LEPL 1504 - Mission 1
Réponse dynamique d'un système masses-ressorts-amortisseurs

Février 2025

1 Enoncé

Cette mission consiste à modéliser un système dynamique simple pouvant représenter un véhicule et ses suspensions (contenant chacune un ensemble ressort-amortisseur).

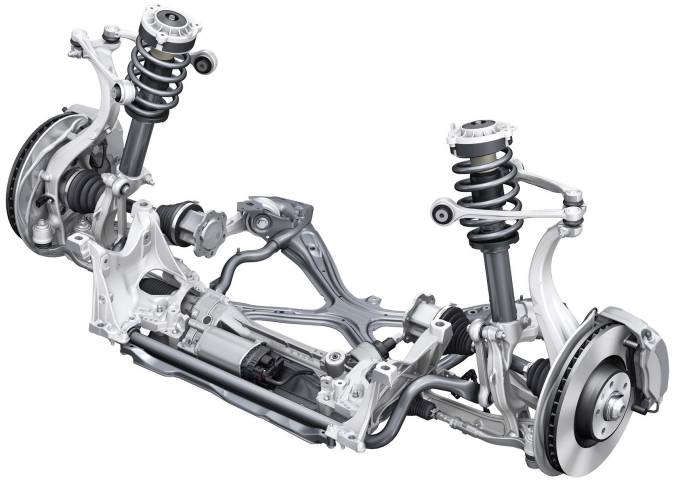

FIGURE 1 – Illustration d'un train de suspension avant d'une Audi A4 (Image de https ://www.audi-technology-portal.de/en/).

La figure 1 illustre un train avant avec les ressorts hélicoïdaux bien visibles (les roues ne sont pas montrées).

L'utilisation de suspensions permet d'améliorer à la fois le confort et la tenue de route d'un véhicule.

Néanmoins, comprendre le comportement dynamique d'un véhicule soumis à différents types d'excitations n'est pas du tout évident et prévoir la réponse du combiné ressort-amortisseur est un beau challenge.

Cette mission aura pour objectif de redécouvrir un intégrateur numérique un peu plus complexe que le schéma d'Euler explicite et de vous réapproprier la programmation en Python, sur la base d'une application mécanique concrète. Dans ce cas-ci, il s'agira d'analyser la réponse dynamique (ODE[1] du 2ème ordre) d'un système simple à deux degrés de liberté.

Pour cette mission, nous allons donc représenter le véhicule comme un système à deux corps reliés au sol et entre eux par un ressort et un amortisseur. Une telle approche, bien que fort simplifiée, permet déjà de mettre en évidence de nombreuses informations utiles (réponse en fréquence, stabilité, etc.) et est souvent employée, par exemple, pour concevoir le contrôleur de suspensions actives.

La première tâche sera d'effectuer une mise à l'équilibre du système à partir de conditions initiales, grâce à la dynamique directe.
La seconde consistera à calculer le mouvement des deux corps lorsqu'une force extérieure est appliquée sur la masse supérieure. La force externe aura une forme sinusoïdale d'amplitude constante et de fréquence variable.

1. *Ordinary Differential Equation*

2 Modèle

L'objectif est de pouvoir analyser le comportement dynamique du modèle "double masse-ressort" qui permet de décrire la dynamique verticale d'un quart de voiture (une suspension). Le corps supérieur (masse m_2, dite "suspendue") représente le châssis tandis que le corps inférieur inclut tous les éléments de la suspension (masse m_1, dite "non-suspendue"), cfr Fig 2. Les symboles k_1 et d_1(resp. k_2 et d_2) désignent la raideur et l'amortissement du pneu (resp. de la suspension).

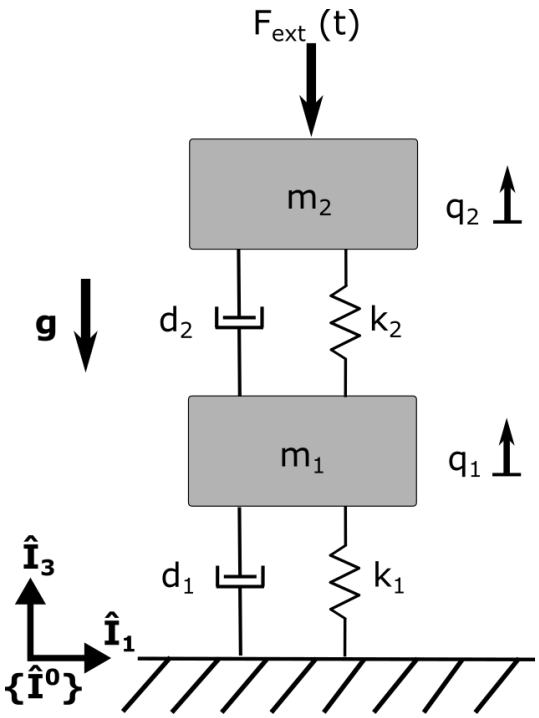

FIGURE 2 – Illustration du système à modéliser.

