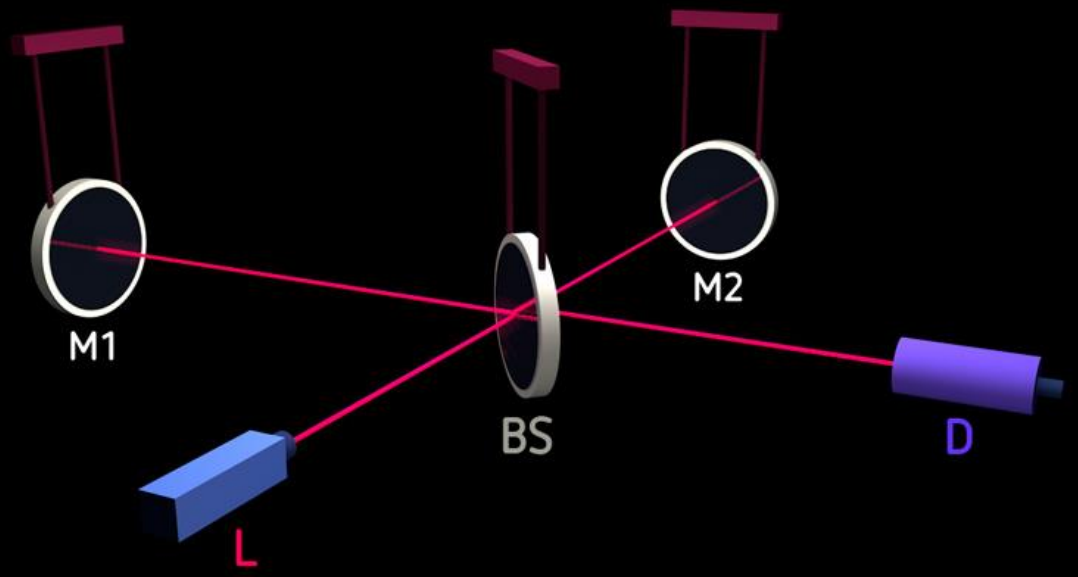
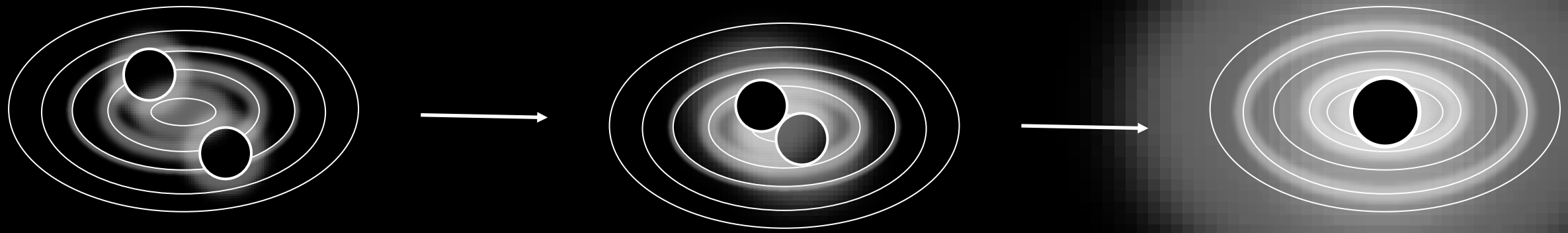
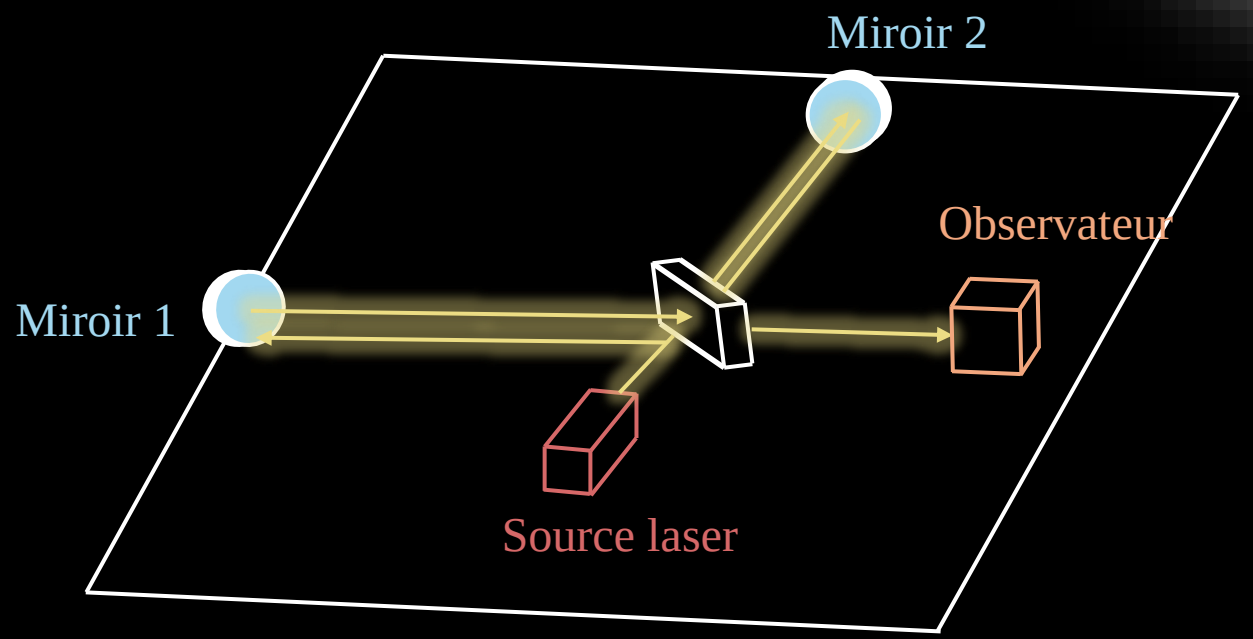
Projet de modélisation numérique - Le double pendule dans les interféromètres

