Modélisation et analyse de la sustentation électromagnétique d'un train Maglev sur une voie flexible soumise à des vibrations

Khaldoun Yassine

24746

Session 2025

INTRODUCTION

Train Maglev – Japon

Train Maglev – Chine (Shanghai)

Problématique

Comment modéliser et maîtriser l'influence des vibrations de la voie sur la qualité de la sustentation d'un train Maglev afin d'optimiser l'efficacité du système de contrôle ?

Plan

- Introduction
- Problématique
- Principe de la technologie EMS
- Étude mécanique
- Étude électrique
- Introduction du contrôleur
- Conclusion

Technologie EMS

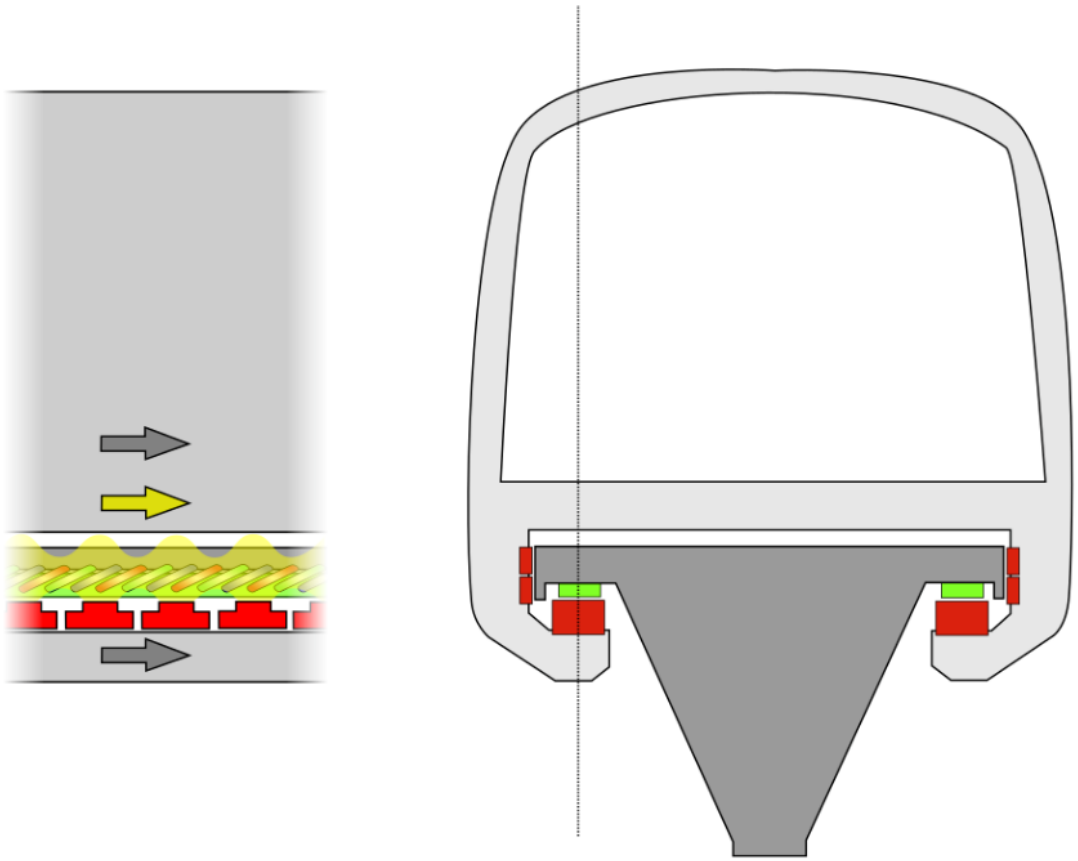
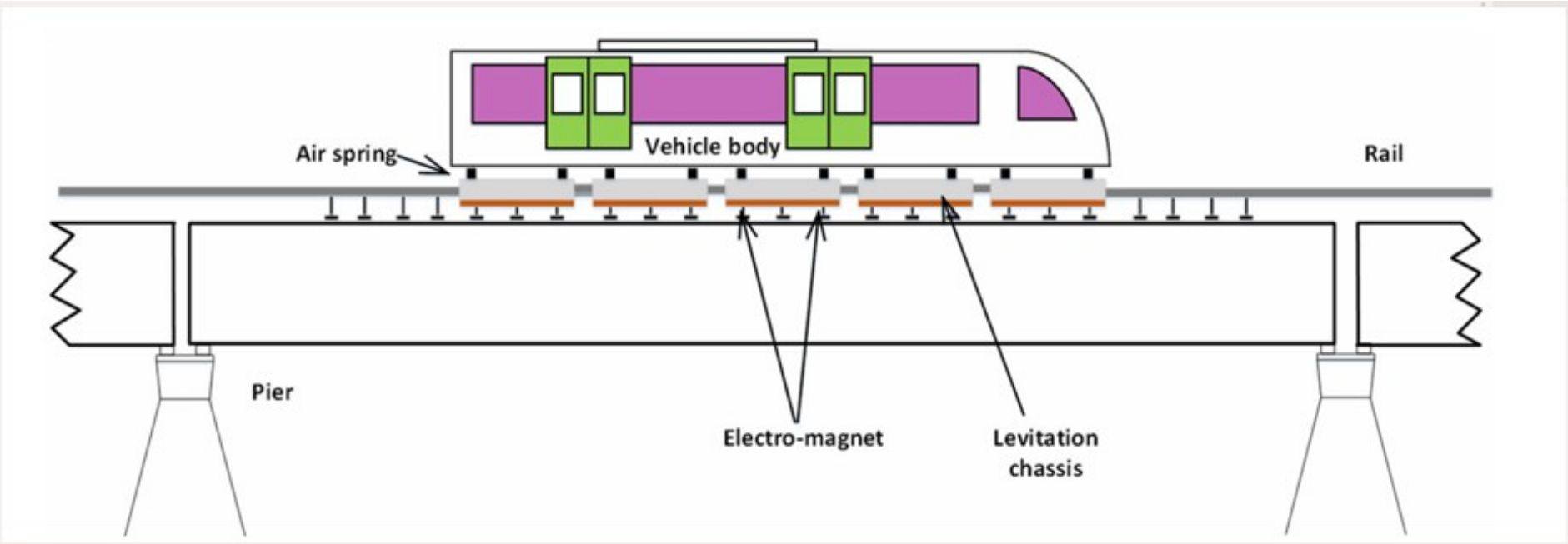


Figure: Principe de la sustentation électromagnétique (EMS)

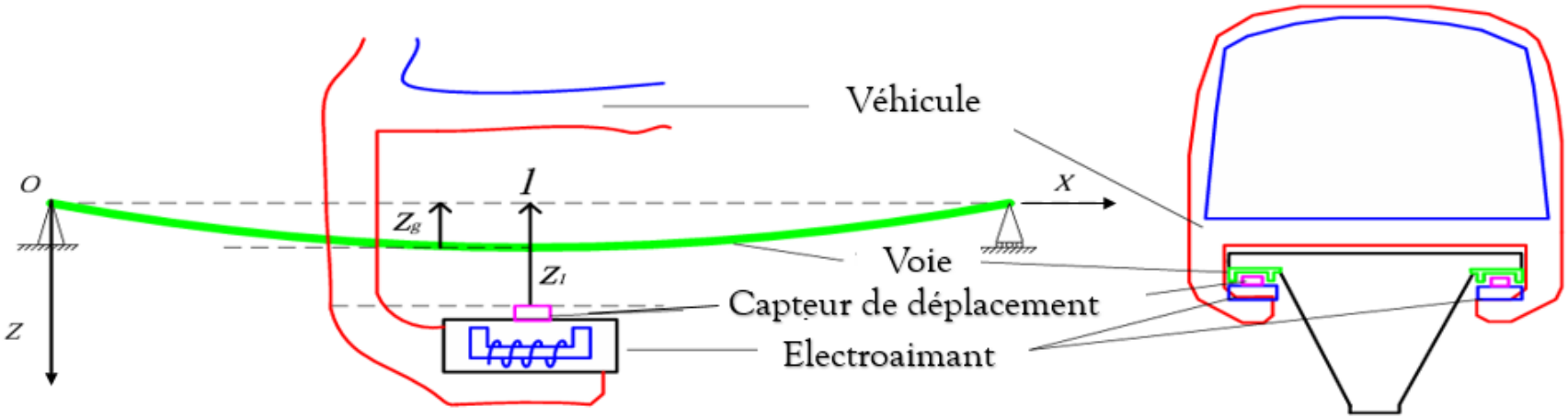
Étude mécanique

Hypothèses



- Le train est à l'arrêt et est en sustentation
- La voie est modélisée comme une poutre d'Euler-Bernoulli
- Les unités de sustentation sont entièrement découplées : modèle de sustentation à point unique
- La longueur de l'électroaimant est négligeable devant la portée de la voie

Schéma du Modèle retenu pour l'étude



Modèle du système de suspension avec la voie élastique

Equation de vibration de la poutre

Le déplacement vertical de la voie à n'importe quel point l remplit l'équation différentielle partielle de la vibration d'une poutre d'Euler–Bernoulli :

$$E_g I_g \frac{\partial^4 z_g}{\partial l^4} + \rho_g \frac{\partial^2 z_g}{\partial t^2} + b_g \frac{\partial z_g}{\partial t} = p(l, t)$$

- E_g : module d'élasticité du matériau (ou module de Young), exprimé en pascals (Pa).
- I_g : moment d'inertie de la section transversale de la voie (en m^4).
- ρ_g : masse linéique de la voie (en $\mathrm{kg/m}$).
- b_g : coefficient d'amortissement équivalent de la voie (en $\mathrm{kg \cdot m^{-1} \cdot s^{-1}}$).
- $p(l, t)$: densité de charge répartie appliquée sur la poutre, en fonction de la position l et du temps t (exprimée en $\mathrm{N/m}$).

