Sorbonne University
Master 2 SAR


# Interface Haptique FALCON

December 23, 2023

Compte Rendu

Par
Zhichen LU 21117174
M2 SAR

# 1 Partie I : découverte du Novint Falcon

Le espace de travail du Novint Falcon est tridimensionnel, comprenant les trois axes x, y et z.

Tout d'abord, on réalise un modèle virtuel de ressort-amortisseur. Ce modèle standard peut être exprimé par Equation 1 :

$$F = -kx - cv \tag{1}$$

F représente la force totale générée par le ressort et l'amortisseur. k est la constante de raideur du ressort. x est l'allongement ou la compression du ressort. c est le coefficient d'amortissement de l'amortisseur. v est la vitesse de la masse.

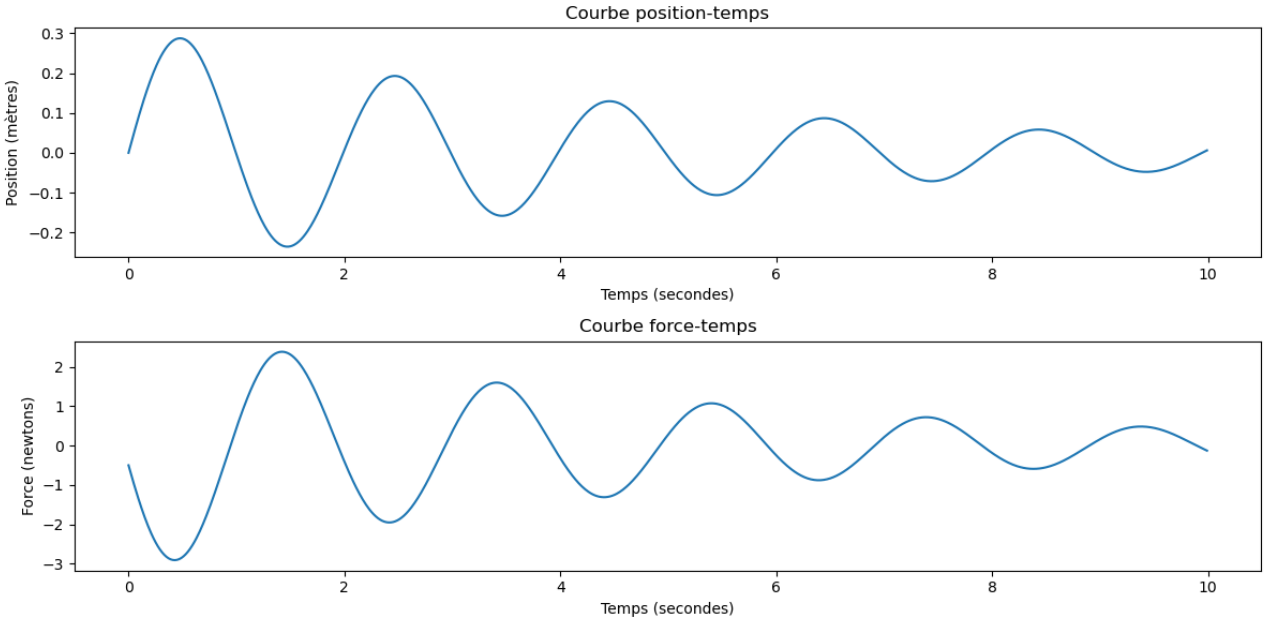



**Figure 1:** Modèle ressort-amortissement

Pour choisir un coefficient d'amortissement approprié, il est d'abord nécessaire de déterminer l'objectif principal du système. Si le but de l'amortisseur est de réduire les vibrations, un coefficient d'amortissement plus élevé sera nécessaire. Si l'objectif est de maintenir une certaine réponse élastique, un coefficient d'amortissement plus faible peut être requis. Idéalement, la valeur d'amortissement que nous choisissons est généralement proche de la valeur d'amortissement critique. L'amortissement critique est le niveau d'amortissement qui permet au système de retourner à l'état d'équilibre dans le plus court délai possible sans oscillation. Cette valeur peut être calculée en fonction de la masse du système et de la constante de raideur du ressort (Equation 2):

$$c_{\text{critical}} = 2\sqrt{km} \tag{2}$$

$c_{\text{critical}}$ est le coefficient d'amortissement critique. k est la constante de raideur du ressort. m est la masse du système.