Valentin, Ali

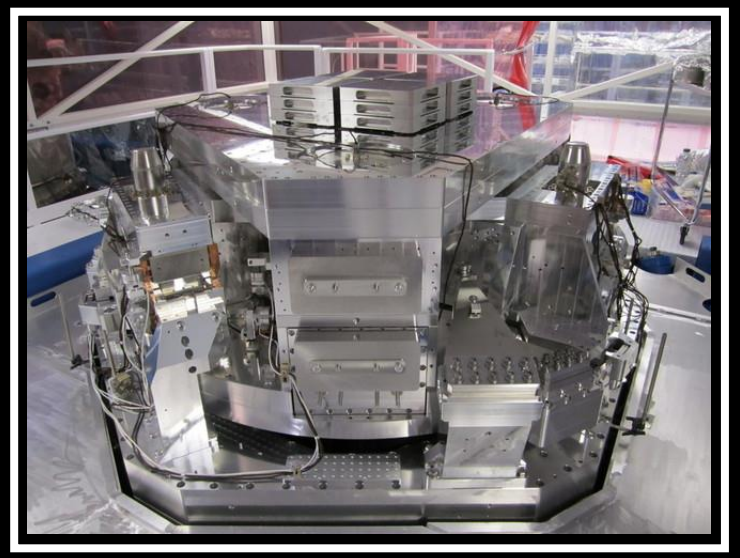






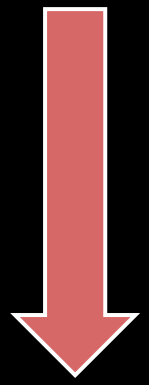
Déplacement de l'ordre de $10^{-21}\ m$

Mécanismes de suspensions



Miroirs optiques

Comment le **double pendule** peut-il être utilisé en tant que mécanisme de suspension pour stabiliser les **miroirs optiques** des interféromètres contre le **bruit sismique** ?



Comment concevoir un **algorithme de contrôle numérique** pour stabiliser un **système donné**, en tenant compte des **contraintes externes** ?


jupyter

Adaptive Modal Damping for Advanced LIGO Suspensions

by

Brett Noah Shapiro

B.S.,Engineering Science (2005)
The Pennsylvania State University

S.M., Mechanical Engineering (2007)
Massachusetts Institute of Technology

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Mechanical Engineering
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
June 2012

Author .....
Department of Mechanical Engineering
May 11, 2012

Certified by .....
Nergis Mavalvala
Professor of Physics
Thesis Supervisor

Certified by .....
Kamal Youcef-Toumi
Professor of Mechanical Engineering
Thesis Supervisor

Accepted by .....
David E. Hardt
Chairman, Committee on Graduate Students

Brett Noah Shapiro, 2012 (MIT)

Chapter 2

The Quadruple Pendulum

The quadruple pendulum is an extension of the three stage, triple pendulum designed for use in the German-British Gravitational Wave Detector (GEO600). Reference [33] details the design of the triple pendulum and is useful for reviewing the thought process used in extending the mathematics behind a single pendulum to an n-stage pendulum. References [34, 35, 36] describe the quadruple pendulum design.