Solution générale

- La solution générale s'écrit :

$$z_g(l,t) = \sum_{n=1}^{\infty} \varphi_n(l)\, q_n(t)$$

- $\varphi_n(l)$ représente la n-ième fonction propre correspondant à la poutre simplement supportée ;
- $q_n(t)$ représente la coordonnée généralisée correspondant à la fonction au temps t .

Approximation au premier ordre

- On se limite au premier ordre

$$z_g(l,\, t) \;=\; \varphi_1(l) \cdot q_1(t)$$

Avec :

$$\varphi_1(l) \;=\; \sqrt{\frac{2}{m_g}}\, \sin\!\left(\frac{\pi\, l}{L}\right)$$

- Finalement, on met l'équation sous la forme :

$$\boxed{\ddot{z}_g \;+\; 2\,\eta_1\,\omega_1\,\dot{z}_g \;+\; \omega_1^2\, z_g \;=\; \frac{2}{\rho_g\, l_g}\, \sin^2\!\left(\frac{\pi\, l}{L}\right) F_e}$$

Écriture du principe fondamental de la dynamique :

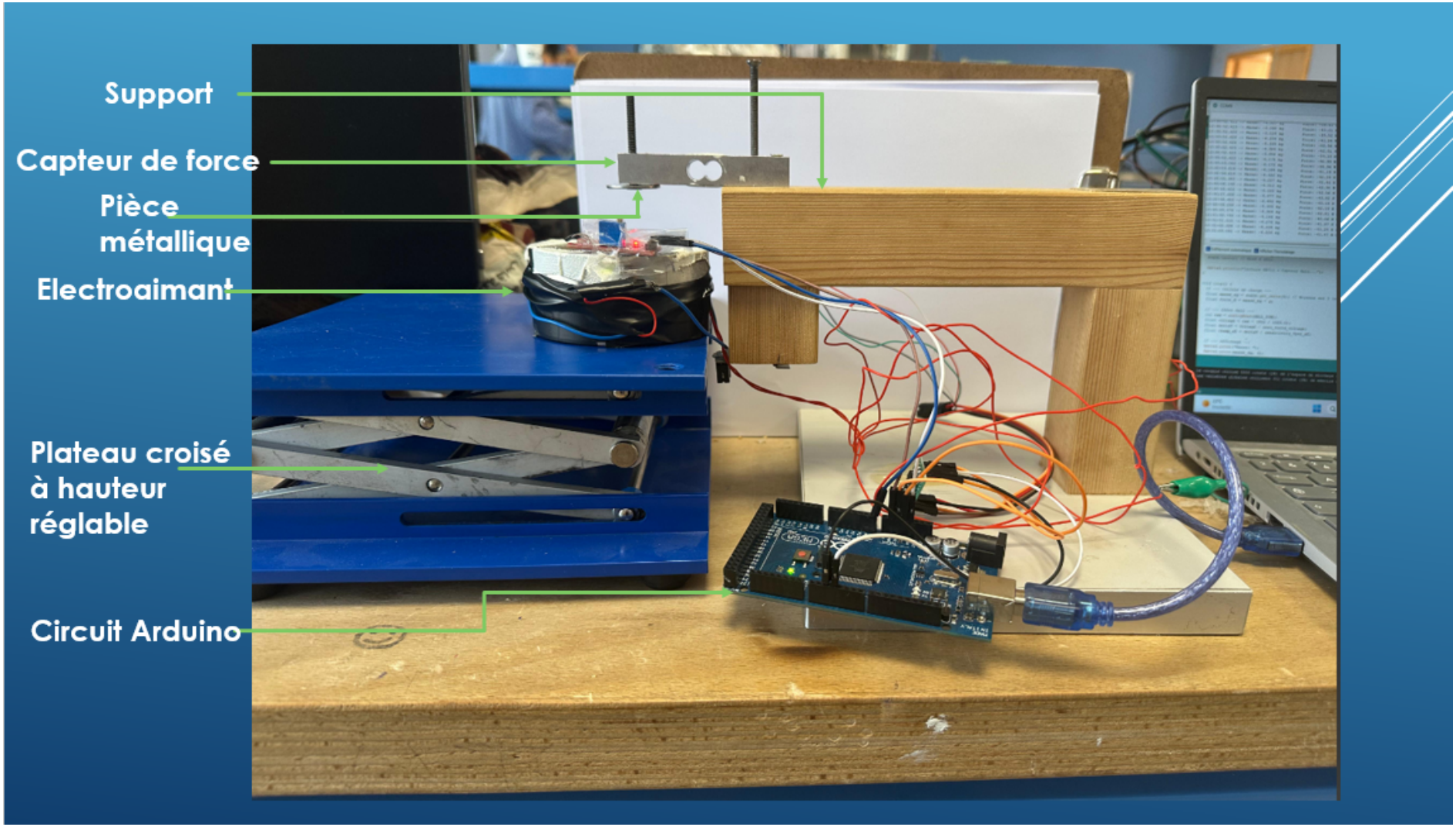
$$\boxed{(m+M)\,g \;-\; F_e \;=\; m\,\ddot{z}_1}$$

- L'intensité de la force exercée par l'électroaimant sur la poutre s'exprime par :

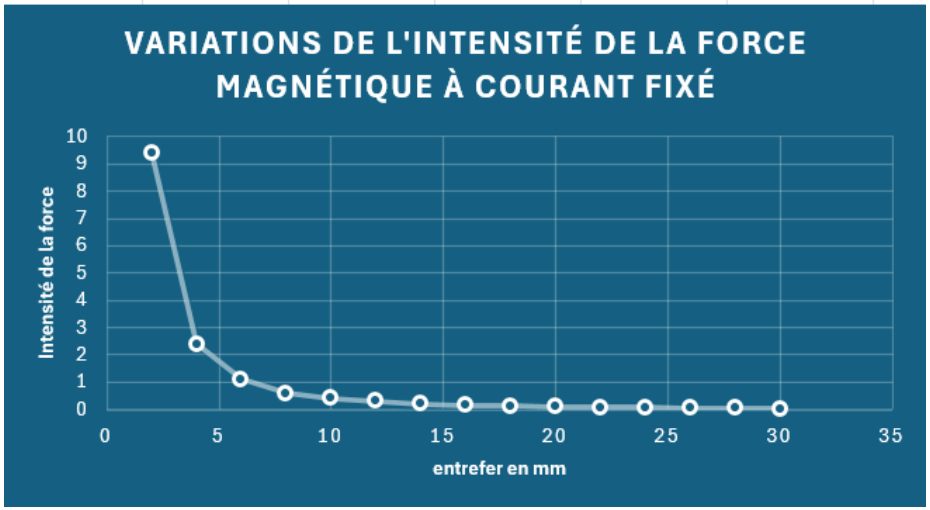
$$\boxed{F_e \;=\; \frac{\mu_0\, N^2\, A}{4}\left(\frac{i}{z_1 - z_g}\right)^2}$$

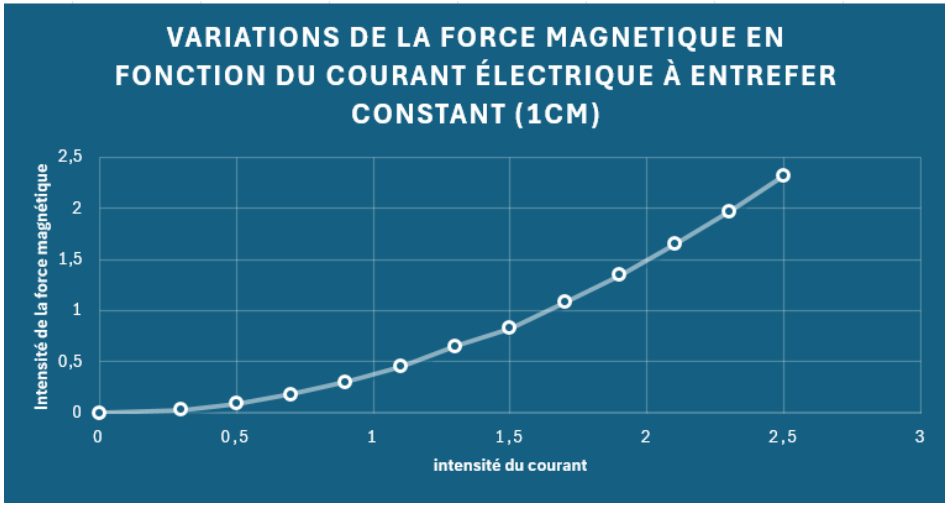
Vérification expérimentale de la loi de force

Montage expérimental



Résultats


VARIATIONS DE L'INTENSITÉ DE LA FORCE MAGNÉTIQUE À COURANT FIXÉ
Intensité de la force
10
9
8
7
6
5
4
3
2
1
0
0
5
10
15
20
25
30
35
entrefer en mm


VARIATIONS DE LA FORCE MAGNETIQUE EN FONCTION DU COURANT ÉLECTRIQUE À ENTREFER CONSTANT (1CM)
Intensité de la force magnétique
2,5
2
1,5
1
0,5
0
0
0,5
1
1,5
2
2,5
3
intensité du courant

Étude électrique

Hypothèses

- Les fuites de flux magnétique sont négligées.
- Les effets de bord sont négligés.
- Le flux magnétique est supposé uniformément réparti dans l'entrefer.

