

# Mathématiques HEI

Robin Delabays

January 9, 2026

# Table of contents

<b>Liste des cours</b>	<b>3</b>
<b>I    Analyse 3</b>	<b>4</b>
<b>Semestre d'automne 2025-26</b>	<b>5</b>
<b>Activité du mardi 4 novembre 2025</b>	<b>6</b>
Consignes . . . . .	6
Exercice . . . . .	6
<b>Système masse-ressort-amortisseur interactif (Desmos)</b>	<b>12</b>
Marche à suivre . . . . .	12

# Liste des cours

Cette page regroupe quelques documents additionnel pour les cours de mathématiques donnés à la Haute Ecole d'Ingénierie de Sion.

- [Analyse 3](#)

**Part I**

**Analyse 3**

# Semestre d'automne 2025-26

Les pages qui suivent regroupent les activités en ligne du cours d'Analyse 3, pour la classe SE3fb de l'automne 2025.

- [Activité du 4 novembre 2025](#)
- [Système masse-ressort-amortisseur interactif](#)

# Activité du mardi 4 novembre 2025

## Consignes

- Les zones de code ci-dessous peuvent être modifiées et relancées à l'aide du bouton “Run code” ou au clavier avec “Ctrl+Enter”.
- Vous serez amené à faire quelques calculs. Ces passages sont en italique.
- Il faut lancer les cases de codes dans l'ordre, car certaines font appel aux cases précédentes.

## Exercice

Dans l'exercice 3 de la série 21 du cours de Mathématiques Appliquées 1, nous avons considéré le système suivant, où la constante du ressort est de  $k = 4[\text{kg/s}^2]$ , la masse du chariot est de  $m = 1[\text{kg}]$ , et la constante de l'amortisseur est de  $c = 5[\text{kg/s}]$ .

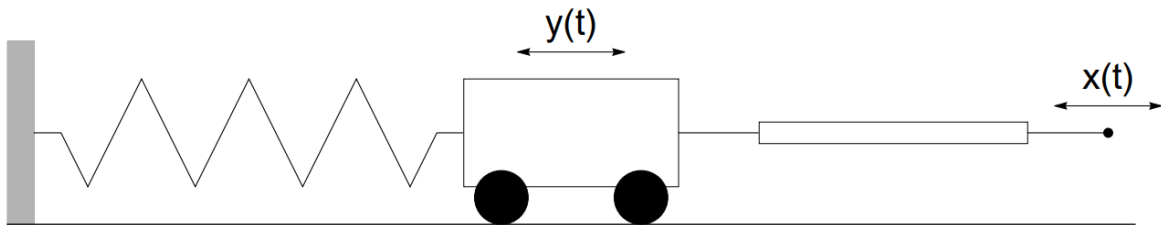


Figure 1: Chariot

On en déduit l'équation différentielle

$$y'' + 5y' + 4y = 5x'$$

Pour rappel, la réponse fréquentielle du système ci-dessus est donnée par

$$H(\omega) = \frac{i5\omega}{4 - \omega^2 + i5\omega}$$

(Chaque dérivée est remplacée par une multiplication par  $i\omega$  et on a le membre de droite au numérateur et le membre de gauche au dénominateur.)

Pour commencer, représentons le module et l'argument de  $H(\omega)$  en fonction de la vitesse angulaire  $\omega$ :

```
# On charge les librairies nécessaires.
import numpy as np
import matplotlib.pyplot as plt
import cmath
j = complex(0,1) # On définit l'unité imaginaire comme 'j'.

# On calcule la réponse fréquentielle pour des vitesses angulaires allant de 0 à 100 [rad/s]
w = np.linspace(0,100,500)
H = (j*5*w)/(4 - w**2 + j*5*w)

# On affiche les graphes de |H| et de arg(H).
plt.figure('fig1',(8,4))
plt.subplot(1,2,1)
plt.plot(w,abs(H))
plt.xlabel('w')
plt.ylabel('|H|')

# Calcul de l'argument de la réponse fréquentielle.
phi = []
for i in np.arange(len(w)):
    phi.append(cmath.phase(H[i]))

plt.subplot(1,2,2)
plt.plot(w,phi)
plt.xlabel('w')
plt.ylabel('arg(H)')

plt.show()
```

---

Pour le moment, nous n'avons pas spécifié la forme du signal d'entrée  $x(t)$ . Pour rappel, ce signal d'entrée donne la position du point d'attache mobile, à droite du chariot dans le système considéré ici.