Ensuite pour déterminer la masse et l'amortissement apparents du Falcon, on fait à une identification dynamique par oscillation libre.

$$m\ddot{x} + c\dot{x} + kx = 0 \Rightarrow \ddot{x} + 2\alpha\omega\dot{x} + \omega^2 x = 0 \tag{3}$$

$$T = \frac{1}{F}, \quad 2\pi T = \frac{\omega}{\sqrt{(1-\alpha^2)}}, \quad \omega^2 = \frac{k}{m}, \quad \alpha = \frac{\ln \frac{x_0}{x_1}}{2\pi} \tag{4}$$

Given the period $T$ and the damping ratio $\alpha$, the mass $m$ can be determined by the equation:

$$m_{Falcon} = \frac{k}{\omega^2} \tag{5}$$

with the angular frequency $\omega$ given by:

$$\omega = \frac{2\pi F}{\sqrt{1-\alpha^2}} \tag{6}$$

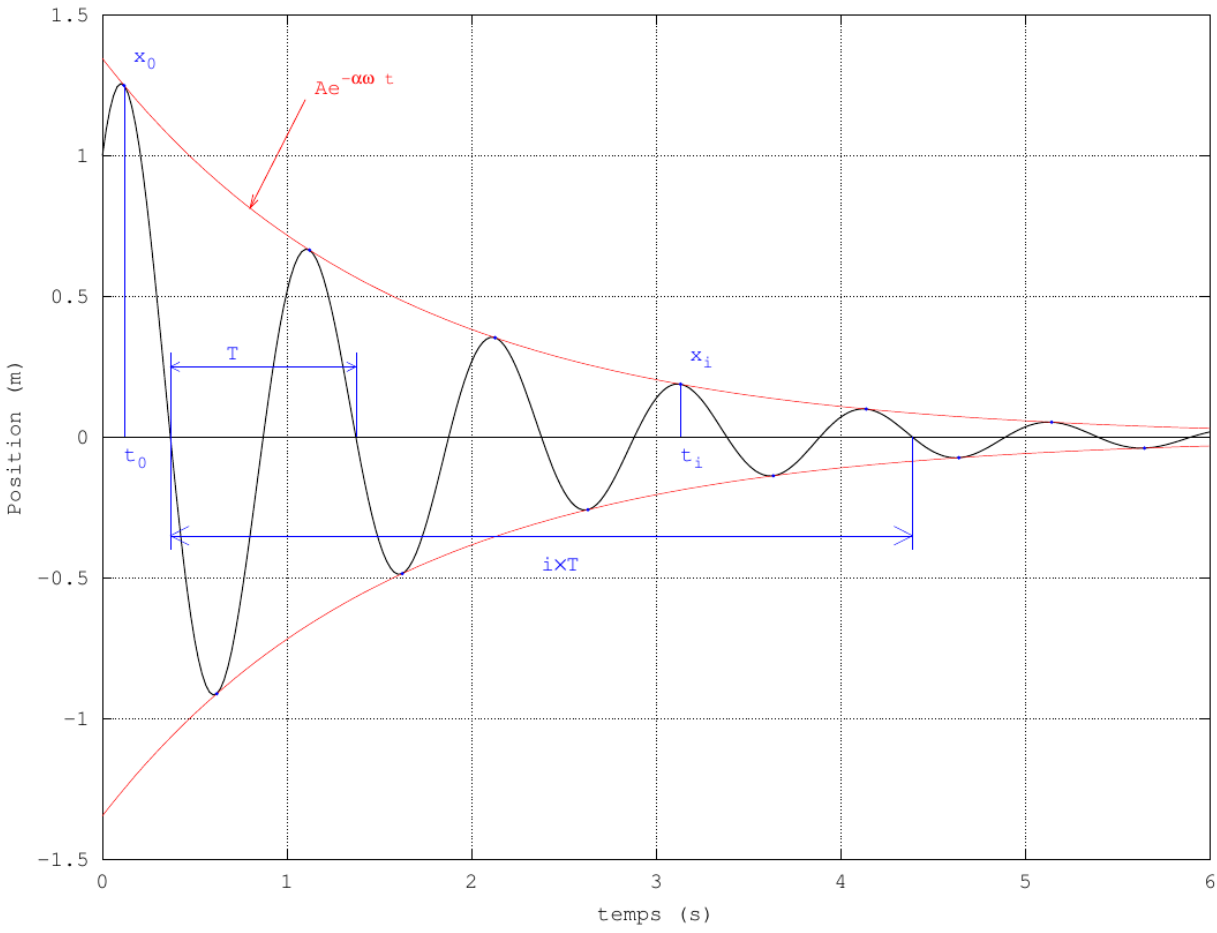



**Figure 2:** Rappel de la réponse libre d'un système masse-ressort+amortisseur.

## 1.1 Réalisation de simulations de divers espaces virtuels.

Dans le fichier de code ***functions.py***, j'ai défini neuf types différents de mise en œuvre d'espaces de mouvement virtuels comme suit :

1. **Plan:** *GeneratePlan(origin, position, vitesse, v1, v2, k, c)*
2. **Mur:** *GenerateMur(origin, position, vitesse, v1, v2, borne, k, c)*
3. **Sphère:** *GenerateSphere(origin, position, vitesse, rayon, k, c)*
4. **Cube:** *GenerateCube(origin, position, vitesse, longueur, v1, v2, k, c)*
5. **Glissière:** *GenerateGlissiere(position, vitesse, k, c)*
6. **Glissière à clic:** *GenerateGlissiereAClic(position, vitesse, omega, k, c)*
7. **Plan à clic:** *GeneratePlanAClic(position, vitesse, omega, k, c)*
8. **Pivot:***GeneratePivot(origin, position, vitesse, longueur, v1, v2, k, c)*
9. **Tube:** *GenerateTube(origin, position, vitesse, rayon, hauteur, epaisseur, v1, v2, k, c)*

Les neuf fonctions retournent la force résultante générée pour créer l'espace correspondant.