Tension de l'électroaimant

- Le flux du champ magnétique traversant l'entrefer de l'électroaimant s'écrit :

$$\Phi = \frac{\mu_0 N^2 A}{4} \frac{i}{z_1 - z_g}$$

- La tension est donc :

$$u = R\,i + \frac{d\Phi}{dt}$$

- En posant $C_1 = \frac{\mu_0 N^2 A}{4}$, on obtient :

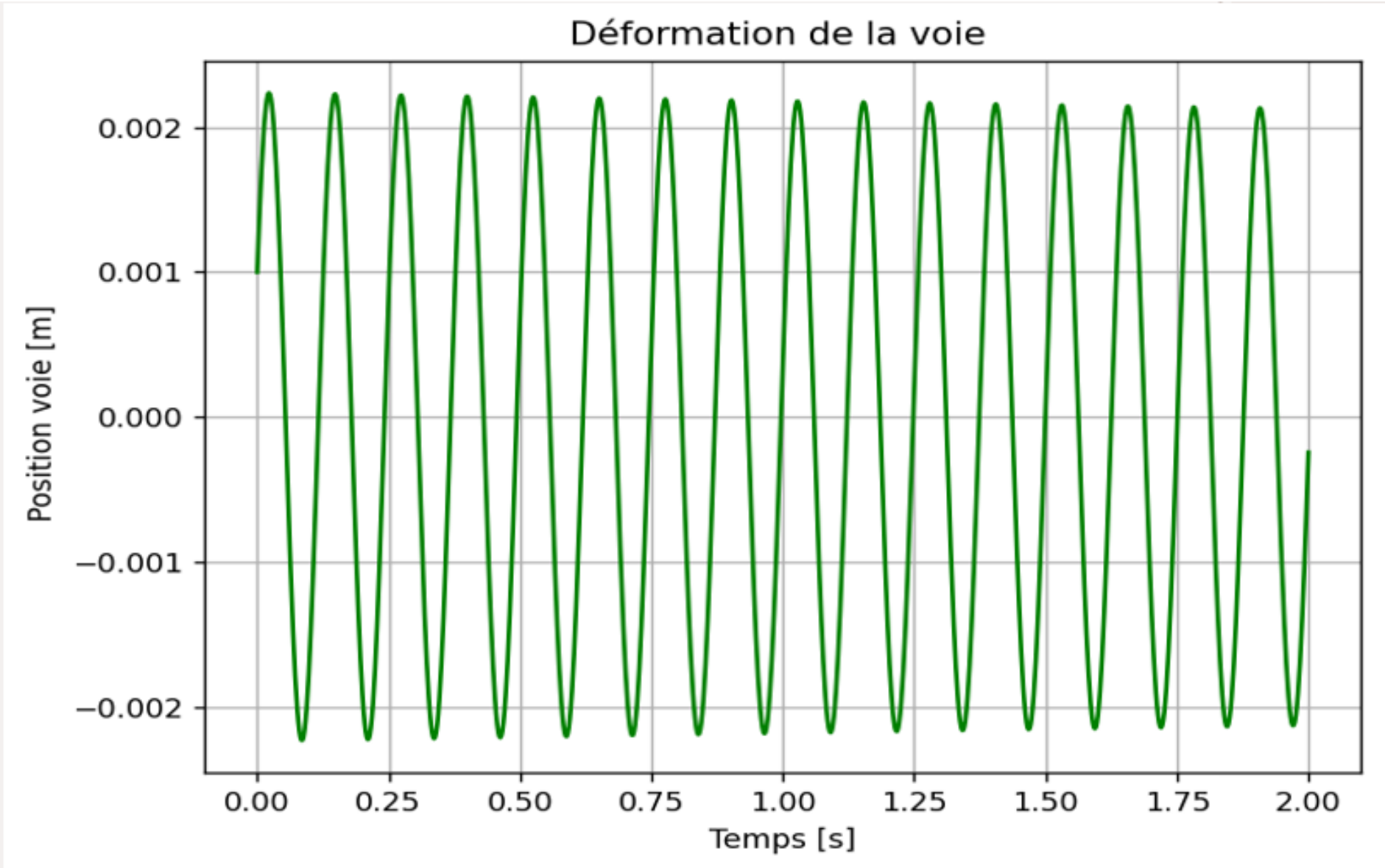
$$\boxed{u = R\,i + \frac{2\,C_1}{z_1 - z_g}\frac{di}{dt} - \frac{2\,C_1\left(\dot{z}_1 - \dot{z}_g\right)}{(z_1 - z_g)^2}\,i}$$

Bilan:

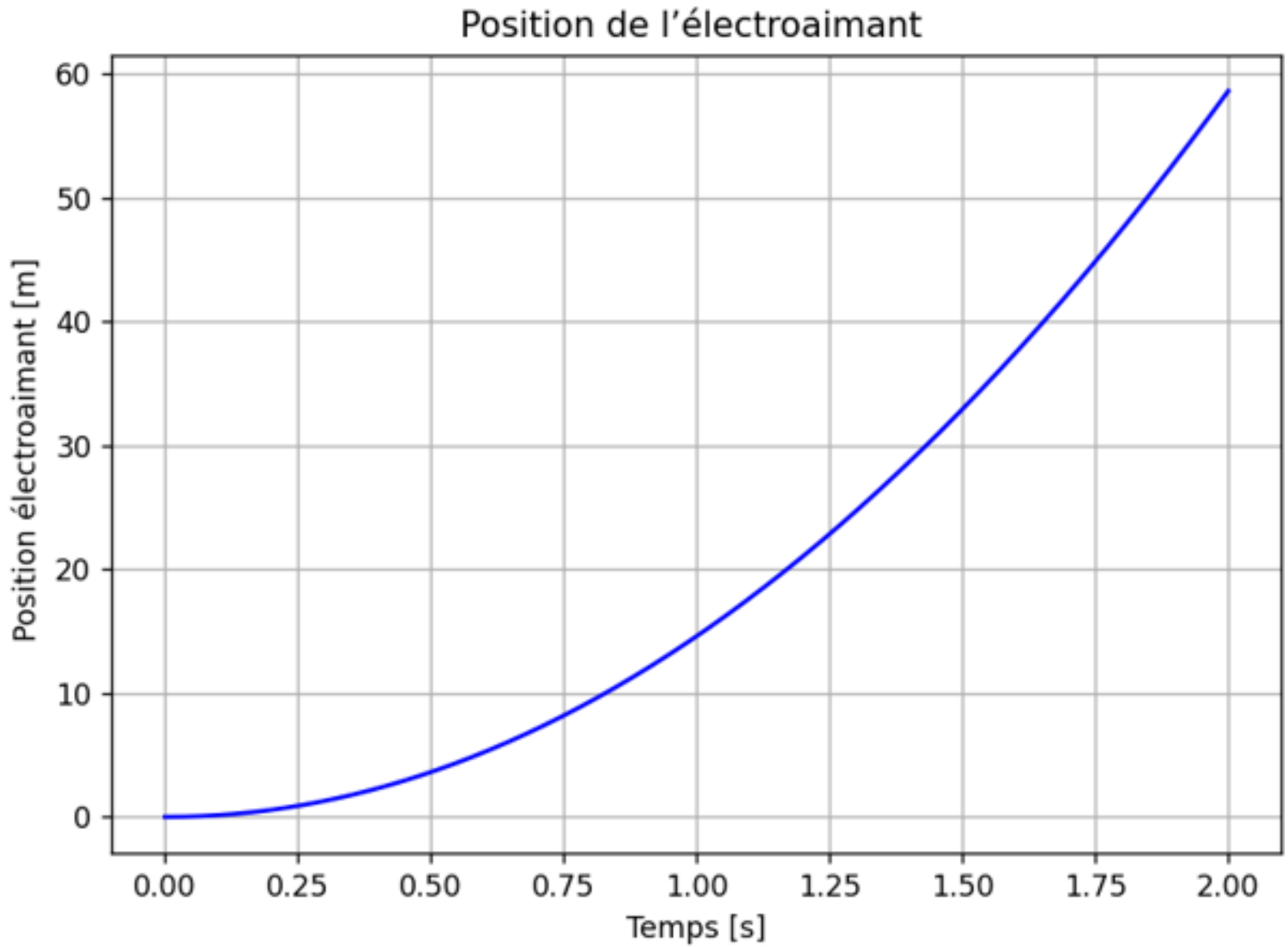
$$\begin{cases} \ddot{z}_g + 2\,\eta_1\,\omega_1\,\dot{z}_g + \omega_1^2\,z_g = C_2\left(\dfrac{i}{z_1 - z_g}\right)^2, \\ (m+M)\,g \;-\; C_1\left(\dfrac{i}{z_1 - z_g}\right)^2 = m\,\ddot{z}_1, \\ u = R\,i + \dfrac{2\,C_1}{z_1 - z_g}\,\dfrac{di}{dt} - \dfrac{2\,C_1\,i}{(z_1 - z_g)^2}\,(\dot{z}_1 - \dot{z}_g) \end{cases}$$

