système et effacer tous les blocs à l'exception du port d'entrée `U` et des 3 ports de sorties `enc_l`, `enc_r` et `gyro`.
2. Connecter alors les ports d'entrées et sorties aux blocs Ev3 motor, Encoder et Gyro Sensor correspondant afin de remplacer le simulateur par le système réel. Comme dans votre simulation précédente, penser à :
   — ajouter les blocs `conversions` en double en sortie des capteurs ;
   — ajouter un gain -1 si le signe du capteur n'est pas le même que celui du modèle ;
   — dans les propriétés de chaque bloc, définir le port de connexion et T_s comme fréquence d'échantillonnage des capteurs.
3. Préparer le déploiement sur le Lego Ev3 comme précédemment via l'onglet l'onglet `Modeling` dans le bandeau Simulink. Configurer et déployer sur le Lego.
   Tester :

— sélectionner le programme dans le menu ;
— mettre le segway à la verticale : chercher manuellement le point d'équilibre vertical du Lego (il faut qu'au démarrage, $\psi = 0$ car on obtient ψ par intégration en partant de 0) ;
— tenir délicatement le Segway et appuyer sur le bouton central pour lancer le programme ;
— lâcher immédiatement.
Si tout a bien été fait, il reste en équilibre !

[Point de validation : Faire valider par un encadrant]

7 Asservissement en poursuite du Segway Ev3

Souvent, en plus de stabiliser le système, on veut pouvoir imposer son régime permanent. En particulier asservir la sortie du système Y à une consigne de référence Y_{ref}. On introduit donc une référence dans le schéma de commande et un gain de préfiltre $\bar{N}$ qui assure un gain statique de 1 entre Y et Y_{ref} en régime permanent (figure 7).

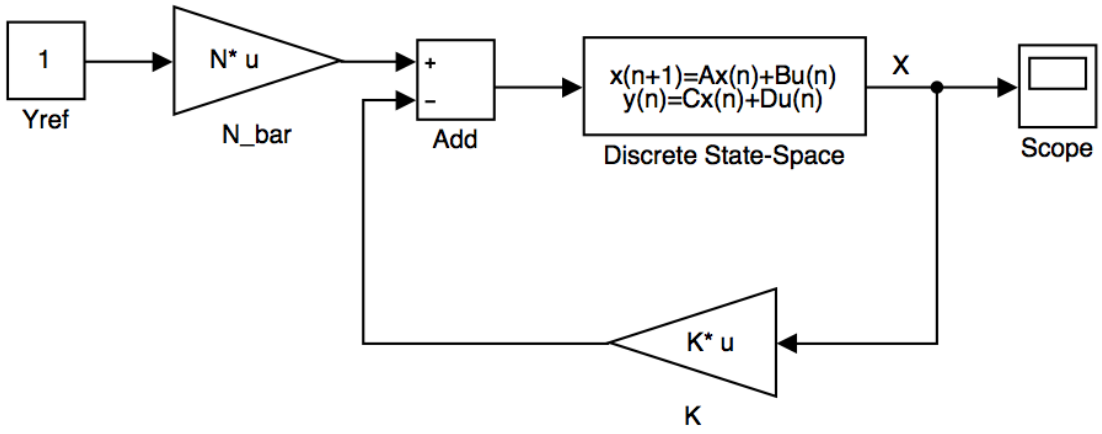

FIGURE 7 – Structure complète d'une commande par retour d'état

1. Les seuls régimes permanents correspondent à un état d'équilibre. Quelle variable du vecteur d'état est-il possible d'imposer à une référence dans le cas du Lego Segway ?
2. Le schéma de régulation figure 7 est décrit par la représentation d'état suivante :

$$X(k+1) = (\Phi - \Gamma K)X(k) + \Gamma \bar{N} Y_{ref} \tag{19}$$
$$Y = C\,X(k) \tag{20}$$

Une fois le régime permanent atteint, par définition, quelle relation lie $X(k+1)$ et $X(k)$?
Montrer alors qu'en régime permanent et avec les équations précédentes, $Y = Y_{ref}$ impose que $\bar{N} = [C(\mathbf{I} + \Gamma K - \Phi)^{-1}\Gamma]^{-1}$ dans le cas discret.
3. Calculer la valeur du gain $\bar{N}$ en choisissant C en fonction de la variable de sortie à réguler dans le système ci-dessus. Compléter `Simulation.slx` contenant le simulateur du lego Ev3 avec la structure complète du retour d'état. Vérifier alors qu'en régime permanent la sortie est égale à la consigne.