2.1 Performance Requirements

2.1.1 Spectral Requirements

The purpose of the quadruple pendulum is to limit the contribution of seismically induced motions of the ITM and ETM optics within the GW detection band of Advanced LIGO starting at 10 Hz. Due to the strict nature of this requirement, care must be taken in the design of the pendulum such that it does not introduce other noise contributions that overpower and negate its seismic filtering property. Figure 2-1 shows the expected seismic motion of the table from which the pendulum hangs (blue line) and the required motion of the test mass (green line). To achieve the requirement given the input motion, the pendulum must provide more than six orders of magnitude isolation within the GW detection band.

The dominant two noise contributions stemming from the pendulum itself are

45

Utilisation d'un quadruple pendule

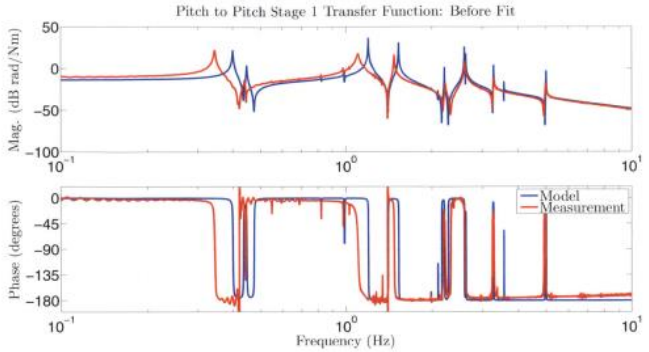


Figure 4-3: Original quadruple pendulum model against the measured stage 1 pitch to pitch transfer function. The blue curve is the model, the red the measurement.

measurements that contribute negligible information are not required.

The parameter estimation results show a significant improvement in the match between the model and the measurements. After the fit, the maximum percent error in mode frequencies is reduced from 16% to 1%. Thus, the selection procedure is effective in choosing those parameters that require estimation. The 27 least sensitive measurements contribute less than 1% of the parameter uncertainty information contained in the full set of 47 measurements. Discarding these measurements has negligible impact on the parameter estimation results. This measurement selection procedure is useful in determining how to focus limited resources to those measurements that are most important. For the quadruple pendulum parameter estimation problem, there would be no need to collect the least sensitive 27 measurements at all. The time required to measure them could be better spent refining the top 20.

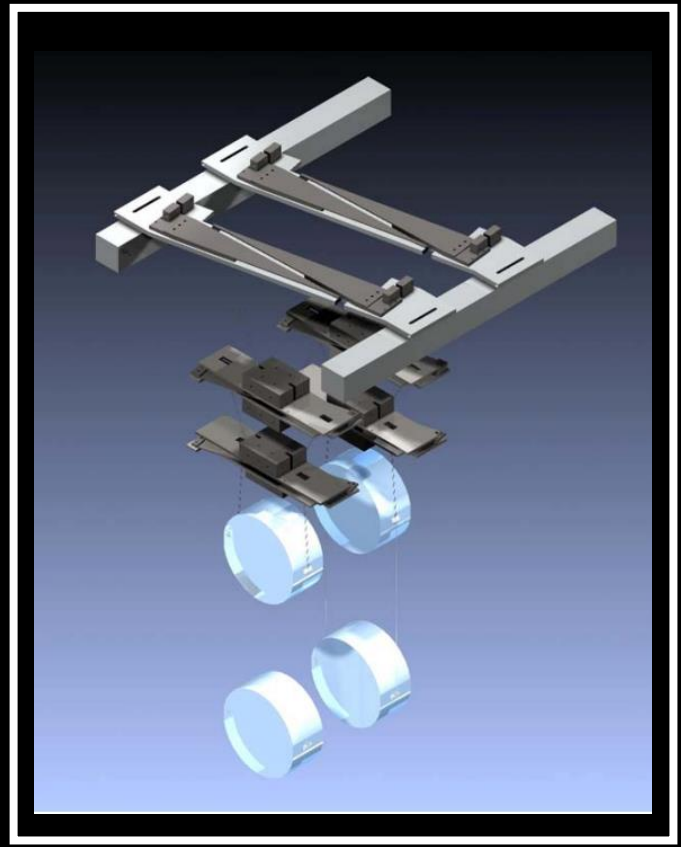
This sensitivity analysis also provides a lot of useful information about the quadruple pendulum itself. It states that the most important uncertainty is focused on effective wire and fiber bending points. Spring stiffness follows in importance. A distant third is rotational inertia. Some further investigation reveals that the bending points