Dans ce premier exemple, nous allons prendre le signal d'entrée

$$x(t) = \sin(3t)$$

```
t = np.linspace(0,10,500)
x = np.sin(3*t)

plt.figure('fig2',(8,4))
plt.plot(t,x)
plt.xlabel('t')
plt.ylabel('x')
plt.show()
```

Afin de calculer le signal de sortie  $y(t)$  (donc la position du chariot), nous allons utiliser la réponse fréquentielle  $H(\omega)$ . Pour cela nous avons besoin de la série de Fourier complexe de  $x(t)$ ,

$$x(t) = \sin(3t) = -\frac{1}{2i}e^{-i3t} + \frac{1}{2i}e^{i3t}$$

Comme nous l'avons vu, on trouve le signal de sortie en appliquant la réponse fréquentielle à chaque harmonique de la série de Fourier, avec la fréquence associée. Ainsi, pour le terme  $e^{-i3t}$ , nous devons calculer  $H(-3)$ , et pour le terme  $e^{i3t}$ , nous devons calculer  $H(3)$ .

---

Vérifiez que

$$H(-3) = \frac{9+3i}{10}$$

$$H(3) = \frac{9-3i}{10}$$


---

Ainsi, la série de Fourier complexe du signal de sortie est

$$y(t) = -\frac{1}{2i} \cdot \frac{9+3i}{10} e^{-i3t} + \frac{1}{2i} \cdot \frac{9-3i}{10} e^{i3t} = \frac{-3+9i}{20} e^{-i3t} + \frac{-3-9i}{20} e^{i3t}$$

Sans avoir besoin de simplifier cette expression, on peut se faire une idée de la trajectoire du chariot en la calculant numériquement :



```

y = complex(-3/20,9/20)*np.exp(-j*3*t) + complex(-3/20,-9/20)*np.exp(j*3*t)

plt.figure()
plt.subplot(2,1,1)
plt.plot(t,x)
plt.ylabel('x'); plt.axis([0,10,-1.1,1.1])
plt.subplot(2,1,2)
plt.plot(t,y.real,color='C1')
plt.xlabel('t'); plt.ylabel('y'); plt.axis([0,10,-1.1,1.1])
plt.show()

```

---

Dans le code ci-dessous, en modifiant la vitesse angulaire du signal d'entrée ( $w$  à la première ligne), vous pouvez voir directement l'effet sur le signal de sortie  $y(t)$  dans la figure.

*Qu'observez-vous lorsque la vitesse angulaire devient très basse (mais non-nulle) ? Et lorsqu'elle est très élevée ?*

```

w = 3    # Vitesse angulaire à modifier

#####

tmax = np.max([10,30/w])
t = np.linspace(0,tmax,500)
x = np.sin(w*t)

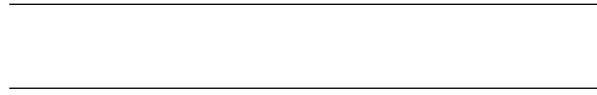
# Calcul de la réponse fréquentielle pour -w et +w
H1 = (-j*5*w)/(4-w**2 - j*5*w)
H2 = (j*5*w)/(4-w**2 + j*5*w)

# Calcul du signal de sortie
y = (-1/(2*j))*H1*np.exp(-j*w*t) + (1/(2*j))*H2*np.exp(j*w*t)

# Affichage
plt.figure()
plt.subplot(2,1,1)
plt.plot(t,x)
plt.ylabel('x'); plt.axis([0,tmax,-1.1,1.1])
plt.subplot(2,1,2)
plt.plot(t,y.real,color='C1')

```

```
plt.xlabel('t'); plt.ylabel('y'); plt.axis([0,tmax,-1.1,1.1])
plt.show()
```



Considérons maintenant un signal d'entrée  $x(t)$  (donc l'excitation du chariot) tel que :

- $x(t) = 2t - \frac{\pi}{2}$  pour  $0 \leq t \leq \pi/2$
- $x(t)$  est pair
- $x(t)$  est  $\pi$ -périodique



Esquissez le graphe de  $x(t)$ .