Réponse du système en absence de correction

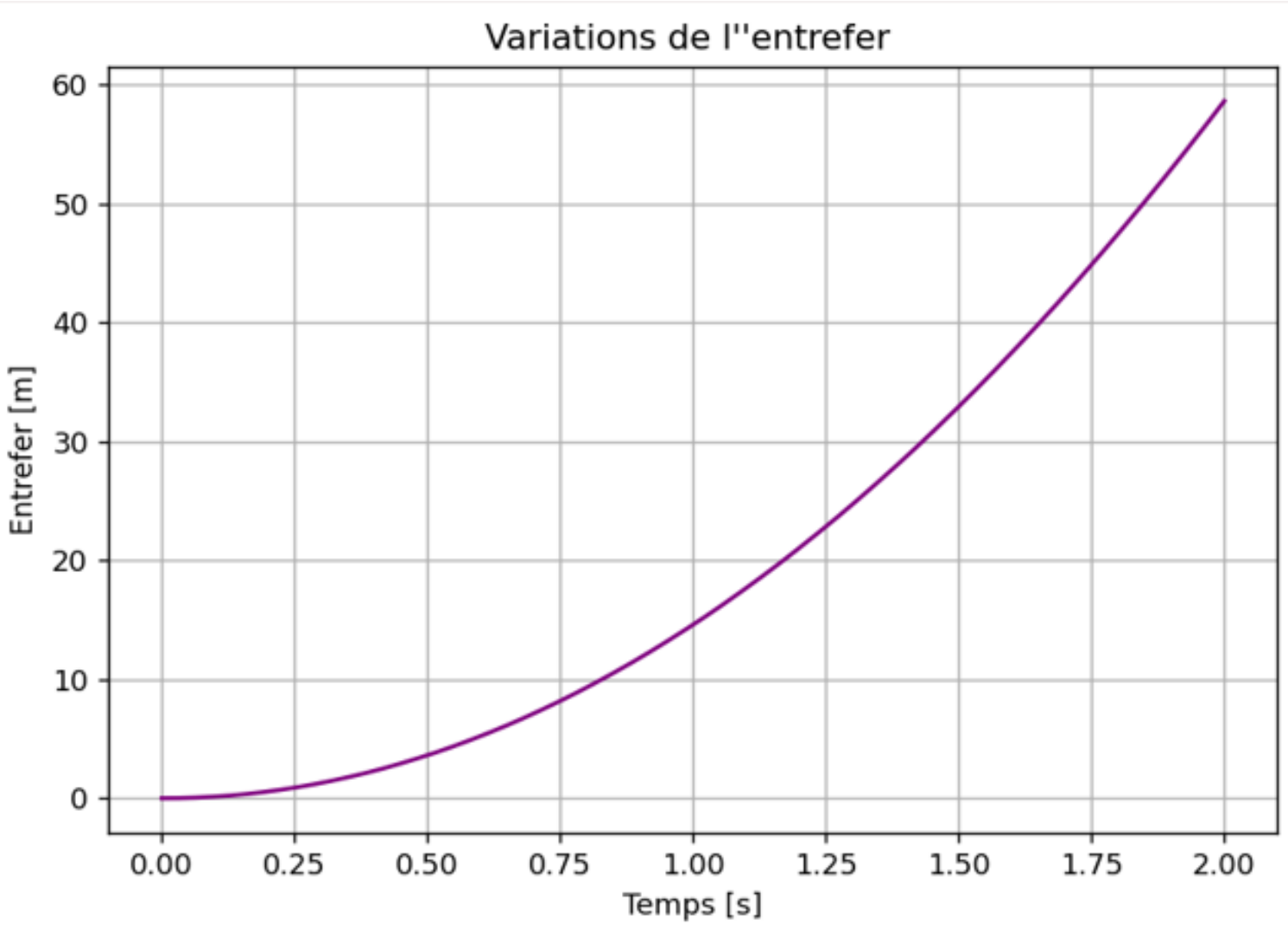
Réponse de la poutre



Réponse de l'électroaimant

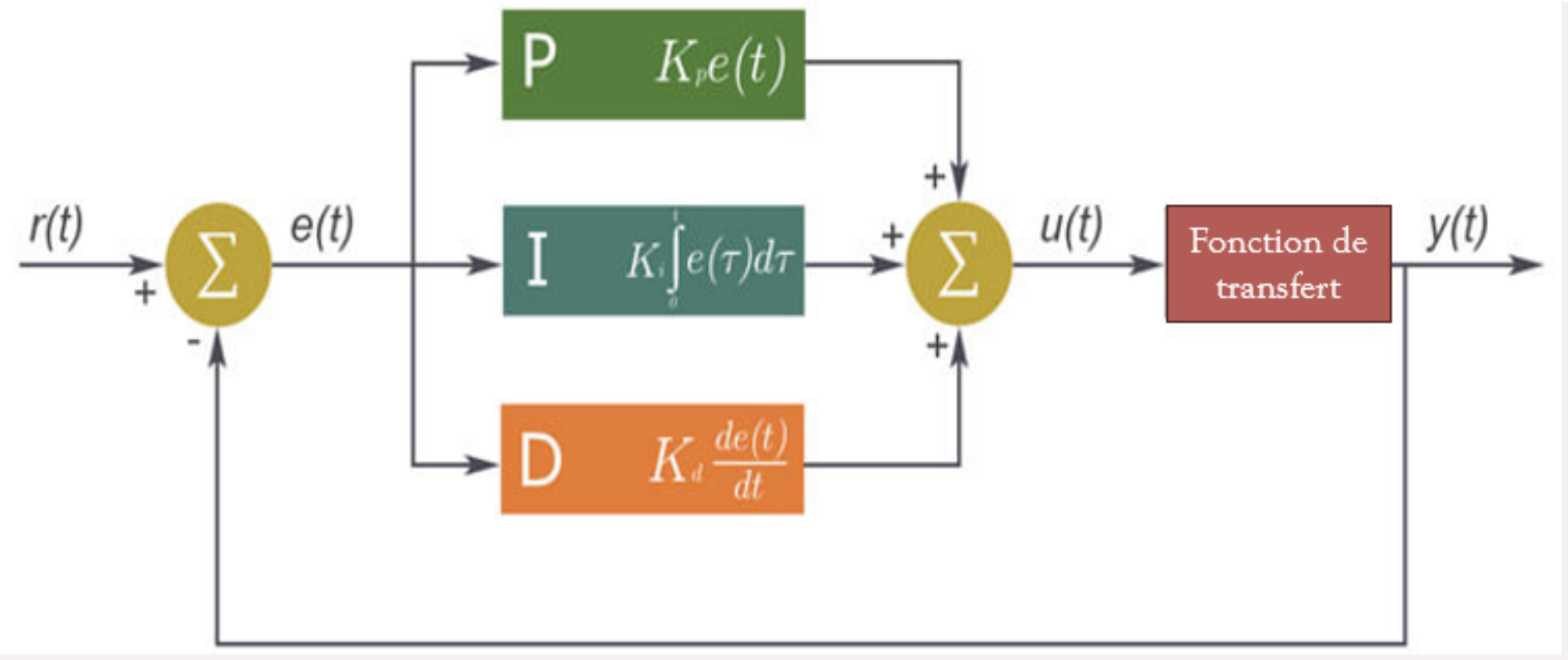


Variations de l'entrefer



Introduction du correcteur

Correcteur PID



$$u(t) = K_P\, e(t) + K_I \int_0^t e(\tau)\,\mathrm{d}\tau + K_D \frac{\mathrm{d}e(t)}{\mathrm{d}t}$$

Test du correcteur - Voie Rigide

Modèle simplifié - Voie rigide

$$\begin{cases} m\,\ddot{z}_1 = (m+M)\,g \;-\; C_1\,\dfrac{i^2}{z_1^2}\,, \\ u = R\,i \;+\; \dfrac{2\,C_1}{z_1}\,\dfrac{di}{dt} \;-\; \dfrac{2\,C_1\,i}{z_1^2}\,\dot{z}_1\,. \end{cases}$$

À l'équilibre: $\begin{cases} (m+M)\,g = C_1\,\dfrac{i_0^2}{z_0^2}\,, \\ u_0 = R\,i_0\,. \end{cases}$