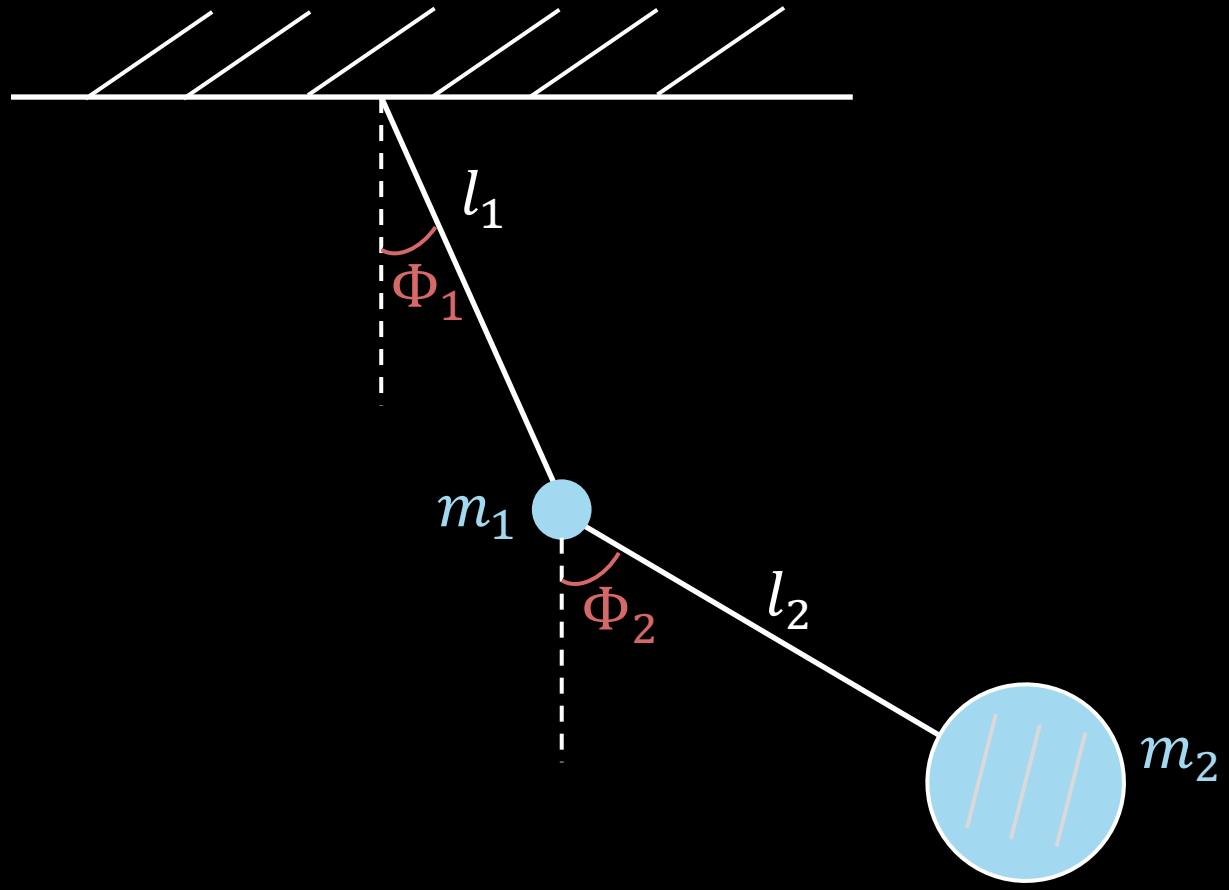
83

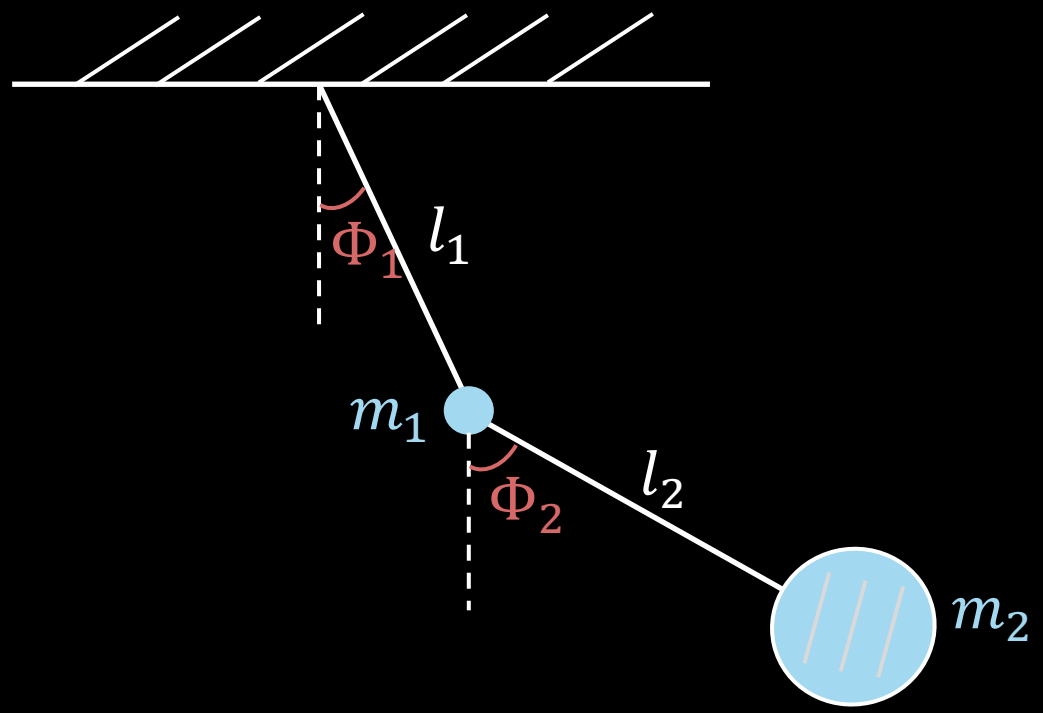
Influence du quadruple pendule sur la fonction de transfert