Dans le fichier ***main.fy***, j'ai appelé neuf fonctions différentes pour générer leurs propres espaces virtuels respectifs. Il est possible de choisir entre différentes appels de fonctions en utilisant des commentaires ou en retirant les commentaires.

Le fichier contient les implémentations spécifiques suivantes :

1. **Plan:** $y + 0.02 = 0$, $k = 2000$, $c = 0$. En exécutant le code correspondant,on obtient des images représentant la position en mouvement du manche dans l'espace 3D(Fig 3), ainsi que la relation entre les forces et les déplacements sur les différents axes(Fig 4).

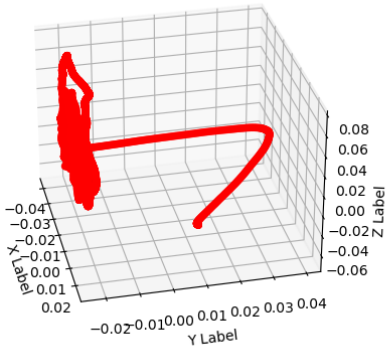
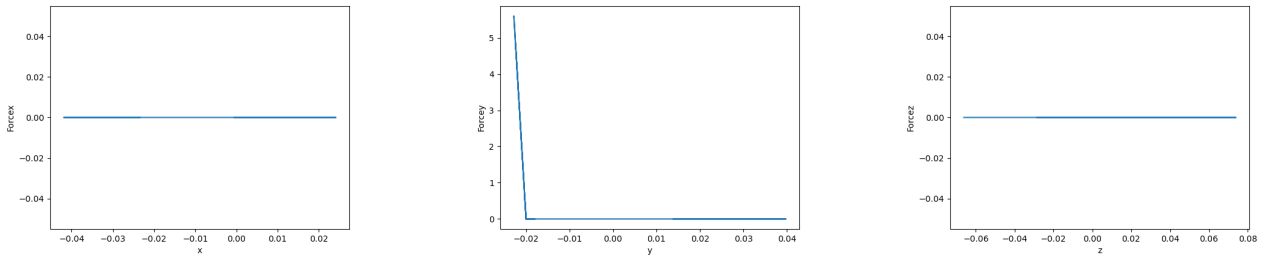



**Figure 3:** 3D-Plot plan




**(a)** Force-Position(x) **(b)** Force-Position(y) **(c)** Force-Position(z)

**Figure 4:** Force-Position

2. **Mur:** $y + 0.02 = 0$, $k = 2000$, $c = 0$. En exécutant le code correspondant,on obtient des images représentant la position en mouvement du manche dans l'espace 3D(Fig 5), ainsi que la relation entre les forces et les déplacements sur les différents axes(Fig 6).

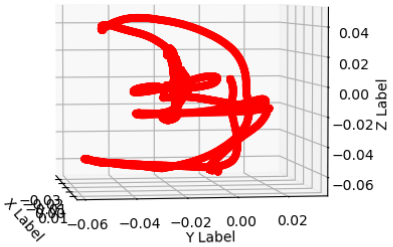
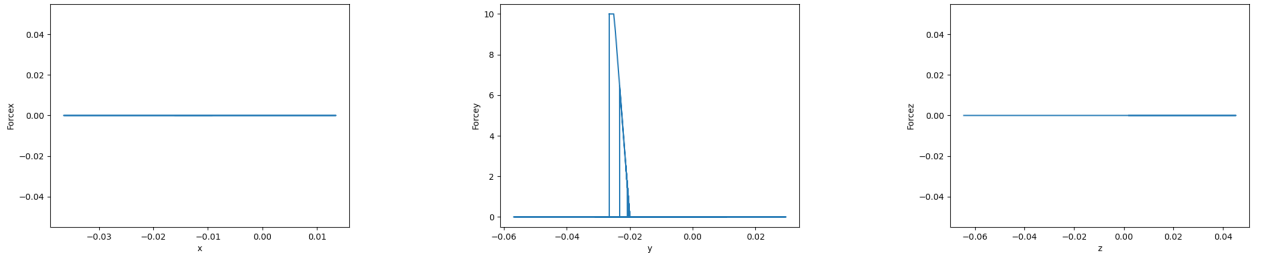



**Figure 5:** 3D-Plot Mur




**(a)** Force-Position(x) **(b)** Force-Position(y) **(c)** Force-Position(z)

**Figure 6:** Force-Position

3. **Sphère:** Centre de la sphère = $(0,0,0)$, rayon = 0.03, $k = 2000$, $c = 0$. En exécutant le code correspondant,on obtient des images représentant la position en mouvement du manche dans l'espace 3D(Fig 7), ainsi que la relation entre les forces et les déplacements sur les différents axes(Fig 8).

