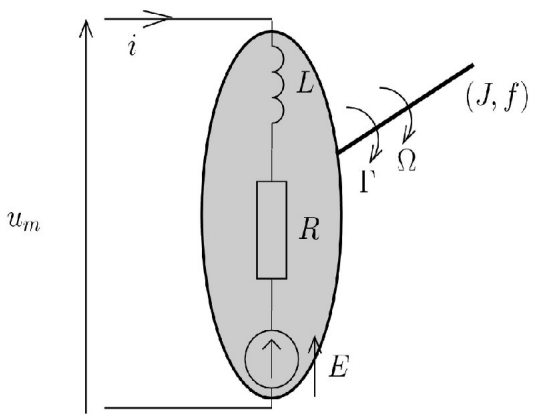
1e Partie : Simulation d'un moteur à courant continu (CC).

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.

4. Même question, avec une **commande en position**

Prototypes

Classe MoteurCC
attributs :
- Caractéristiques physiques : grandeurs életriques & 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)
-

```
# 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 PID, moteurCC contrôlé
- Entrée : Vitesse désirée
- Sortie : tension envoyée au moteur

Méthodes

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

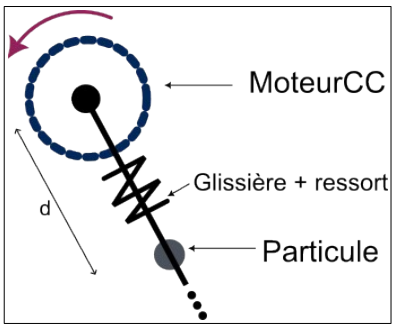
```
# Fonctionnement en boucle fermé, en vitesses
#

m_bo = MoteurCC(…..)
m_bf = MoteurCC(...)
control = ControlPID_vitesse(m_bf,P,I,D)
t = 0
step = 0.01
temps = [t]

while t<2 :
     t=t+step
     temps.append(t)
     control.setTarget(1)
     m_bo.setVoltage(1)

     m_bo.simule(step)
     control.simule(step)

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

Classe controlPID_position(moteur,P,I,D) :

?

2e partie : Simulation [moteur + particule]

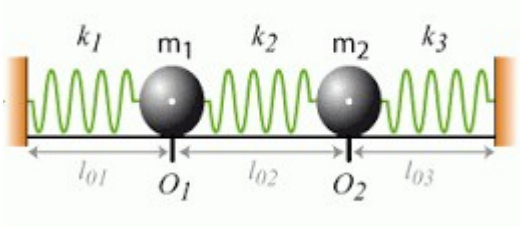
1. Intégrer le MoteurCC (BO et BF) dans l'architecture « Univers »
2. Faire en sorte de pouvoir simuler le système moteur + ressort + particule de la figure
3. Tracer la distance d en fonction de la vitesse de rotation, avec un k judicieusement choisie.



3e partie : Validations divers

Réaliser les simulations suivantes, les illustrer dans votre rapport avec des captures d'écran et des courbes commentées. Inclure un script py séparé pour chacun. Assurez-vous que les scripts se terminent proprement (*pygame.quit(), sys.exit()*). *Je ne déboguerai pas les scripts qui ne fonctionnent pas.*

1. Un système masse + (ressort+amortisseur) avec application au clavier d'une force constante ou une force harmonique. Vérifier l'accord avec le comportement théorique (comparer les fréquences propres théoriques et simulés).
2. Trois pendules de l=10, 20 et 30 cm. Vérifier l'accord avec le comportement théorique. (bonus : https://www.youtube.com/watch?v=yVkdfJ9PkRQ)
3. Une pendule double, avec l= 15 et l2=10
4. Un système 2 ddl couplé 2 masses + 3 ressorts, avec m1 = m2 et k1=k3, l1 = l2 = l3 = 10 cm. Montrez qu'on retrouve bien les 2 modes propres en phase et opposition de phase aux bonnes fréquences théoriques. (bonus: pour ddl =3, 4 ou plus)



5. Reproduction du TP pendule inverse sur base mobile : créer en simulation un système équivalent, effectuer l'identification pour retrouver les paramètres choisis.
    1. Commande au clavier : l'utilisateur garde l'équilibre avec des touches gauche et droite, en déplaçant la base mobile.
    2. Commande automatique : proposer un contrôleur qui assure l'équilibre. Les touches gauche et droite perturbent l'équilibre par une petite force appliqué sur la masse en haut.

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. Un bon rapport se lit sans avoir besoin d'aller voir le code.
- 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.
- Sur la page de titre, bien mettre votre nom, numéro d'étudiant et **une photo d'identité.**