‘Le travail du système d'amortissement passif de LIGO est de tromper les masses de test [...] (miroirs) en leur donnant l'impression de flotter dans l'espace en utilisant un pendule à 4 étages [...]. Les [...] miroirs sont suspendues au bas de ce pendule à 4 segments par des fibres de silice fondue (verre) de ***0,4 mm*** *d'épaisseur.’*

California Institute of Technology




l_1
Φ_1
m_1
Φ_2
l_2
m_2



- $U = -m_1 g l_1 \cos(\Phi_1) - m_2 g[l_1 \cos(\Phi_1) + l_2 \cos(\Phi_2)]$
- $T = \frac{1}{2} m_1 l_1^2 \dot{\Phi}_1^2 + \frac{1}{2} m_2 \left[l^2{}_1 \dot{\Phi}^2{}_1 + l^2{}_1 \dot{\Phi}^2{}_1 + l^2{}_2 \dot{\Phi}^2{}_2 + 2 l_1 l_2 \dot{\Phi}_1 \dot{\Phi}_2 \right]$
- $\mathcal{L} = T - U$

  $= \frac{1}{2} m_1 l_1^2 \dot{\Phi}_1^2 + \frac{1}{2} m_2 \left[l^2{}_1 \dot{\Phi}^2{}_1 + l^2{}_1 \dot{\Phi}^2{}_1 + l^2{}_2 \dot{\Phi}^2{}_2 + 2 l_1 l_2 \dot{\Phi}_1 \dot{\Phi}_2 \right]$

  $+ m_1 g l_1 \cos(\Phi_1) + m_2 g (l_1 \cos(\Phi_1) + l_2 \cos(\Phi_2))$
- $\begin{cases} \frac{d}{dt}\left[\frac{\partial \mathcal{L}}{\partial \dot{\Phi}_1}\right] - \frac{\partial \mathcal{L}}{\partial \Phi_1} = 0 \\ \frac{d}{dt}\left[\frac{\partial \mathcal{L}}{\partial \dot{\Phi}_2}\right] - \frac{\partial \mathcal{L}}{\partial \Phi_2} = 0 \end{cases}$
- ⋮

