Notice explicative du package *autom_demos*

1 Régulation de vitesse

1.1 speed.py

On considère la régulation d'une vitesse v à l'aide d'une commande d'accélération u. Le système dynamique est simplement défini comme un intégrateur :

$$\dot{v} = u \tag{1}$$

Ce système peut être régulé vers une référence constante v_{ref} via un correcteur proportionnel de la forme $u = -k_p\tilde{v}$ avec $\tilde{v} = v - v_{\text{ref}}$ et $k_p \in \mathbb{R}$ le gain du correcteur. Le système bouclé est alors

$$\dot{\tilde{v}} = -k_p\tilde{v} \tag{2}$$

Ce système dont le pôle unique est égal à $-k_p$ est donc stable si $k_p > 0$. Le script **speed.py** permet de simuler cette première boucle en temps continu (topics /**ref** et /**vel** à visualiser, paramètres à modifier directement dans le script).

1.2 speed_continu_discret.py

En temps discret (période dt), le système (1) devient :

$$\tilde{v}_{k+1} = \tilde{v}_k + dt \cdot u_k \tag{3}$$

En appliquant le même correcteur $u_k = -k_p\tilde{v}_k$, on obtient le système bouclé en temps discret

$$\tilde{v}_{k+1} = (1 - dt \cdot k_p)\,\tilde{v}_k \tag{4}$$

dont le pôle unique est égal à $(1 - dt \cdot k_p)$. Ce système est stable sans oscillations seulement si $0 < k_p \leq (dt)^{-1}$. Cette condition peut être testée dans le script **speed_continu_discret.py** où il est possible de comparer le comportement du système à temps continu et à temps discret en modifiant les différents paramètres (topics /**ref**, /**vel**, /**veld** à visualiser).

1.3 speed_observer.py

On considère le système à temps discret en boucle ouverte

$$v_{k+1} = v_k + dt \cdot u_k$$

où l'entrée u_k (aléatoire) et la vitesse mesurée v_k sont ici connues avec un bruit de mesure additif. On applique alors un observateur de Luenberger pour filtrer la vitesse à partir de ces deux quantités, celui-ci s'écrit comme :

$$\hat{v}_{k+1} = (1 - L) \cdot (\hat{v}_k + dt \cdot u_k) + L \cdot v_k$$

où u_k, v_k sont l'entrée et la mesure bruitée et $\hat{v}_k$ l'estimée du filtre à l'instant k. On voit ici que ce filtre se résume à un filtre du premier ordre ou encore une somme pondérée (par le gain L de l'observateur) entre la grandeur prédite et la grandeur mesurée. Le script **speed_observer.py** permet de tester ce filtre en faisant varier les différents paramètres, les topics à visualiser étant /**vel** (vraie vitesse non bruitée), /**vel_noisy** (mesure de vitesse bruitée) /**vel_obs** (vitesse filtrée).

2 Pendule inverse

Le script **inv_pendule.py** simule un pendule inverse présentant un angle θ par rapport à la verticale selon le modèle simplifié $\ddot{\theta} = \frac{g}{l}(\sin\theta - u)$, le modèle linéarisé sous hypothèse de petit déplacement angulaire est $\ddot{\theta} = \frac{g}{l}(\theta - u)$. Ce système s'écrit sous forme d'état avec $x_1 = \theta$ et $x_2 = \dot{\theta}$ comme suit :

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{g}{l} & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ -\frac{g}{l} \end{bmatrix} \mathbf{u}$$
$$y = [1, 0] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Afin d'asservir θ et $\dot{\theta}$ à 0, on applique le correcteur par retour d'état (proportionnel dérivé) :

$$u = \begin{bmatrix} -k_p & -k_d \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = -k_p\theta - k_d\dot{\theta}$$

Le script fourni permet d'observer le comportement du système en fonction des valeurs des gains k_p et k_d du correcteur, les topics à visualiser étant /**theta**, /**dtheta**.