F_{ext} est la force externe appliquée sur le corps 2 en fonction du temps.

Le repère inertiel $\{I^0\}$ est tel que représenté ci-contre, tandis que les variables généralisées (= les degrés de liberté) sont définies en terme de positions **relatives** :

- $q_1(t)$ est la position relative de m_1 par rapport au sol,
- $q_2(t)$ la position relative de m_2 par rapport à m_1.

Différentes hypothèses simplificatrices sont les suivantes :

- Masses supposées ponctuelles
- Ressorts et amortisseurs supposés sans masse
- Mouvement purement vertical
- Pas de frottement (air, ressorts, amortisseurs)
- Ressorts et amortisseurs linéaires et sans butées
- Pas de contact entre corps

La loi de comportement d'un ensemble ressort-amortisseur est la suivante (en faisant abstraction du signe de la force) :

$$F = k\,(z - z0) + d\,\dot{z}$$

Avec $(z - z0)$ l'élongation du ressort, $\dot{z}$ la vitesse dans l'amortisseur, z0 la longueur neutre du ressort, k la raideur du ressort et d le cœfficient d'amortissement.

La dynamique directe d'un système à plusieurs corps consiste à calculer les accélérations généralisées $\ddot{\mathbf{q}}$ pour une configuration donnée ($\mathbf{q}$, $\dot{\mathbf{q}}$) du système sur lequel des forces (ou couples) peuvent être appliquées.

Les équations de la dynamique directe sont utilisées pour prédire le mouvement du système ($\mathbf{q}(t)$, $\dot{\mathbf{q}}(t)$) en partant d'une configuration initiale ($\mathbf{q}(t=0)$, $\dot{\mathbf{q}}(t=0)$) et en intégrant temporellement les accélérations $\ddot{\mathbf{q}}$.

Les équations de translation pour les deux corps du système double masse-ressort peuvent être dérivées (à la main) et mises sous la forme matricielle suivante [2] :

$$M(q)\,\ddot{q} + c(..., frc, g) = Q(q, \dot{q})$$

avec

- $M(q)$ [2x2], la matrice de masse généralisée (symétrique)
- q [2x1], les coordonnées généralisées relatives
- $\dot{q}$ [2x1], les vitesses généralisées relatives
- $\ddot{q}$ [2x1], les accélérations généralisées relatives
- $c(..., frc, g)$ [2x1], le vecteur dynamique non linéaire qui contient notamment les termes gravitationnels ainsi que la contribution des forces externes *frc*
- $Q(q, \dot{q})$ [2x1], les forces généralisées agissant dans les articulations, c'est-à-dire dans les ressorts et amortisseurs dans ce cas-ci

Les accélérations $\ddot{q}$ peuvent donc être obtenues en résolvant le système linéaire ci-dessus. Ensuite, les vitesses et positions peuvent être intégrées temporellement via un intégrateur numérique. Dans cette mission, vous pouvez utiliser la fonction *solve_ ivp* de la librairie scipy [3] pour faire l'intégration. Vous allez voir dans la documentation les paramètres qui définissent la fonction et analyser leur impact sur la qualité de la solution pour faire le choix de ceux qui sont les plus appropriés.

Il faut souligner que, pour utiliser la fonction *solve_ ivp*, on ne pourra pas mettre directement la fonction *compute_ derivatives* comme argument, car elle a 3 arguments au lieu de 2. Il faudra appliquer la modification suivante à partir de la fonction lambda :

```
fprime = lambda t,y: compute_derivatives(t, y, data)
```

Les plus attentifs auront remarqué que le comportement dynamique du système se formule sous la forme d'un système d'équations différentielles du second ordre. Pour pouvoir utiliser la fonction mentionnée, une petite manipulation des vecteurs d'état $\mathbf{q}$ et $\dot{\mathbf{q}}$ est nécessaire afin de se ramener à un système du premier ordre :

$$\mathbf{q} = \begin{pmatrix} q_1 \\ q_2 \end{pmatrix} \quad \text{et} \quad \dot{\mathbf{q}} = \begin{pmatrix} \dot{q_1} \\ \dot{q_2} \end{pmatrix}$$