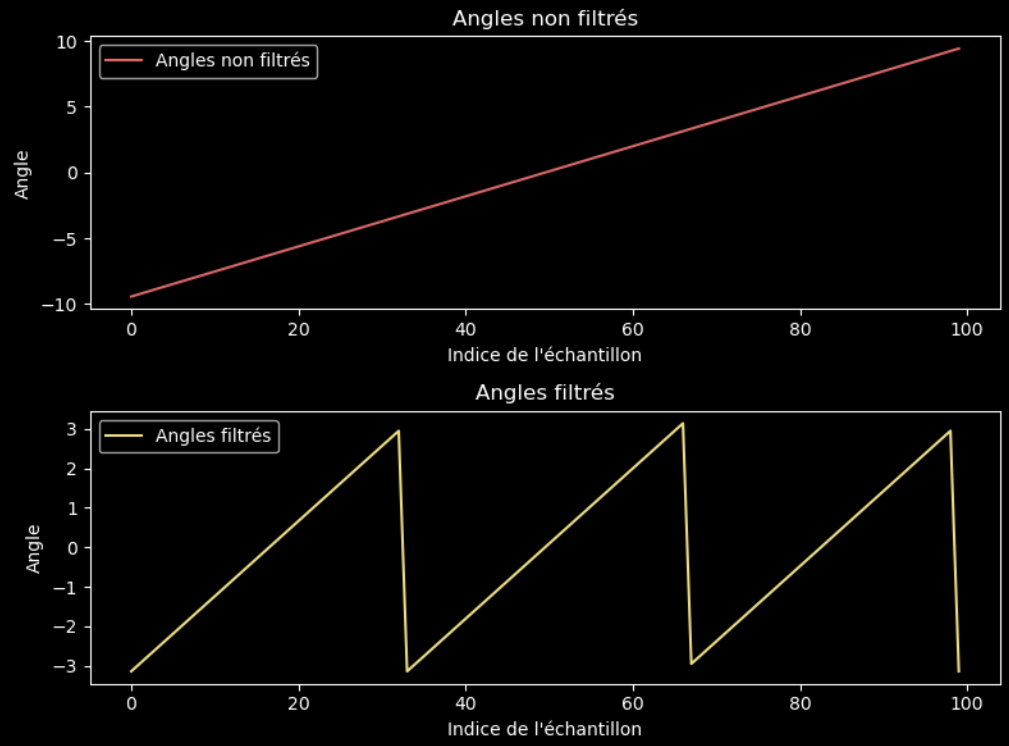
$$\begin{cases} 2\ddot{\Phi}_1 + \ddot{\Phi}_2 = -2\omega_0^2 \Phi_1 \\ \ddot{\Phi}_1 + \ddot{\Phi}_2 = -\omega^2{}_0 \Phi_2 \end{cases}$$

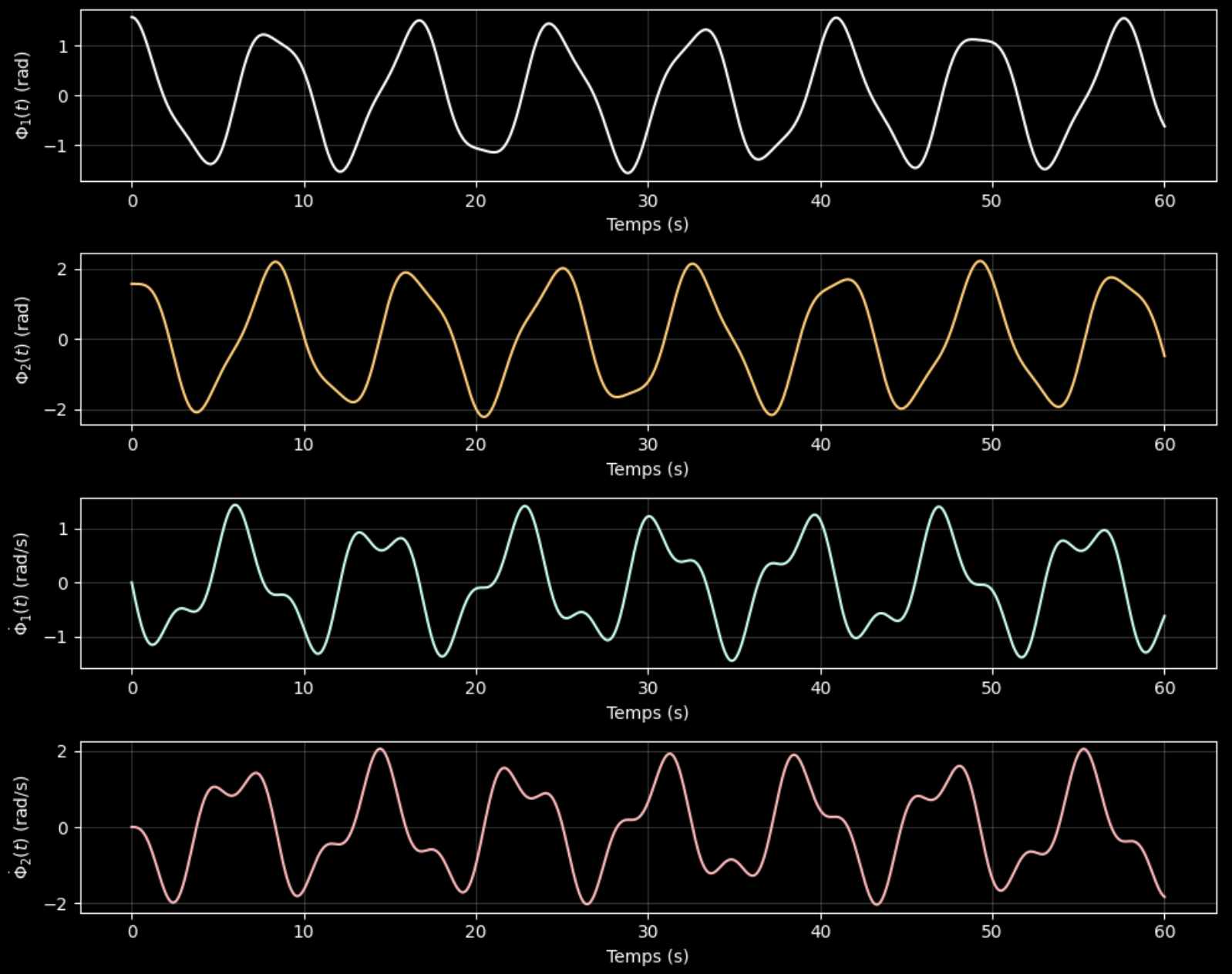
```python
def equation_of_motion_of_a_coupled_harmonic_oscillator(u, t, ω0):
    # On récupère les valeurs des angles et de leurs vitesses angulaires
    Φ1, Φ2, Φ1_dot, Φ2_dot = u

    # On calcule les dérivées
    dΦ1 = Φ1_dot
    dΦ2 = Φ2_dot

    dΦ1_dot = (-2 * ω0**2 * Φ1 + ω0**2 * Φ2)
    dΦ2_dot = 2*(ω0**2)*(Φ1-Φ2)

    u_dot = np.array([dΦ1, dΦ2, dΦ1_dot, dΦ2_dot])

    # On retourne les dérivées sous forme de tableau
    return u_dot
```