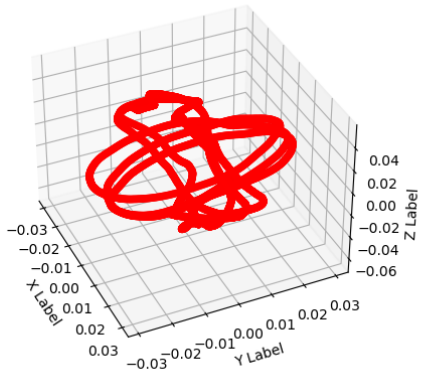
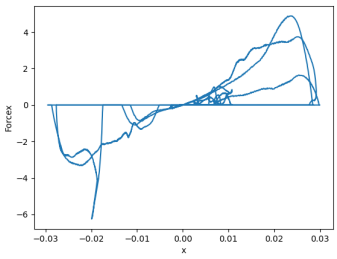
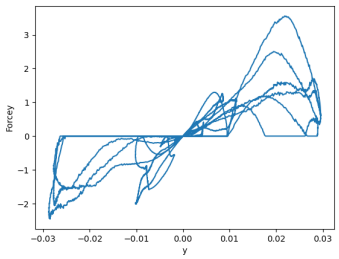


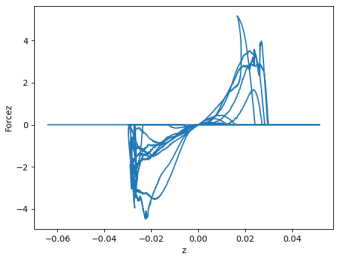

**Figure 7:** 3D-Plot Sphère




**(a)** Force-Position(x)




**(b)** Force-Position(y)




**(c)** Force-Position(z)

**Figure 8:** Force-Position

4. **Cube:** La longueur des arêtes est de 0.06 et les vecteurs prédéfinis sont $v1 = (1,0,0)$, $v2 = (0,0,1)$, $k = 2000$, $c = 0$. En exécutant le code correspondant,on obtient des images représentant la position en mouvement du manche dans l'espace 3D(Fig 9), ainsi que la relation entre les forces et les déplacements sur les différents axes(Fig 10).

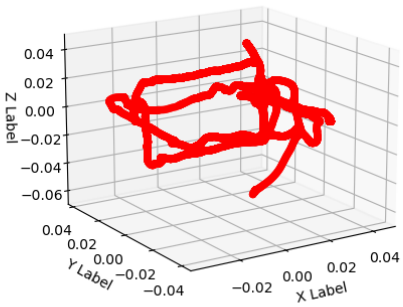
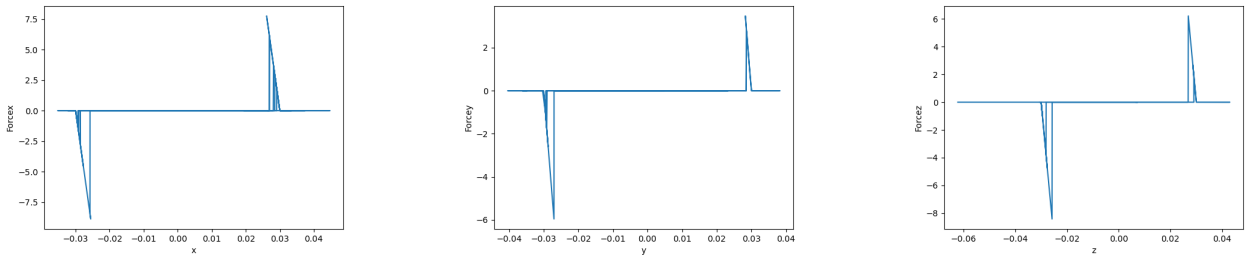



**Figure 9:** 3D-Plot Cube




**(a)** Force-Position(x) **(b)** Force-Position(y) **(c)** Force-Position(z)

**Figure 10:** Force-Position

5. **Glissière:** Sur l'axe x, $k = 2000$, $c = 0$. En exécutant le code correspondant,on obtient des images représentant la position en mouvement du manche dans l'espace 3D(Fig 11), ainsi que la relation entre les forces et les déplacements sur les différents axes(Fig 12).

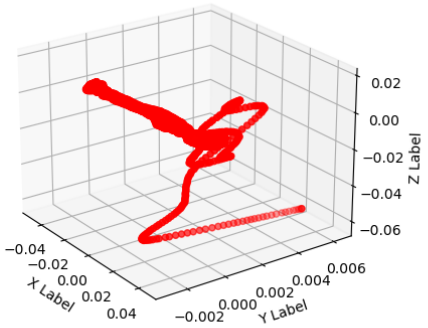
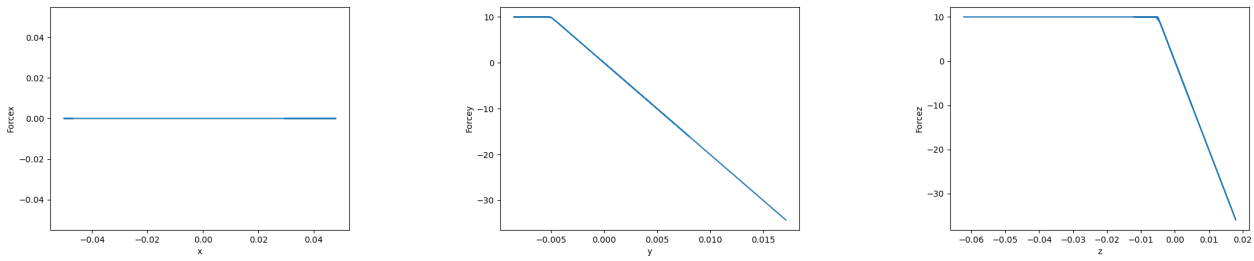