En posant $\mathbf{y}$ et $\dot{\mathbf{y}}$:

$$\mathbf{y} = \begin{pmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \end{pmatrix} = \begin{pmatrix} q_1 \\ q_2 \\ \dot{q_1} \\ \dot{q_2} \end{pmatrix} \quad \text{et} \quad \dot{\mathbf{y}} = \frac{d\mathbf{y}}{dt} = \begin{pmatrix} \dot{q_1} \\ \dot{q_2} \\ \ddot{q_1} \\ \ddot{q_2} \end{pmatrix}$$

L'intégration numérique s'effectuera donc sur les nouveaux vecteurs d'état $\mathbf{y}$ et $\dot{\mathbf{y}}$.

2. Notez que l'ajout d'un point au-dessus d'une variable dénote sa dérivée temporelle, deux points correspondant à une dérivée seconde : $\dot{f} = \frac{df}{dt}$; $\ddot{f} = \frac{d^2 f}{d^2 t}$

3. Fonction Runge Kutta de la librairie scipy (`https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.solve_ivp.html`).

3 Mission

Les paramètres du modèle pourront être déterminés à partir du tableau suivant.

Masse non suspendue	m_1	25 kg
Masse suspendue	m_2	315 kg
Raideur du pneu	k_1	190 kN/m
Raideur de suspension	k_2	37 kN/m
Amortissement du pneu	d_1	107 Ns/m
Amortissement de suspension	d_2	4000 Ns/m
Longueur neutre (pneu)	$z0_1$	0.375 m
Longueur neutre (suspension)	$z0_2$	0.8 m
Gravité	g	9.81 m/s^2

La première étape consiste à observer la mise à l'équilibre du système ($\ddot{\mathbf{q}} = 0$ et $\dot{\mathbf{q}} = 0$). Pour ce faire, les équations dynamiques doivent être résolues et le mouvement doit être intégré temporellement au départ des positions initiales des deux corps jusqu'à la stabilisation du système. Le résultat permet de trouver la position d'équilibre des deux corps pour le jeu de paramètres donné (Fig 3).

Les valeurs d'équilibre (droites horizontales en pointillés sur la figure) peuvent également être obtenues en résolvant analytiquement le système d'équations en posant les accélérations à zéro.

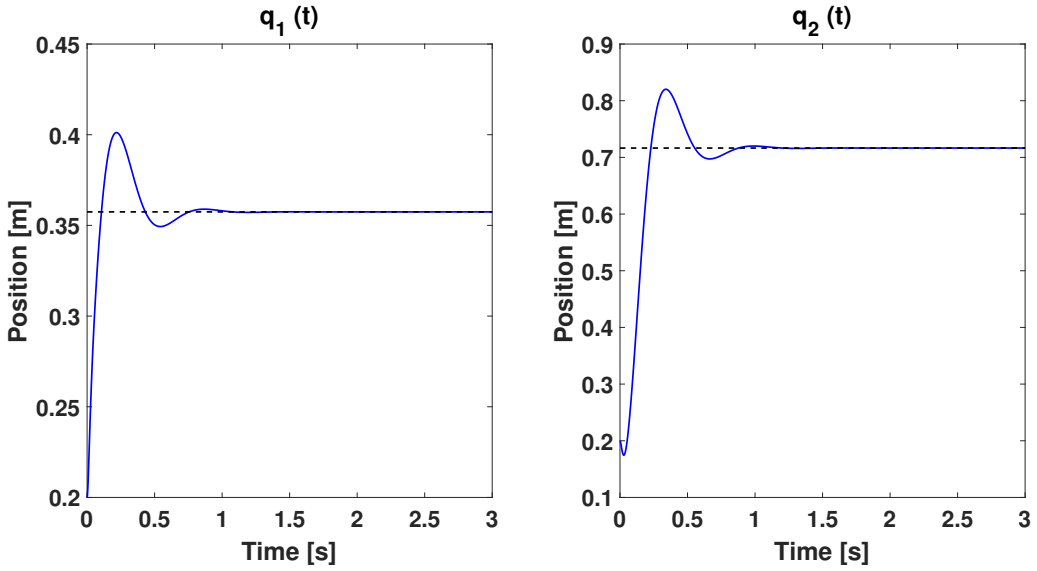

FIGURE 3 – Mise à l'équilibre du système pour deux positions initiales : $q_1 = 0.2\ m$ et $q_2 = 0.2\ m$ ($\ddot{\mathbf{q}} = 0$ et et $\dot{\mathbf{q}} = 0$).