$$\begin{cases} \boldsymbol{d_1} = h \cdot f(t_n, y_n) \\ \boldsymbol{d_2} = h \cdot f\left(t_n + \frac{h}{2}, y_n + \frac{d_1}{2}\right) \\ \boldsymbol{d_3} = h \cdot f\left(t_n + \frac{h}{2}, y_n + \frac{d_2}{2}\right) \\ \boldsymbol{d_4} = h \cdot f(t_n + h, y_n + d_3) \end{cases}$$

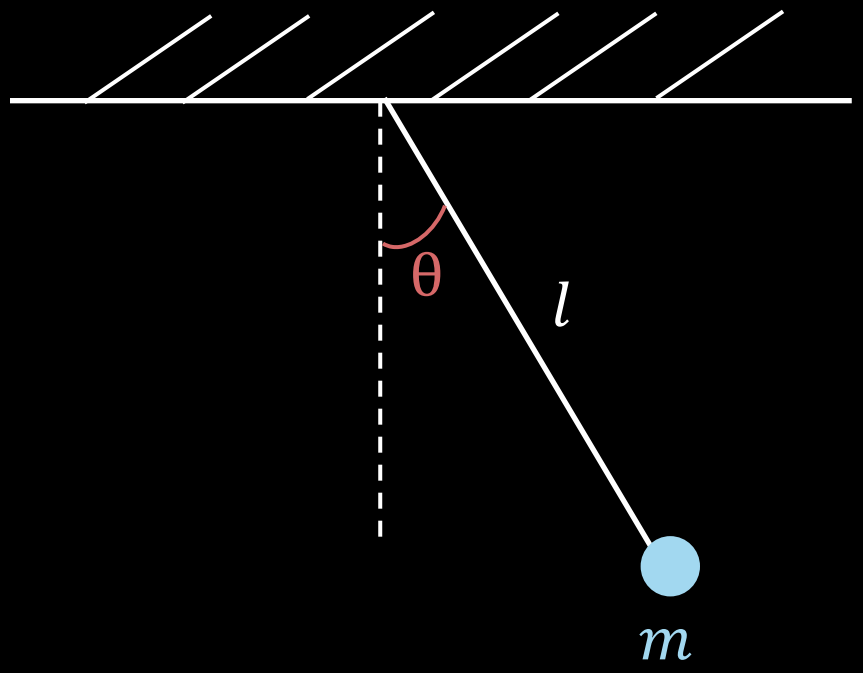
```python
def rk4_coupled(start, end, step, v_0, derivated, order, ω0):
    # Création du tableau temps
    interval = end - start                      # Intervalle
    num_points = int(interval / step) + 1       # Nombre d'éléments
    t = np.linspace(start, end, num_points)     # Tableau temps t

    # Initialisation du tableau v
    v = np.empty((order, num_points))

    # Condition initiale (toutes les lignes à la colonne 0)
    v[:, 0] = v_0

    # On calcule les pentes en bouclant sur le nombre de points
    for i in range(num_points - 1):
        d1 = derivated(v[:, i], t[i], ω0)
        d2 = derivated(v[:, i] + step / 2 * d1, t[i] + step / 2, ω0)
        d3 = derivated(v[:, i] + step / 2 * d2, t[i] + step / 2, ω0)
        d4 = derivated(v[:, i] + step * d3, t[i] + step, ω0)
        v[:, i + 1] = v[:, i] + step / 6 * (d1 + 2 * d2 + 2 * d3 + d4)

    # On retourne le tableau des t et le tableau v qui décrit l'état du système
    return t, v
```