Vérifiez que les coefficients de Fourier complexes sont données par

$$c_k = \begin{cases} -\frac{2}{\pi k^2} & k \text{ impair} \\ 0 & k \text{ pair} \end{cases}$$



Ainsi

$$x(t) = \dots - \frac{2}{9\pi} e^{-i6t} - \frac{2}{\pi} e^{-i2t} - \frac{2}{\pi} e^{i2t} - \frac{2}{9\pi} e^{i6t} - \dots$$

et on peut calculer la série de Fourier du signal de sortie (donc de la position du chariot) en appliquant la réponse fréquentielle à chaque harmonique

$$y(t) = \dots - \frac{2}{9\pi} H(-3) e^{-i6t} - \frac{2}{\pi} H(-1) e^{-i2t} - \frac{2}{\pi} H(1) e^{i2t} - \frac{2}{9\pi} H(3) e^{i6t} - \dots$$

Dans le code ci-dessous, on représente le résultat des 9 premières harmoniques de  $x(t)$  et  $y(t)$ .

```
# Grandeurs à modifier #####
m = 1 # masse du chariot
c = 5 # constante de l'amortisseur
k = 4 # constante du ressort
#####

def H(w):
```

```

    return (c*j*w)/(k - w**2 + c*j*w)

tmin = -5; tmax = 10; t = np.linspace(tmin,tmax,500)

# Calcul de chaque harmonique de x(t)
def X(k):
    return -2/(np.pi*k**2)*(np.exp(-j*2*k*t) + np.exp(j*2*k*t))

x1 = X(1); x3 = X(3); x5 = X(5); x7 = X(7); x9 = X(9); x = x1 + x3 + x5 + x7 + x9

# Calcul des harmoniques pour y(t)
def Y(k):
    return -2/(np.pi*k**2)*(H(-k)*np.exp(-j*2*k*t) + H(1)*np.exp(j*2*k*t))

y1 = Y(1); y3 = Y(3); y5 = Y(5); y7 = Y(7); y9 = Y(9); y = y1 + y3 + y5 + y7 + y9

# Affichage
plt.figure()
plt.subplot(2,1,1)
plt.plot(t,x.real)
plt.axis([tmin,tmax,-2,2]); plt.ylabel('x')
plt.subplot(2,1,2)
plt.plot(t,y.real,color='C1')
plt.xlabel('t'); plt.ylabel('y'); plt.axis([tmin,tmax,-2,2])
plt.show()

```

# Système masse-ressort-amortisseur interactif (Desmos)

Le code ci-dessous permet de calculer les paramètres à entrer dans [ce calculateur Desmos](#). Le système est composé (de gauche à droite) d'un point fixe, d'un amortisseur (de constante  $d$ ), d'une masse ( $m = 1$ , position  $y(t)$ ), d'un ressort (de constante  $k$ ) et d'un point mobile qui suit un horaire sinusoïdal ( $x(t) = \sin(\omega t)$ ).

## Marche à suivre

Dans le champ de code ci-dessous, entrer les constantes d'amortissement  $d$  et du ressort  $k$ , ainsi que la vitesse angulaire de l'excitation  $\omega$ . En sortie, le code affiche les quatre constantes  $A$ ,  $B$ ,  $C$  et  $D$  à renseigner au [calculateur Desmos](#), ainsi que :

- les deux pôles ( $s_1$  et  $s_2$ ) de la fonction de transfert si ceux-ci sont réels ;
- les parties réelle ( $\alpha$ ) et imaginaire ( $\beta$ ) des pôles de la fonction de transfert si ceux-ci sont conjugués complexes.

Dans le [calculateur Desmos](#) :