Linéarisation autour du point d'équilibre $(i_0,\ z_0)$

On linéarise autour du point d'équilibre $(i_0,\ z_0)$.

$$\begin{cases} m\,\Delta\ddot{z}_1(t) = K_z\,\Delta z_1(t) \ - \ K_i\,\Delta i(t), \\ \Delta u(t) = R\,\Delta i(t) \ + \ L_0\,\Delta\dot{i}(t) \ - \ K_i\,\Delta\dot{z}_1(t)\,. \end{cases}$$

- $K_z = \left.\dfrac{\partial F}{\partial z}\right|_{(z_0,i_0)} = \dfrac{\mu_0\,N^2\,A\,i_0^2}{2\,z_0^3}$
- $K_i = \left.\dfrac{\partial F}{\partial i}\right|_{(z_0,i_0)} = \dfrac{\mu_0\,N^2\,A\,i_0}{2\,z_0^2}$
- $L_0 = \dfrac{\mu_0\,N^2\,A}{2\,z_0}$

Fonction de transfert

Par application de la transformée de Laplace au modèle linéarisé, la fonction de transfert du système s'écrit :

$$G(s) = \frac{\Delta Z(s)}{\Delta U(s)} = -\frac{k_i}{m\,L_0}\,\frac{1}{s^3 + \frac{R}{L_0}\,s^2 - \frac{k_z\,R}{m\,L_0}}\,.$$

Simulation du résultat avec Simulink

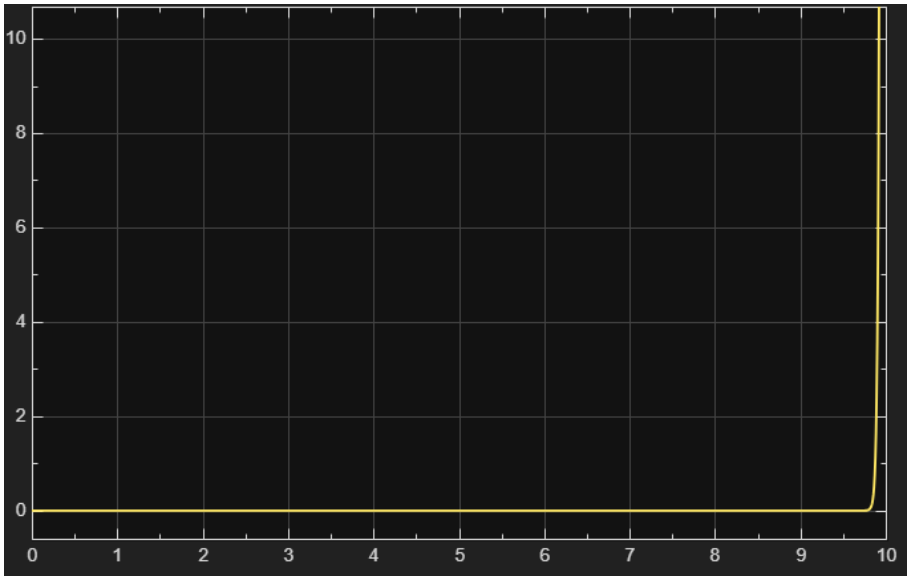


Figure: Échec de la lévitation

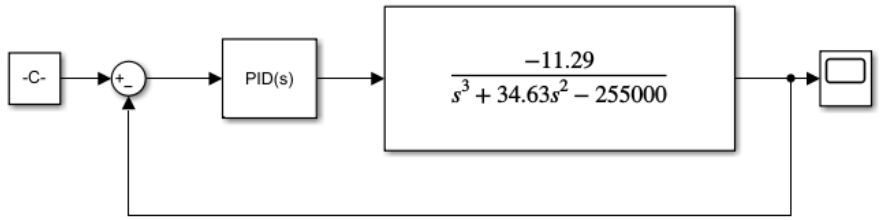
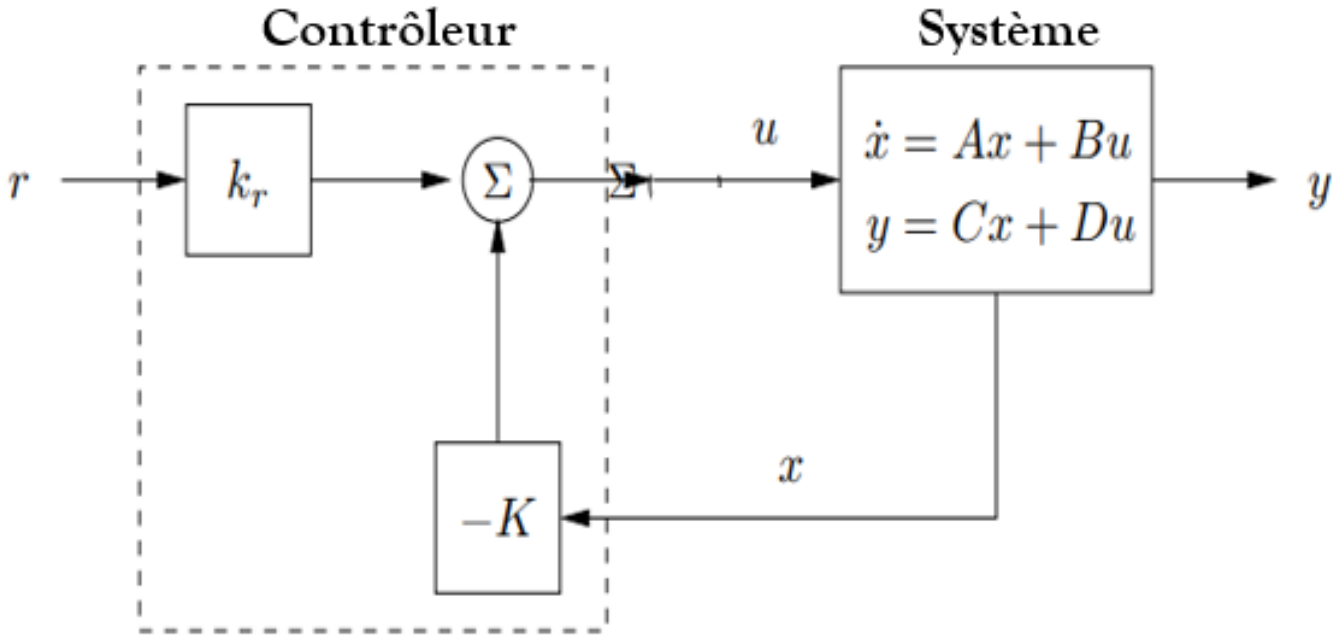


Figure: Schéma bloc du contrôle PID

Échec du contrôle PID

L'entrefer diverge inexorablement de sa consigne, compromettant la stabilité du système: la loi de commande PID est inopérante

