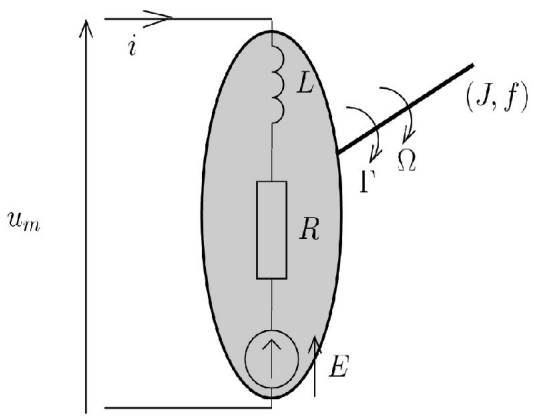
Simulation d’un moteur à courant continu (CC)

Un moteur à courant continu est généralement modélisé comme un système linéaire du second ordre caractérisé par les grandeurs introduites sur le schéma de principe de la figure 1 et qui représentent :

Grandeurs électriques :
— R : résistance de l’induit ;
— L : inductance de l’induit ;
— E(t) : force contre-électromotrice ;
— U_m (t) : tension au bornes du moteur ;
— i(t) : courant ;

Grandeurs mécaniques :
— J : inertie du rotor ;
— f : frottements visqueux ;
— Γ(t) : couple moteur ;
— Ω(t) : vitesse du rotor.



Le modèle se construit à partir des relations électriques et mécaniques suivantes :

— le couple moteur : $\Gamma(t) = k_c\, i(t)$ (k_c : constante de couple) ;

— la force contre-électromotrice : $E(t) = k_e\, \Omega(t)$ (k_e : constante contre-électromotrice) ;

— l’équation électrique : $U_m(t) = E(t) + Ri(t) + L\,di(t)/dt$;

— l’équation mécanique : $Jd\Omega(t)/dt + f\,\Omega(t) = \Gamma(t)$.

L’objectif est de créer un modèle numérique du moteur à courant continu, avec U_m comme entrée, et vitesse Ω & couple Γ comme sortie.

Voir https://slides.com/sinanh/uerob_td2-1/fullscreen pour rappel de fonctionnement du moteur CC

1. **Solution analytique** : Établir la solution analytique de Ω(t) en fonction de U_m(t) avec la simplification L ≈ 0H

2. **Simulation** : Construire une classe ‘MoteurCC’ et l’intégrer dans un simulateur. Pour valider votre modèle, tracer la réponse indicielle théorique en boucle ouverte (échelon unité de tension) avec votre simulateur et comparer à la réponse théorique. Quelle est l’influence de la simplification L=0 ? Comment enrichir votre modèle en ajoutant les actions extérieures (inertie de la charge, couples extérieurs, viscosité du milieu...)
Pour cette partie, on utilisera ces valeurs pour explorer et valider le modèle :
— résistance de l’induit : 1Ω ;
— inductance de l’induit : 0,001H ⇒≈ 0H ;
— constante de couple : 0, 01N m/A ;
— constante de la fcem : 0, 01V.s ;
— inertie du rotor : 0,01kg.m2 ;
— constante de frottement visqueux : 0,1N m.s.

3. **Commande** : Insérez votre modèle dans un schéma de commande en boucle fermé pour la régulation de la vitesse. Explorer les cas d’un correcteur proportionnel puis proportionnel+intégrateur (temps de réponse, erreur statique, Tension max, Ωmax…). On peut par exemple créer un ‘contrôleur’ qui a comme entrée la vitesse désirée et la vitesse actuelle et génère la tension pour piloter le moteur. Montrer l’influence des gains P et I .

Prototypes

Classe MoteurCC
attributs :
- Caractéristiques physiques : grandeurs électriques & mécaniques
- Entrées : Tension, charge (inertie ajoutée), couple extérieur résistif, viscosité
- Sorties : Courant, couple, vitesse, position

méthodes :
- _init_, _str_, _repr_
- setVoltage(V)
- getPosition(), getSpeed(), getTorque(), getIntensity()
- simule(step)
- plot()

```
% Validation
% fonctionnement boucle ouverte en vitesse

m = MoteurCC(…..)

t = 0
step = 0.01
temps = [t]

while t<2 :
     t=t+step
     temps.append(t)

     m.setVoltage(1)
     m.simule(step)

figure()
plot(m.speed,temps)
show()
```