**Figure 11:** 3D-Plot Glissière




**(a)** Force-Position(x) **(b)** Force-Position(y) **(c)** Force-Position(z)

**Figure 12:** Force-Position

6. **Glissière à clic:** Sur l'axe x, $\omega = 2100$, $k = 2000$, $c = 0$. En exécutant le code correspondant,on obtient des images représentant la position en mouvement du manche dans l'espace 3D(Fig 13), ainsi que la relation entre les forces et les déplacements sur les différents axes(Fig 14).

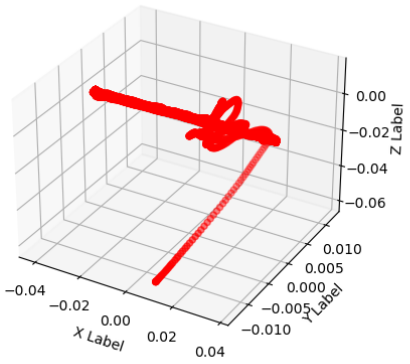
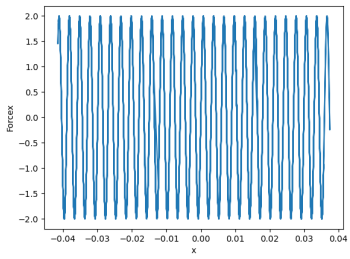
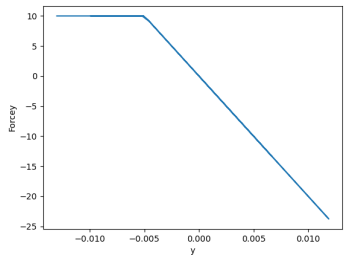


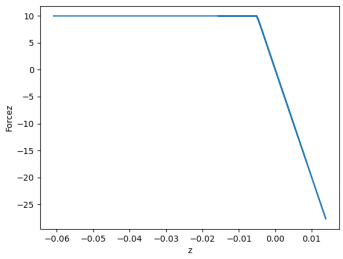

**Figure 13:** 3D-Plot Glissière à clic




**(a)** Force-Position(x)




**(b)** Force-Position(y)




**(c)** Force-Position(z)

**Figure 14:** Force-Position

7. **Plan à clic:** Sur le plan $xy$, $\omega = 2100$, $k = 2000$, $c = 0$. En exécutant le code correspondant,on obtient des images représentant la position en mouvement du manche dans l'espace 3D(Fig 15), ainsi que la relation entre les forces et les déplacements sur les différents axes(Fig 16).

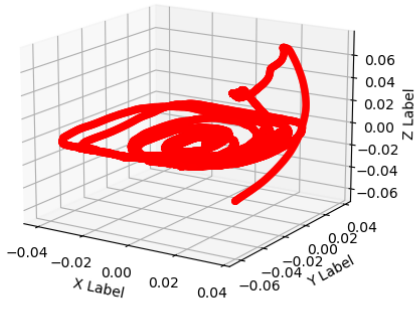
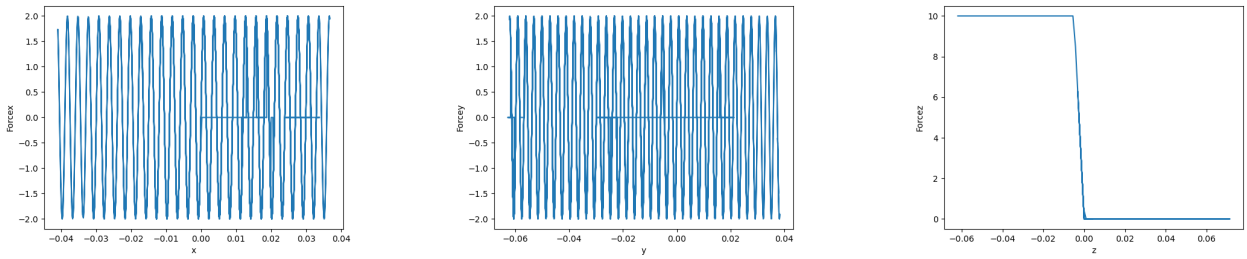



**Figure 15:** 3D-Plot Plan à clic




**(a)** Force-Position(x) **(b)** Force-Position(y) **(c)** Force-Position(z)

**Figure 16:** Force-Position

8. **Pivot:** Point fixé = $(0,0,1)$, longueur de pivot = 0.03, les vecteurs prédéfinis sont $v1 = (1,0,0)$, $v2 = (0,1,0)$, $k = 2000$. En exécutant le code correspondant,on obtient des images représentant la position en mouvement du manche dans l'espace 3D(Fig 17), ainsi que la relation entre les forces et les déplacements sur les différents axes(Fig 18).