Schéma de commande en retour d'état

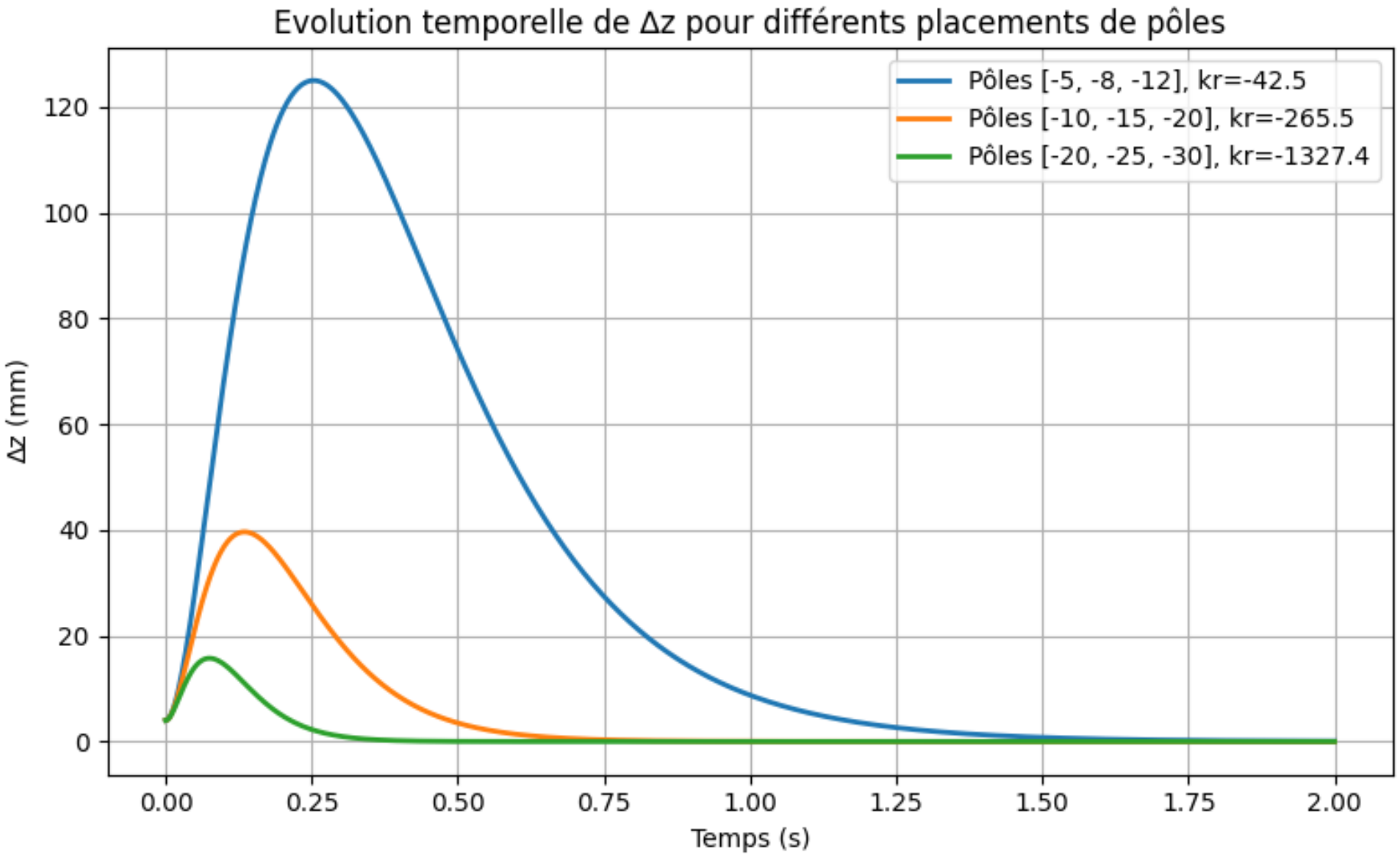


Système de commande en rétroaction avec retour d'état

Loi de commande et dynamique en boucle fermée

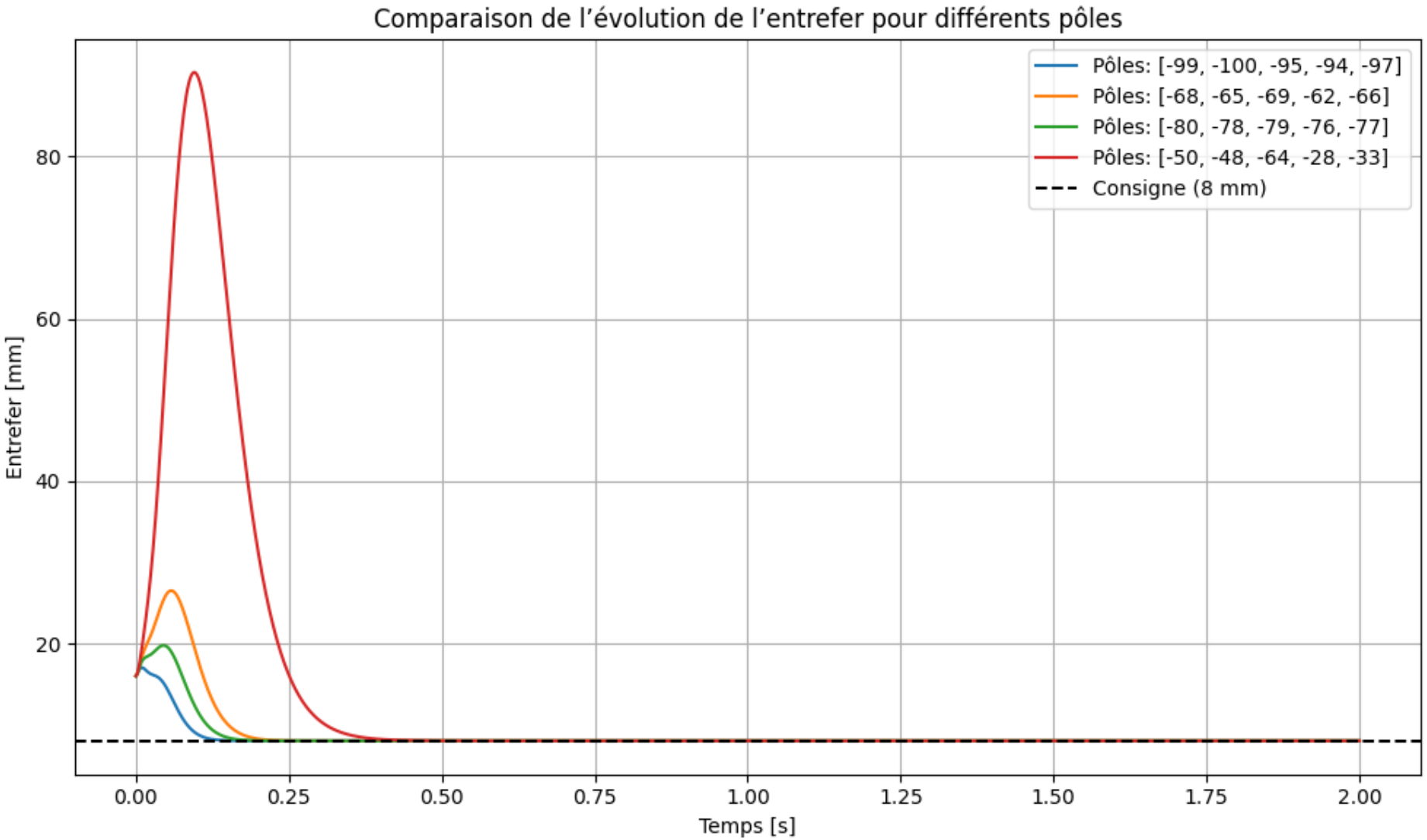
$$u(t) = k_r\, r(t) \;-\; K\, x(t), \quad \dot{x}(t) = \big(A - B\, K\big)\, x(t) \;+\; B\, k_r\, r(t).$$