La deuxième étape simule la réaction du système à partir de la position d'équilibre trouvée ci-dessus – prenez les valeurs $q_1(equil) = 0.357445263\ m$ et $q_2(equil) = 0.716482432\ m$ pour débuter la simulation suivante – lorsqu'il est soumis à une force extérieure $F_{ext}(t)$ entre $t_0 = 0\ s$

et $t_1 = 10\ s$ fournie par l'équation suivante :

$$F(t) = F_{max}\ sin\left(2\pi\left(f_0 + \frac{f_1 - f_0}{t_1 - t_0}\frac{t}{2}\right)t\right)$$

avec $F_{max} = 10\ kN$, $f_0 = 1\ Hz$ en $t_0 = 0\ s$ et $f_1 = 10\ Hz$ en $t_1 = 10\ s$.

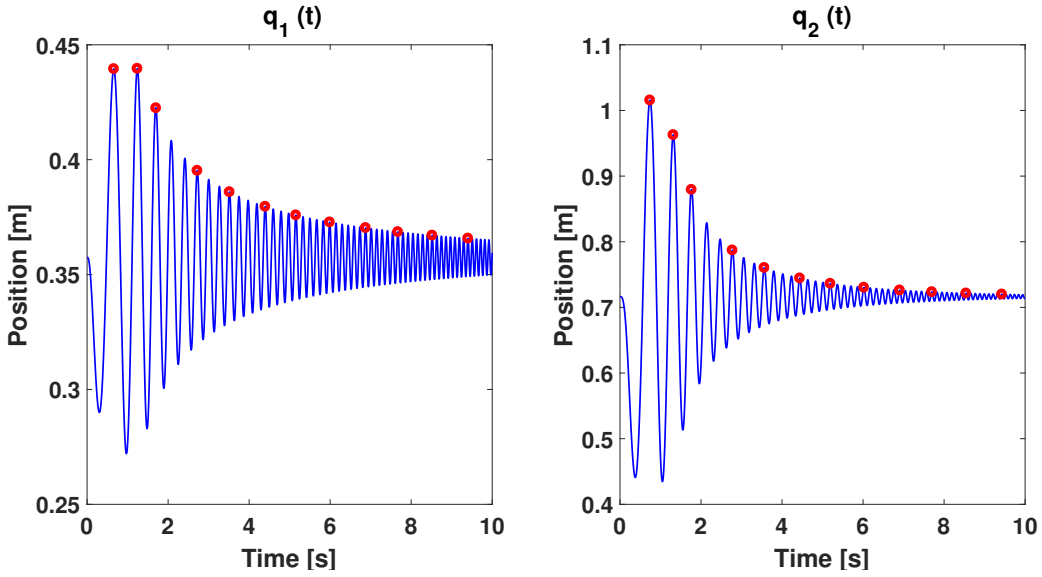

FIGURE 4 – Evolution temporelle du mouvement des corps pour une excitation F_{ext} dont la fréquence augmente avec le temps.

Sur la figure 4, les points rouges illustrent les points de passage présents dans le fichier de résultats partiels fourni : `dirdyn_positions_ref.res`. Le fichier contient les valeurs pour les deux corps selon le schéma suivant (en [m]) :

```
t1_0    q1_0    t2_0    q2_0
t1_1    q1_1    t2_1    q2_1
t1_2    q1_2    t2_2    q2_2
...
t1_n    q1_n    t2_n    q2_n
...
t1_f    q1_f    t2_f    q2_f
```

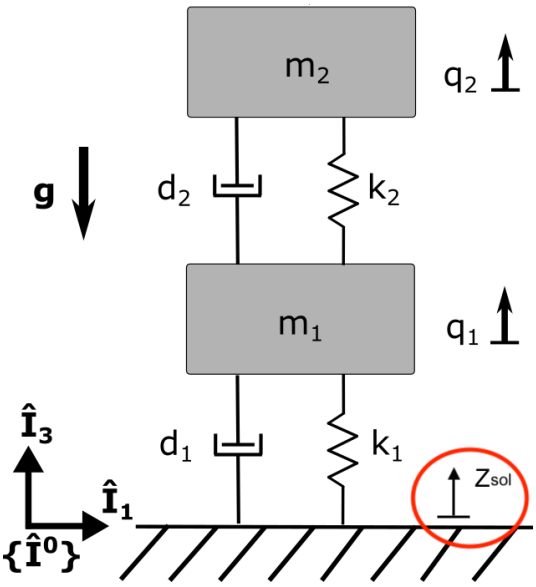

FIGURE 5 – Excitation du système par déplacement du sol.