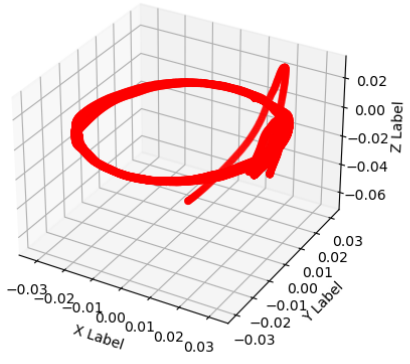
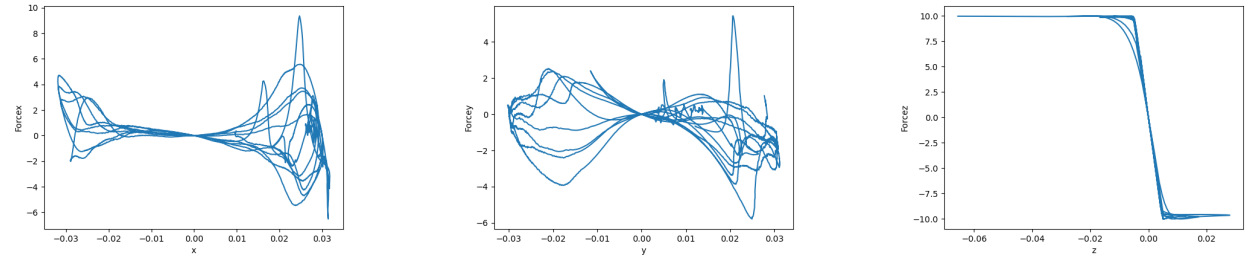



**Figure 17:** 3D-Plot pivot




**(a)** Force-Position(x) **(b)** Force-Position(y) **(c)** Force-Position(z)

**Figure 18:** Force-Position

9. **Tube:** Position du centre du cercle sur la base du cylindre = $(-0.01,0,0)$, rayon = 0.04, hauteur = 0.04, épaisseur = 0.02, les vecteurs prédéfinis sont $v1 = (0,1,0)$, $v2 = (0,0,1)$, $k = 2000$, $c = 0$. En exécutant le code correspondant,on obtient des images représentant la position en mouvement du manche dans l'espace 3D(Fig 19), ainsi que la relation entre les forces et les déplacements sur les différents axes(Fig 20).

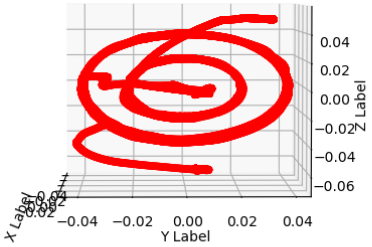
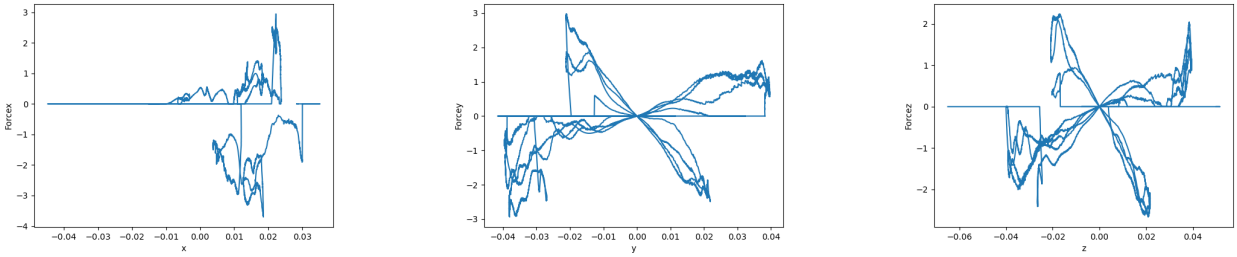