Résutats - Python



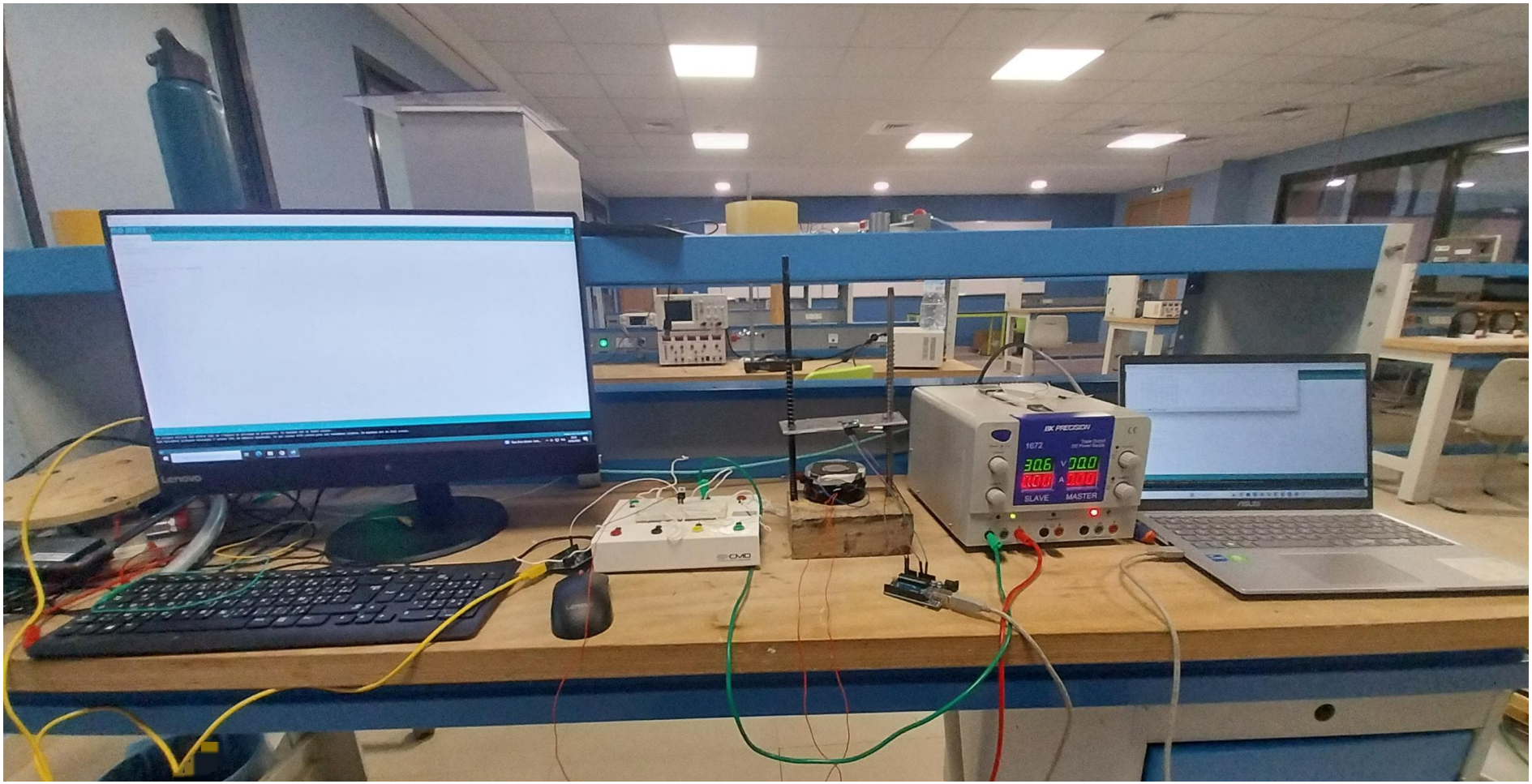
Test du correcteur - Voie flexible

Résultats de la simulation

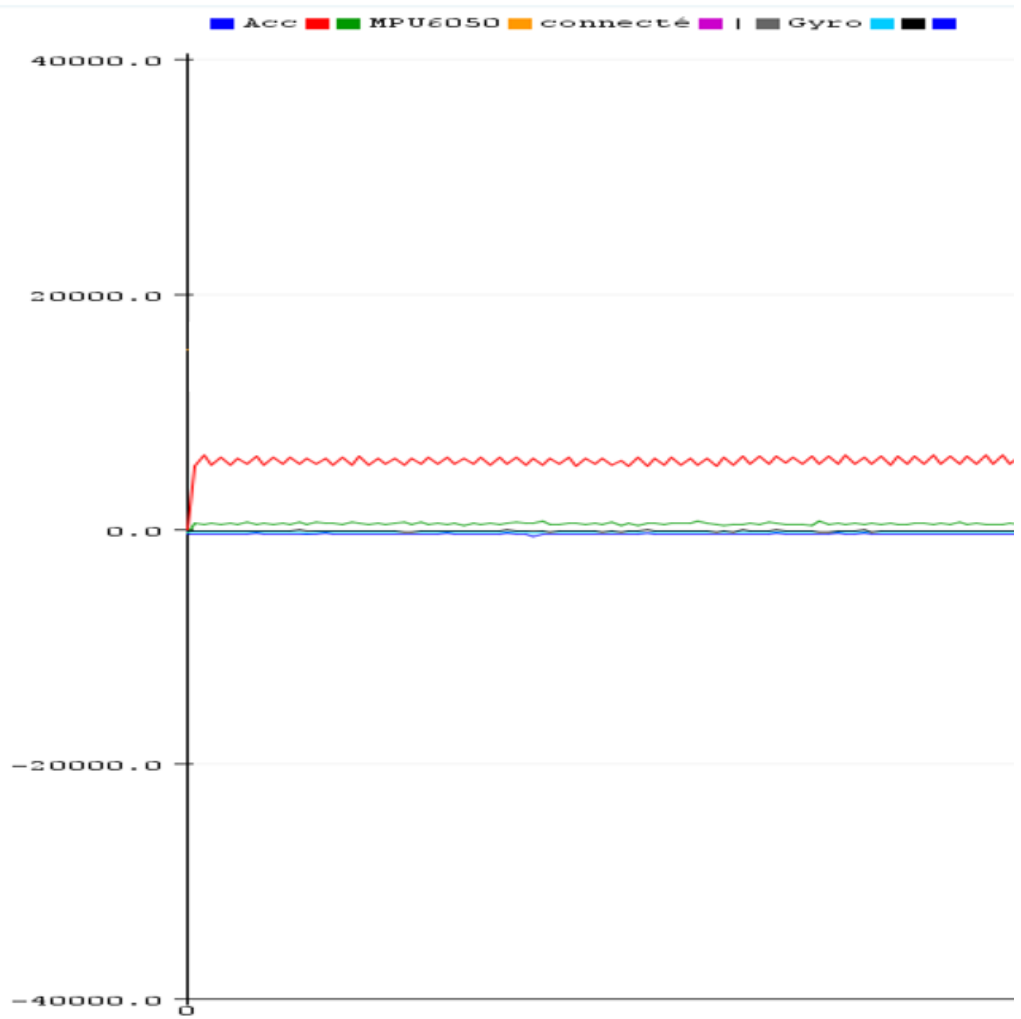


Manipulation expérimentale

Montage expérimental



Graphique généré par le Traceur série



Conclusion

Annexe : code Python – Système non corrigé

```python
import numpy as np
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt

# === Constantes physiques ===
m = 500        # Masse électroaimant [kg]
M = 1000       # Masse du train [kg]
g = 9.81       # Gravité [m/s^2]
mu0 = 4 * np.pi * 1e-7  # Perméabilité vide [H/m]
N = 700        # Nombre de spires
A = 0.003     # Section de noyau [m^2]
R = 4         # Résistance bobine [Ohm]
y0 = 0.008     # Position d'équilibre [m]
L = 1          # Longueur voie [m]
Eb = 2e11      # Module Young [Pa]
Ib = 1e-6      # Moment quadratique [m^4]
rho_b = 7800   # Masse volumique [kg/m³]
zeta = 0.005   # Amortissement

# === Paramètres dérivés ===
C2 = (2 / M) * np.sin(np.pi * y0 / L)**2
C1 = mu0 * N**2 * A / 4
omega = (np.pi**2 / L**2) * np.sqrt(Eb * Ib / rho_b)
# === Système d'équations d'état non linéaire ===
def state_equations(t, X, u):
    x1, x2, x3, x4, x5 = X
    delta = x1 - x3
    dx1 = x2
    dx2 = ((m + M) / m) * g - (C1 / m) * (x5 / delta)**2
    dx3 = x4
    dx4 = C2 * C1 * (x5 / delta)**2 - 2 * zeta * omega * x4 - omega**2 * x3
    dx5 = (x5 / delta) * (x2 - x4) - (R / (2 * C1)) * delta * x5 + (delta * u) / (2 * C1)
    return [dx1, dx2, dx3, dx4, dx5]
```

Annexe : code Python – Système non corrigé

```python
u = 10  # Tension constante [V]
t_span = (0, 2)  # Durée [s]
t_eval = np.linspace(t_span[0], t_span[1], 1000)

# Conditions initiales avec perturbation
X0 = [0.01, 0, 0.001, 0.1, 30]  # [x1, x2, x3, x4, x5]

# === Intégration numérique ===
sol = solve_ivp(state_equations, t_span, X0, args=(u,), t_eval=t_eval)

# courbe 1 : Position aimant ===
plt.figure()
plt.plot(sol.t, sol.y[0], color='blue')
plt.title("Position de l'électroaimant ")
plt.xlabel("Temps [s]")
plt.ylabel(" Position électroaimant [m]")
plt.grid()
plt.tight_layout()
plt.show()

# courbe 2 : Déformation de la voie ===
plt.figure()
plt.plot(sol.t, sol.y[2], color='green')
plt.title("Déformation de la voie ")
plt.xlabel("Temps [s]")
plt.ylabel(" Position voie [m]")
plt.grid()
plt.tight_layout()
plt.show()
```

Annexe : code Python – Système non corrigé

```
# courbe 3 : Entrefer ===
entrefer = sol.y[0] - sol.y[2]
plt.figure()
plt.plot(sol.t, entrefer, color='purple')
plt.title("Variations de l''entrefer ")
plt.xlabel("Temps [s]")
plt.ylabel("Entrefer [m]")
plt.grid()
plt.tight_layout()
plt.show()
```

Annexe : code Python – Commande par retour d'état - Voie Rigide

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import place_poles, StateSpace, lsim

# 1. Paramètres physiques
m    = 500.0       # kg
R    =   4.0       # Ω
N    = 700         # spires
Asec = 3e-3        # m²
z0   = 8e-3        # m (équilibre)
mu0  = 4*np.pi*1e-7
i0   = 45.2        # A (équilibre)
u0   = R * i0      # V (équilibre)

# 2. Constantes linéaires
ki = mu0 * N**2 * Asec * i0 / (2 * z0**2)
kz = mu0 * N**2 * Asec * i0**2 / (2 * z0**3)
C1 = mu0 * N**2 * Asec / 4
L0 = 2 * C1 / z0

# Modèle en espace d'état linéarisé (voie rigide)
A = np.array([
    [0,      1,      0],
    [kz/m,   0,   -ki/m],
    [0,    ki/L0, -R/L0]
])
B = np.array([[0], [0], [1/L0]])
C = np.array([[1, 0, 0]])
D = np.array([[0]])
```

Annexe : code Python – Commande par retour d'état - Voie Rigide

```python
# pôles à tester
poles_dict = {
    "[-5, -8, -12]":    [-5, -8, -12],
    "[-10, -15, -20]": [-10, -15, -20],
    "[-20, -25, -30]": [-20, -25, -30]}

# 5. Simulation de la réponse libre (Δz(0)=4 mm, r=0)
t = np.linspace(0, 2, 1000)
u_in = np.zeros_like(t)
x0 = [0.004, 0, 0]  # Δz(0)=4 mm

plt.figure(figsize=(8,5))
for label, p_set in poles_dict.items():
    # 5.1 Placement de pôles et calcul des gains
    K = place_poles(A, B, p_set).gain_matrix
    kr = float(-1.0 / (C @ np.linalg.inv(A - B@K) @ B))

    # 5.2 Système en boucle fermée (r=0)
    Acl = A - B @ K
    sys_cl = StateSpace(Acl, np.zeros_like(B), C, D)

    # 5.3 Réponse libre
    _, y, _ = lsim(sys_cl, u_in, t, X0=x0)

    plt.plot(t, y * 1e3, linewidth=2, label=f"Pôles {label}, kr={kr:.1f}")