1. Renseigner les paramètres  $A$ ,  $B$ ,  $C$  et  $D$  dans les champs correspondants ;
2. Renseigner les pôles ( $s_1$  et  $s_2$ ) ou leur partie réelle ( $\alpha$ ) et imaginaire ( $\beta$ ) dans les champs correspondants ;
3. Adapter la définition de  $x_t$  :
  - Utiliser  $x_t = x_0 + x_R$  si les pôles de la fonction de transfert sont réels ;
  - Utiliser  $x_t = x_0 + x_I$  si les pôles de la fonction de transfert sont complexes ;
4. Lancer l'animation en laissant  $t$  évoluer.

```
import numpy as np
import matplotlib.pyplot as plt

d = 5
k = 4
omega = 1
```

```

# Discriminant pour la partie homogène (m = 1)
Delta = d**2 - 4.0 * 1.0 * k

if Delta < 0:
    print(f"Irréductible,  $\Delta = \{Delta\}$ :")

    # Système linéaire pour A, B, C, D
    M = np.array([
        [1.0, 0.0, 1.0, 0.0],
        [d, 1.0, 0.0, 1.0],
        [k, d, omega**2, 0.0],
        [0.0, k, 0.0, omega**2]
    ], dtype=float)
    rhs = np.array([0.0, 0.0, 0.0, k*omega], dtype=float)
    A, B, C, D = np.linalg.solve(M, rhs)

    alpha = -d / 2.0
    beta = np.sqrt(4.0 * k - d**2) / 2.0

    print(f"A = {A}")
    print(f"B = {B}")
    print(f"C = {C}")
    print(f"D = {D}")
    print(f" = {alpha}")
    print(f" = {beta}")

    # Figure des pôles et de +/- omega
    plt.figure()
    plt.axhline(0, color='k')
    plt.axvline(0, color='k')
    # Points (alpha, ±beta)
    plt.plot([alpha, alpha], [beta, -beta], "o")
    # Points (0, ±omega)
    plt.plot([0, 0], [omega, -omega], "o")
    plt.title("Pôles complexes et ±")
    plt.xlabel("Réel")
    plt.ylabel("Imaginaire")
    plt.axis([-7,7,-7,7])
    plt.show()

    # Signaux temporels
    t = np.linspace(0.0, 10.0, 200)

```

```

x = np.sin(omega * t)
y = (A * np.cos(omega * t) + (B / omega) * np.sin(omega * t) + np.exp(alpha * t) * (C * np

plt.figure()
plt.subplot(2, 1, 1)
plt.plot(t, x)
plt.ylabel("x(t)")
plt.title("Entrée et sortie ( $\Delta < 0$ )")
plt.subplot(2, 1, 2)
plt.plot(t, y)
plt.xlabel("t")
plt.ylabel("y(t)")
plt.tight_layout()
plt.show()

else:
    print(f"Réductible,  $\Delta = \{\Delta\}$ :")
    s1 = (-d + np.sqrt(Delta)) / 2.0
    s2 = (-d - np.sqrt(Delta)) / 2.0

    # Système linéaire pour A, B, C, D
    M = np.array([
        [1.0, 0.0, 1.0, 1.0],
        [-(s1+s2), 1.0, -s2, -s1],
        [s1*s2, -(s1+s2), omega**2, omega**2],
        [0.0, s1*s2, -omega**2*s2, -omega**2*s1]
    ], dtype=float)
    rhs = np.array([0.0, 0.0, 0.0, k*omega], dtype=float)
    A, B, C, D = np.linalg.solve(M, rhs)

    print(f"A = {A}")
    print(f"B = {B}")
    print(f"C = {C}")
    print(f"D = {D}")
    print(f"s1 = {s1}")
    print(f"s2 = {s2}")

    # Figure des pôles réels et de +/- omega
    plt.figure()
    plt.axhline(0, color='k')
    plt.axvline(0, color='k')
    plt.plot([s1, s2], [0.0, 0.0], "o") # Points (s1, 0) et (s2, 0)

```

```

plt.plot([0.0, 0.0], [omega, -omega], "o") # Points (0, ±omega)
plt.title("Pôles réels et ±")
plt.xlabel("Réel")
plt.ylabel("Imaginaire")
plt.axis([-7,7,-7,7])
plt.show()

# Signaux temporels
t = np.linspace(0.0, 50.0, 200)
x = np.sin(omega * t)
y = (A * np.cos(omega * t) + (B / omega) * np.sin(omega * t) + C * np.exp(s1 * t) + D * np.exp(s2 * t))

plt.figure()
plt.subplot(2, 1, 1)
plt.plot(t, x)
plt.ylabel("x(t)")
plt.title("Entrée et sortie ( $\Delta = 0$ )")
plt.subplot(2, 1, 2)
plt.plot(t, y)
plt.xlabel("t")
plt.ylabel("y(t)")
plt.tight_layout()
plt.show()

```