Classe ControlPID_vitesse :

attributs :

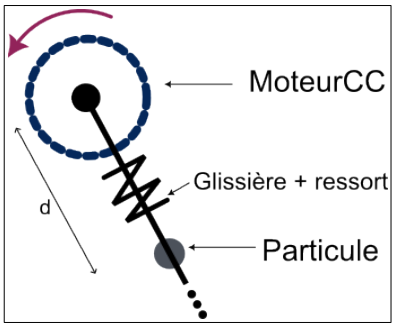
- paramètres K_P,K_I,K_D, moteurCC à contrôler
- Entrée : Vitesse désirée
- Sortie : tension envoyée au moteur ; vitesse du moteur

Méthodes
- _init_, _str_, _repr_
- setTarget(vitesse)
- getVoltage()
- simule(step) [avec m.setVoltage(V) et m.simule(step)]
- plot()

```
% Validation du fonctionnement en boucle fermé, en vitesse

m_bo = MoteurCC(…) # moteur en boucle ouverte
m_bf = MoteurCC(…) # moteur boucle fermé pour le controleur
control = ControlPID_vitesse(m_bf,P,I,D)
t = 0
step = 0.01
temps = [t]

while t<2 :
     t=t+step
     temps.append(t)

     m_bo.setVoltage(1/K)  # pour avoir la même réponse en régime
permanent
     control.setTarget(1)  # rad/s
     control.simule(step)  # m_bf.setVoltage et simule seront
appelés ici

     m_b0.simule(step)

figure()
plot(m_bo.speed,temps)
plot(m_bf.speed,temps)
legend()
show()
```

Simulation [moteur centrifugeuse]

1. Intégrer le MoteurCC (BO et BF) dans l'architecture « Univers »
2. Simuler le système moteur + ressort +amortisseur+ particule de la figure, tracer la distance d en fonction de la vitesse de rotation en régime permanent, avec un k judicieusement choisi.



Commande en position du moteur CC

Implémenter une commande en position (PID) du moteur CC. Montrer l'influence des gains P et D.

Turtlebots

Améliorer le modèle cinématique des TurtleBots en implémentant les roues : la commande de déplacement s'effectuera par le contrôle de rotation des roues. Dans un premier temps, la vitesse des roues seront imposées directement. Puis, ajouter des moteurs CC sur chaque roue pour commander les robots à travers la tension des moteurs (ou le contrôleur en vitesse précédemment développé). Illustrer la différence de comportement dans les 2 cas, par exemple sur un suivi de trajectoire.

Simulation des barres 2D

Proposer un objet de type barre dynamique en mouvement plan 2D.

Respecter ces prototypes :

```
class Barre2D(object):

    def __init__(self, mass=1, long=1, theta=0, pos=V3D(), 
fixed=False, color='red', nom='barre')

    def applyEffort (self, Force=V3D(), Torque=V3D(),  Point=0)

     # Point d'application [-1,1] d'une extrémité à l'autre, avec 
0 centre de masse

     #  (ou Torseur ou lieu de Force / Torque)

    def simulate(step) :

    def plot(self):

    def plotRot(self) :

    def gameDraw(self,scale,screen):
```

Modifier la classe Univers pour qu'elle soit compatible avec les Barre2D. Adapter ou créer ces générateurs d'effort pour les Barre2d :
(Attention, sur une barre il peut être nécessaire préciser le point d'application) :

- Gravity
- Force , ForceSelect → Effort (Force et Couple)
- SpringDumper, TorsionSpringDumper (ressort-amortisseur en rotation)
- Liaisons cinématiques : Revolut, Prism (à partir des *class Spring* par exemple)

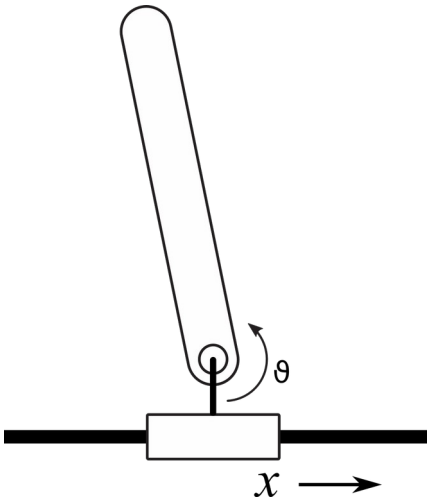
Proposer des tests unitaires qui démontrent le comportement mécanique de base et valident chaque générateur.