# 6. Finitions du tracé
plt.title("Évolution temporelle de Δz pour différents placements de pôles")
plt.xlabel("Temps (s)")
plt.ylabel("Δz (mm)")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

Annexe : code Python – Commande par retour d'état - Voie Flexible

```python
import numpy as np
from scipy.signal import place_poles
from scipy.linalg import inv
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt

# --- Matrices du système linéarisé ---
A = np.array([
    [0, 1, 0, 0, 0],
    [2*C1*C3**2/(m*z0), 0, -2*C1*C3**2/(m*z0), 0, -2*C1*C3/(m*z0)],
    [0, 0, 0, 1, 0],
    [-2*C2*C3**2/z0, 0, 2*C2*C3**2/z0 - omega1**2, -2*eta1*omega1, 2*C2*C3/z0],
    [0, C3, 0, -C3, -z0 * R / (2 * C1)]
])
B = np.array([[0], [0], [0], [0], [z0 / (2 * C1)]])
C = np.array([[1, 0, -1, 0, 0]])

# --- État d'équilibre x0 ---
x1_0 = C2 * C3**2 / omega1**2 + z0
x3_0 = C2 * C3**2 / omega1**2
x5_0 = z0 * C3
x0 = np.array([x1_0, 0, x3_0, 0, x5_0])
u0 = R * x5_0

# --- Condition initiale : entrefer = 16 mm
x_init = x0.copy()
x_init[0] += 0.008  # z1 est 8 mm au-dessus de l'équilibre

# --- Temps de simulation
t_span = (0, 2)
t_eval = np.linspace(*t_span, 1000)

```

Annexe : code Python – Commande par retour d'état - Voie Flexible

```python
# --- pôles à tester
pole_sets = [
    [-99, -100, -95, -94, -97],
    [-68, -65, -69, -62, -66],
    [-80, -78, -79, -76, -77],
    [-50, -48, -64, -28, -33]
]
results = []
for poles in pole_sets:
    try:
        K = place_poles(A, B, poles).gain_matrix
        kr = 1 / (C @ inv(A - B @ K) @ B)
        r = 0.0

        def closed_loop(t, x):
            dx = A @ (x - x0).reshape(-1, 1) + B * (u0 + kr * r - K @ (x - x0).reshape(-1, 1) - u0)
            return dx.flatten()

        sol = solve_ivp(closed_loop, t_span, x_init, t_eval=t_eval)
        z1 = sol.y[0]
        zg = sol.y[2]
        gap = z1 - zg
        results.append((poles, t_eval, gap))
    except Exception as e:
        print(f"Erreur avec pôles {poles}: {e}")
# --- Tracé
plt.figure(figsize=(10, 6))
for poles, t, gap in results:
    plt.plot(t, gap * 1000, label=f"Pôles: {poles}")
plt.axhline(z0 * 1000, color='k', linestyle='--', label='Consigne (8 mm)')
plt.xlabel("Temps [s]")
plt.ylabel("Entrefer [mm]")
plt.title("Comparaison de l'évolution de l'entrefer pour différents pôles")
plt.legend()
plt.grid()
plt.tight_layout()
plt.show()
```

Annexe: code Arduino - Montage 1

```ino
#include "HX711.h"

// ---------- Cellule de charge via HX711 ----------
#define DT  3
#define SCK 2
HX711 scale;

float calibration_factor = -1000; // À ajuster
const float g = 9.81; // m/s²

// ---------- Capteur Effet Hall analogique ----------
#define HALL_PIN A0
const float Vcc = 5.0;
const float sensitivity_Vper_mT = 0.025; // ex: SS49E → 25 mV/mT
const float zero_field_voltage = 2.5;    // tension sans champ

void setup() {
  Serial.begin(9600);

  // HX711
  scale.begin(DT, SCK);
  scale.set_scale(calibration_factor);
  scale.tare(); // Mise à zéro

  Serial.println("Lecture HX711 + Capteur Hall...");
}

void loop() {
  // --- Cellule de charge ---
  float masse_kg = scale.get_units(5); // Moyenne sur 5 lectures
  float force_N = masse_kg * g;

  // --- Effet Hall ---
  int raw = analogRead(HALL_PIN);
  float voltage = raw * (Vcc / 1023.0);
  float deltaV = voltage - zero_field_voltage;
  float champ_mT = deltaV / sensitivity_Vper_mT;

  // --- Affichage ---
  Serial.print("Masse: ");
  Serial.print(masse_kg, 3);
  Serial.print(" kg\tForce: ");
  Serial.print(force_N, 2);
  Serial.print(" N\t");

  Serial.print("Champ magnétique: ");
  Serial.print(champ_mT, 2);
  Serial.println(" mT");

  delay(500);
}
```

Annexe: code Arduino - Montage 2

```
// Moteur branché sur l'interface 3
int moteur = 3;

void setup() {
  // Definir le moteur en sortie
  pinMode(moteur, OUTPUT);
}

void loop() {
  // Alimenter le moteur
  digitalWrite(moteur, HIGH);
  // Pause 1 seconde
  delay(1000);
  // Eteindre le moteur
  digitalWrite(moteur, LOW);
  // Pause 1 seconde
  delay(1000);
}
```

```
void setup() {
  // Configuration de la broche 3 en sortie
  pinMode(3, OUTPUT);
}

void loop() {
  // Mettre la broche 3 à l'état haut pendant 500 ms
  digitalWrite(3, HIGH);
  delay(500);

  // Mettre la broche 3 à l'état bas pendant 500 ms
  digitalWrite(3, LOW);
  delay(500);
}
```

Annexe: code Arduino - Montage 2

```
#include <Wire.h>
#include <MPU6050.h>

MPU6050 mpu;

void setup() {
  Serial.begin(9600);
  Wire.begin();
  mpu.initialize();

  if (mpu.testConnection()) {
    Serial.println("MPU6050 connecté !");
  } else {
    Serial.println("Échec de connexion au MPU6050.");
    while (1);
  }
}

void loop() {
  int16_t ax, ay, az;
  int16_t gx, gy, gz;

  // Lire les données brutes
  mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

  // Afficher les valeurs
  Serial.print("Acc: ");
  Serial.print(ax); Serial.print(" ");
  Serial.print(ay); Serial.print(" ");
  Serial.print(az); Serial.print(" | Gyro: ");
  Serial.print(gx); Serial.print(" ");
  Serial.print(gy); Serial.print(" ");
  Serial.println(gz);

  delay(500);
}
```

Annexe: Équation de vibration de poutre

$$E_g I_g \frac{\partial^4 z_g}{\partial l^4} + \rho_g \frac{\partial^2 z_g}{\partial t^2} + b_g \frac{\partial z_g}{\partial t} = p(l,t) \quad (1)$$

On injecte $z_g(l,t) = \phi_1(l)\, q_1(t)$ avec $\phi_1(l) = \sqrt{\frac{2}{m_g}} \sin\left(\frac{\pi l}{L}\right)$. dans l'équation:

$$E_g I_g \left(\frac{\pi}{L}\right)^4 \phi_1(l)\, q_1(t) + \rho_g\, \phi_1(l)\, \ddot{q}_1(t) + b_g\, \phi_1(l)\, \dot{q}_1(t) = p(l,t) \quad (2)$$

En multipliant par $\phi_1(l)$:

$$E_g I_g \left(\frac{\pi}{L}\right)^2 \phi_1(l)^2\, q_1(t) + \rho_g\, \phi_1(l)^2\, \ddot{q}_1(t) + b_g\, \phi_1(l)^2\, \dot{q}_1(t) = p(l,t)\, \phi_1(l) \quad (3)$$

Annexe: Équation de vibration de poutre

Etant donné,

$$\int_0^L \phi_1(x)^2 \, dx \;=\; \frac{L}{m_g} \;=\; \frac{1}{\rho_g}$$

En intégrant l'équation (3), on obtient l'équation modale :

$$\ddot{q}_1(t) + \frac{b_g}{\rho_g}\,\dot{q}_1(t) + \left(\frac{\pi}{L}\right)^4 \frac{E_g I_g}{\rho_g}\, q_1(t) \;=\; Q_1(t)$$

avec

$$Q_1(t) \;=\; \int_0^L p(x,t)\,\phi_1(x)\,dx.$$

Annexe: Équation de vibration de la poutre

En raison des dimensions de l'électroaimant, on peut écrire :

$$p(l,t) = F_e\,\delta(x-l)\,.$$

Donc :

$$Q_1(t) = \phi_1(l)\,F_e\,.$$

En posant

$$\omega_1 = \left(\frac{\pi}{L}\right)^2\sqrt{\frac{E_g I_g}{\rho_g}}, \quad \eta_1 = \frac{b_g}{2\,\rho_g\,\omega_1},$$

l'équation modale s'écrit enfin :

$$\ddot{q}_1(t) + 2\,\eta_1\,\omega_1\,\dot{q}_1(t) + \omega_1^2\,q_1(t) = \phi_1(l)\,F_e$$