7.1 Détection d'obstacle et asservissement en poursuite du Lego Segway Ev3

1. Imposer θ, soit la position au sol $x = r\theta$, est souvent moins intéressant qu'imposer une vitesse. De plus, il n'est pas possible de demander une valeur de θ_{ref} très différente de la valeur courante, θ car cela se traduit par un transitoire de commande très élevé.
On préfère donc ne pas donner des échelons de référence, mais une rampe de position, c.-à-d. piloter en vitesse : remplacer la consigne constante de position par une consigne croissante de position en utilisant un intégrateur discret et une consigne de vitesse angulaire $\dot{\theta}_{ref}$. Tester en simulation.

2. Tester ensuite sur la maquette Lego Segway en imposant une vitesse constante (pas trop rapide).
3. On souhaite asservir le Segway tel qu'il avance à vitesse constante en l'absence d'obstacle et recule à vitesse constante en présence d'un obstacle.
   Modifier le schéma-bloc en ajoutant le bloc capteur à ultrasons dans le sous-système lego Ev3. Ce capteur renvoie une distance en cm. Utiliser alors un bloc switch (figure 8) qui permet de basculer d'une consigne à une autre en fonction d'une condition sur un signal externe (la mesure du capteur à ultrasons).

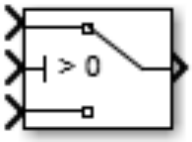

FIGURE 8 – Block switch

8 TP, partie finale : Démontage du Lego Segway

[Déposer vos fichiers .m et simulink (.slx) finaux dans le dépot Moodle séance 2]
Démonter le robot. Mettre dans le fond de la boîte les éléments de la figure 9!!
Laisser votre session Matlab ouverte.

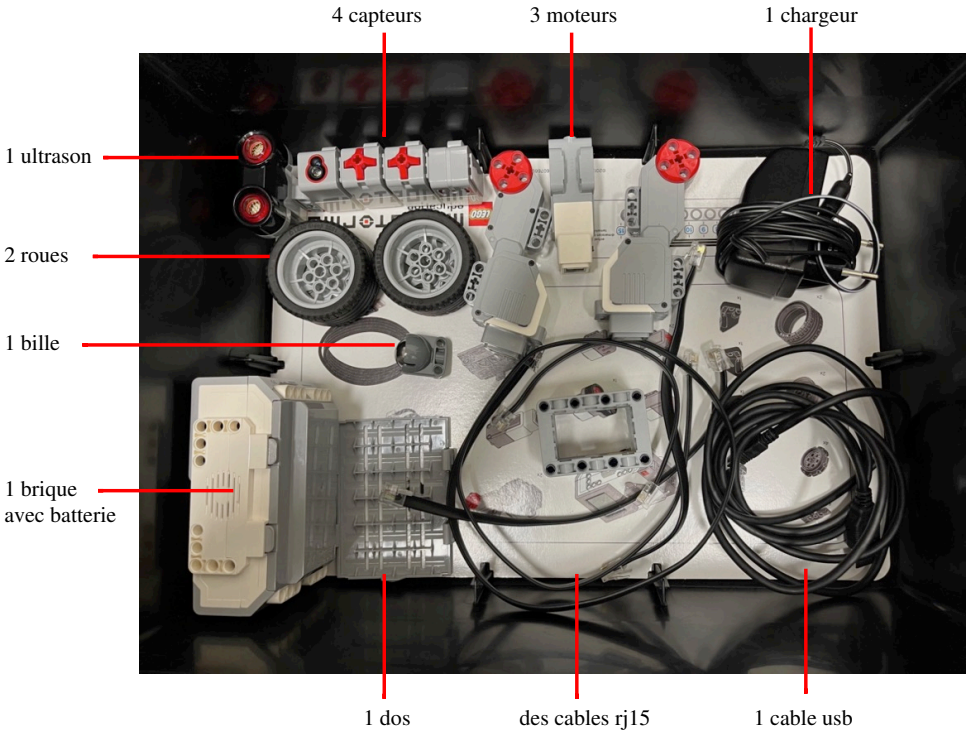

FIGURE 9 – Contenu attendu au fond de la boîte