Validations :

1. Montrer que la fréquence propre d'une pendule de longueur L est identique à une barre accrochée à une extrémité de longueur 2L
2. Créer 2 pendules (barres) couplés avec un ressort-amortisseur à leur extrémité. Montrer l'existence de 2 modes propres. (Bonus : forcer un des pendules à une fréquence contrôlée par l'utilisateur pour observer les 2 modes)
3. Une autre illustration de votre choix (différent de votre camarade!)

Asservissement de l'équilibre d'un pendule inverse

Un système est modélisé par une pendule inverse montée sur une base mobile. L'articulation $\boldsymbol{\theta}$ est passive, et l'axe $\boldsymbol{x}$ est piloté en force, contrainte par une liaison glissière.



1. Mettre en place la simulation d'un tel système.
   La base mobile peut être modélisé par une barre horizontale ou une particule en liaison glissière d'axe *x* avec le bâti.
   Illustrer en simulant le mouvement pendulaire libre de la barre (ou particule+tige) avec la base mobile bloquée ou pas. Valider en comparant avec la solution analytique.

2. Utiliser le simulateur pour implémenter :
   1. Commande manuelle : l'utilisateur garde l'équilibre avec des touches gauche et droite, en déplaçant la base mobile.
   2. Automatique : synthétiser un asservissement de la base mobile pour maintenir le pendule inverse en position verticale. La force générée par le correcteur est supposée s'appliquer sur la base mobile selon *x*.
      Ajouter des perturbations interactives : l'utilisateur peut donner des « pichenettes » à gauche ou à droite avec le clavier pour perturber l'équilibre.
   3. Inclure un moteur CC pour contrôler le déplacement de la base mobile.

Simulation d'un Robot 2R-plan

En utilisant les éléments déjà créés, établir un simulateur pour un robot 2R plan, commandé à travers le couple aux articulations. ***Attention : un moteur placé entre 2 segments d'un robot applique un couple réciproque sur les deux corps.***

1. Dans un premier temps, construire un système 1R (barre en pendule avec un couple appliqué sur la liaison pivot). Synthétiser une commande en boucle fermé pour obtenir un angle $\boldsymbol{\theta}$ désiré. On peut par exemple faire une barre qui pointe vers un objet qu'on déplacerait au clavier.
   Motiver le choix du correcteur (P, PI, PID?) et illustrer son fonctionnement avec des gains différents (donc tracer des courbes de réponse)

2. Établir un simulateur 2R dynamique. Montrer sa commande dans l'espace articulaire avec un schéma adapté, en imposant par exemple l'angle pour chaque articulation.

3. Synthétiser une commande pour atteindre un point de l'espace de travail (modèle géométrique inverse). On peut par exemple acquérir la cible en cordonnées généralisées par un clic de la souris (voir *pygame.mouse.get_pos(), pygame.mouse.get_pressed()*)

4. Idem en ajoutant un suivi de trajectoire en ligne droite, à vitesse constante (modèle cinématique inverse)

- Pas besoin de démontrer les modèles géométriques / cinématiques.
- Voir ici pour l'analyse d'un robot plan.

Instructions pour le rendu

- Un compte rendu en pdf est obligatoire, le code tout seul n'est pas recevable. Il doit illustrer et décrire le travail réalisé, expliquer les choix techniques (quelle classe/méthodes, pourquoi – éviter quand même la répétition avec des choses vues en cours), mettre en valeur votre implémentation (surtout si différente de ce que j'avais suggéré), enfin décrire et illustrer les résultats. Inclure des schémas, **courbes et captures d'écran**.

- Les codes pour les modules et les scripts demandés (***Run_XXX.py***). Inclure des sections de test et de validation dans les modules. Commenter *raisonnablement* le code. Si le script est interactif, ajouter un affichage à l'exécution les instructions (*print* quelles touches pour faire quoi)

- Auto-Évaluation et commentaires, à inclure à la fin du CR :
  Attribuez-vous une note sur 5, qui prend en compte votre assiduité, votre participation, les efforts consacrés, travail personnel, et les progrès que vous avez fait. Justifier avec un texte de longueur raisonnable.

- La qualité du rendu a un vrai impact sur la note.
- L'utilisation du *LateX* et du *git* est conseillée.