Pour la troisième et dernière étape de cette mission, vous analyserez le comportement du système dans le cas où celui-ci n'est plus excité via l'utilisation d'une force externe sur la masse m_2 mais bien via un déplacement du sol comme illustré à la Fig. 5.

Le sol suit la loi sinusoïdale donnée dans l'énoncé de la mission 1 pour $z_{max} = 0.1\ m$, $f_0 = 0.5\ Hz$, $f_1 = 5.0\ Hz$ et avec t_0 et t_1 inchangés.

Cette excitation peut facilement se faire en modifiant l'équation de la loi de comportement du ressort-amortisseur inférieur :

$$F = k\,(z - z0 + z_{sol}) + d\,\dot{z}$$

Notez que l'excitation du système par déplacement du sol nécessitera quelques modifications dans votre code (fonctions `sweep` and `compute_derivatives`).

Pour ce travail, vous veillerez à respecter les contraintes suivantes :

- Votre script sera écrit en langage Python.
- Votre script se basera sur le fichier `two_masses.py` fourni et respectera l'interface qui y est définie.
- Vous utiliserez la fonction d'intégration *solve_ivp* de la librairie scipy.
- Les résultats de vos intégrations numériques (pour la mise à l'équilibre et pour la dynamique directe avec excitation) seront enregistrés dans trois fichiers texte avec au moins huit chiffres significatifs au format suivant (avec q les positions, qd les vitesses et qdd les accélérations, en [m]) [4].

```
t_0     q1_0     q2_0
t_1     q1_1     q2_1
t_2     q1_2     q2_2
...
t_n     q1_n     q2_n
...
t_f     q1_f     q2_f
```

```
t_0     qd1_0     qd2_0
t_1     qd1_1     qd2_1
t_2     qd1_2     qd2_2
...
t_n     qd1_n     qd2_n
...
t_f     qd1_f     qd2_f
```

```
t_0     qdd1_0     qdd2_0
t_1     qdd1_1     qdd2_1
t_2     qdd1_2     qdd2_2
...
t_n     qdd1_n     qdd2_n
...
t_f     qdd1_f     qdd2_f
```

- Le fichier `two_masses.zip` disponible sur Moodle contient le modèle du fichier `two_masses.py`, dans lequel la fonction `main`, qui permet de lancer la simulation, est définie. Le fichier `zip` reprend en plus le fichier `dirdyn_positions_ref.res`, qui contient les coordonnées des

4. Avec comme noms de fichiers *dirdyn_q.res*, *dirdyn_qd.res* et *dirdyn_qdd.res* (resp. *equil_q.res*, *equil_qd.res*, *equil_qdd.res*) pour les positions, vitesses et accélérations lors de la dynamique avec excitation (resp. de la mise à l'équilibre).

points rouges tracés sur la figure 4.

- Une fois votre programme fonctionnel, votre fichier `two_masses.py` sera encodé sur la plateforme INGInious qui se chargera de réaliser une série de tests automatiques sur votre code (exécution, validation de la solution, analyse de l'utilisation mémoire, etc.). Voyez les indications pratiques ci-dessous. **Attention vous ne mettrez sur INGInious que votre code incluant une excitation du système via la force externe**.
- En bonus, vous ferez varier quelques paramètres de simulation et répondrez à la question posée sur INGInious.

3.1 Quelques instructions pour utiliser INGInious

Le code écrit dans le cadre de cette mission est à soumettre sur la plateforme *INGInious* en suivant les étapes décrites ci-dessous.

- Connectez-vous à l'adresse `https://inginious.info.ucl.ac.be` .
- Identifiez-vous en utilisant l'option UCL (et pas INGI).
- Utilisez votre identifiant global UCL comme identifiant sur la plateforme INGInious.
- Inscrivez-vous au cours `[LEPL1504] Projet 4 en mécanique`.
- Inscrivez-vous dans votre sous-groupe dont le numéro vous aura été précisé par l'équipe enseignante.
- Accédez à la `Mission 1: Modélisation d'une suspension pneumatique`.
- Copiez/collez le code de vos fichiers `susp_pneuma.c` et `main.c` dans les zones de texte appropriées.
- En option, répondez aux deux questions bonus.
- Cliquez sur `Soumettre` pour envoyer la solution au serveur et obtenir la correction de votre exercice.

Notez qu'il n'y a pas de limite sur le nombre de soumissions. Si votre code ne produit pas le bon résultat, vous pouvez ré-essayer.