**Figure 19:** 3D-Plot plan

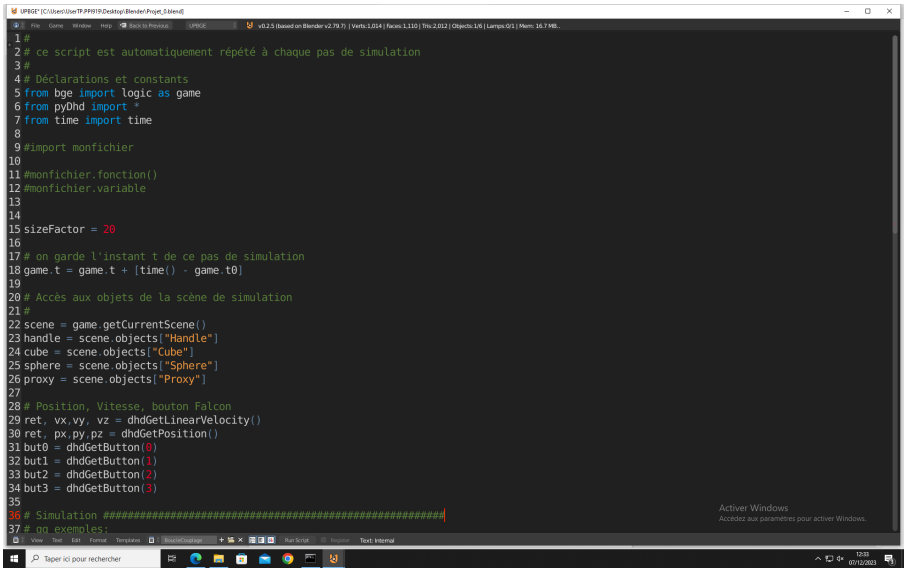



**(a)** Force-Position(x) **(b)** Force-Position(y) **(c)** Force-Position(z)

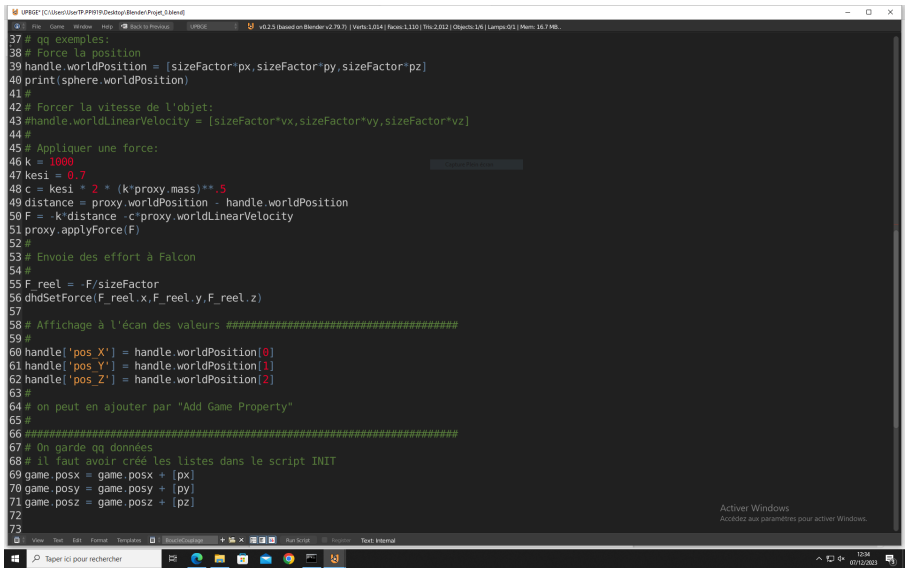
**Figure 20:** Force-Position

# 2 Partie II : Couplage à un simulateur physique

Pour exécuter le contenu du code dans le logiciel Blender comme indiqué dans les figures **????** ci-dessous :



```
#
# ce script est automatiquement répété à chaque pas de simulation
#
# Déclarations et constants
from bge import logic as game
from pyDhd import *
from time import time

#import monfichier

#monfichier.fonction()
#monfichier.variable


sizeFactor = 20

# on garde l'instant t de ce pas de simulation
game.t = game.t + [time() - game.t0]

# Accès aux objets de la scène de simulation
#
scene = game.getCurrentScene()
handle = scene.objects["Handle"]
cube = scene.objects["Cube"]
sphere = scene.objects["Sphere"]
proxy = scene.objects["Proxy"]

# Position, Vitesse, bouton Falcon
ret, vx,vy, vz = dhdGetLinearVelocity()
ret, px,py,pz = dhdGetPosition()
but0 = dhdGetButton(0)
but1 = dhdGetButton(1)
but2 = dhdGetButton(2)
but3 = dhdGetButton(3)

# Simulation ##########################################################
# qq exemples:
```



```
# qq exemples:
# Force la position
handle.worldPosition = [sizeFactor*px,sizeFactor*py,sizeFactor*pz]
print(sphere.worldPosition)
#
# Forcer la vitesse de l'objet:
#handle.worldLinearVelocity = [sizeFactor*vx,sizeFactor*vy,sizeFactor*vz]
#
# Appliquer une force:
k = 1000
kesi = 0.7
c = kesi * 2 * (k*proxy.mass)**.5
distance = proxy.worldPosition - handle.worldPosition
F = -k*distance -c*proxy.worldLinearVelocity
proxy.applyForce(F)
#
# Envoie des effort à Falcon
#
F_reel = -F/sizeFactor
dhdSetForce(F_reel.x,F_reel.y,F_reel.z)

# Affichage à l'écan des valeurs ###################################
#
# On garde qq données
# il faut avoir créé les listes dans le script INIT
game.posx = game.posx + [px]
game.posy = game.posy + [py]
game.posz = game.posz + [pz]
```

En particulier, le ***proxy*** est défini comme un corps rigide et sa masse est modifiée à 0,5 $kg$. (Fig 21)

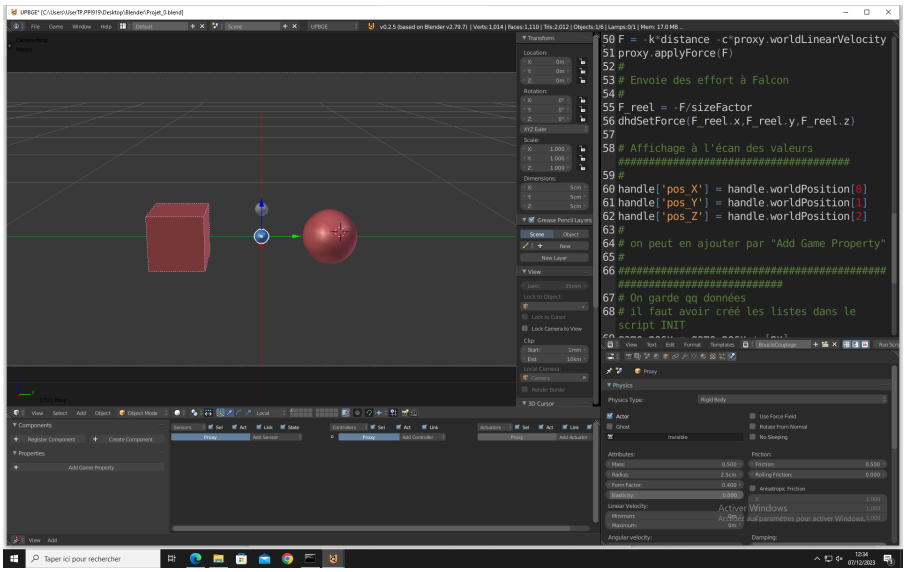


**Figure 21**

En comparant les vidéos $k = 1000\_kesi = 0.7\_f = 200.mp4$ et $k = 1000\_kesi = 0.7\_f = 1000.mp4$, dans les opérations pratiques, il est clairement perceptible que plus la fréquence de mise à jour du simulateur est élevée, plus la manipulation de la manette donne une sensation de "lourdeur" et offre une expérience de "contact réel" plus prononcée. En revanche, à des fréquences plus basses, la manipulation de la manette semble plus "facile" et moins immersive.

La sensation de "lourdeur" ou de "réalisme" accrue lorsque la fréquence du simulateur est élevée est due à plusieurs facteurs :

1. Rendu en temps réel : Une fréquence de mise à jour élevée permet au simulateur de calculer et de rendre les interactions entre les objets avec plus de précision et de réactivité. Cela signifie que chaque petit mouvement ou contact est mieux pris en compte, ce qui donne une sensation de réalisme accru.

2. Fidélité de la réponse haptique : À des fréquences plus élevées, le simulateur peut fournir des réponses haptiques plus détaillées et réalistes, ce qui contribue à créer une sensation de "lourdeur" en simulant la résistance des objets.

3. Meilleure synchronisation : Une fréquence élevée permet une meilleure synchronisation entre l'entrée de la manette et la réponse du simulateur, créant ainsi une expérience plus cohérente et immersive.

En comparant les vidéos $k = 1000\_kesi = 0.2\_Static.mp4$ et $k = 1000\_kesi = 0.7\_Static.mp4$ ou $k = 1500\_kesi = 0.7\_Static.mp4$ et $k = 1500\_kesi = 0.9\_Static.mp4$, plusieurs observations peuvent être faites :

**la constante de raideur du ressort "$k$":**

1. Impact sur l'amplitude : La constante de raideur du ressort "k" influence l'amplitude. Une valeur élevée de "k" conduit à une amplitude plus petite, tandis qu'une valeur plus faible de "k" entraîne une amplitude plus grande.

2. Stabilité du système : Une valeur plus élevée de "k" peut améliorer la stabilité du système car elle augmente la force de rappel. Cependant, si la valeur de "k" est excessive, le système peut devenir instable car il peut générer des forces supérieures à ce que le système peut contrôler.

**Coefficient d'amortissement $\xi$:**

1. Réduction des vibrations : Un coefficient d'amortissement plus élevé réduit les vibrations de l'objet, offrant ainsi une meilleure stabilité et une plus grande fluidité.

2. Ralentissement de la réponse de l'objet : Un coefficient d'amortissement plus élevé ralentit la réponse de l'objet aux forces externes ou aux mouvements. Cela se traduit par un mouvement plus régulier de l'objet, avec moins d'accélérations et de décélérations rapides.

3. Impact sur la transmission des forces : Le coefficient d'amortissement peut influencer la transmission des forces. Un coefficient d'amortissement plus élevé fait décroître plus rapidement la force, réduisant ainsi l'impact entre les objets et rendant l'interaction plus douce.

4. Stabilité : Dans certaines situations, un amortissement approprié peut améliorer la stabilité du système. Des coefficients d'amortissement trop élevés ou trop faibles peuvent rendre le système instable ou moins réactif.