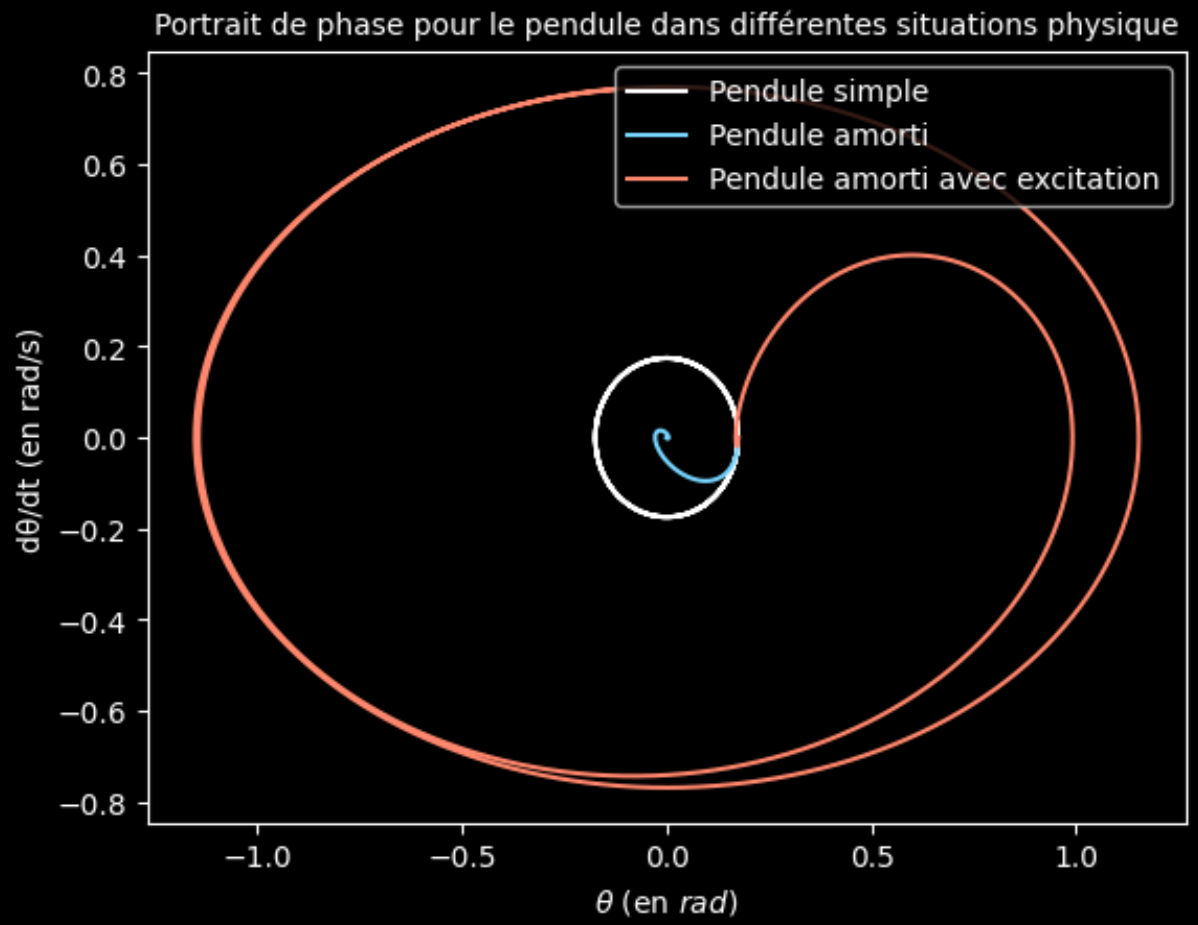
```python
def filter_angle_values(table_to_filter):

    final_table = np.zeros_like(table_to_filter)

    for i, angle in enumerate(table_to_filter):
        # θ>π ou θ<-π  -> θ = (θ+π)[2π] - π
        if angle > np.pi or angle < -np.pi:
            final_table[i] = (angle + np.pi) % (2 * np.pi) - np.pi
        # Sinon, θ est bien compris entre -pi et pi et l'on ne le modifie pas
        else:
            final_table[i] = angle

    return final_table
```

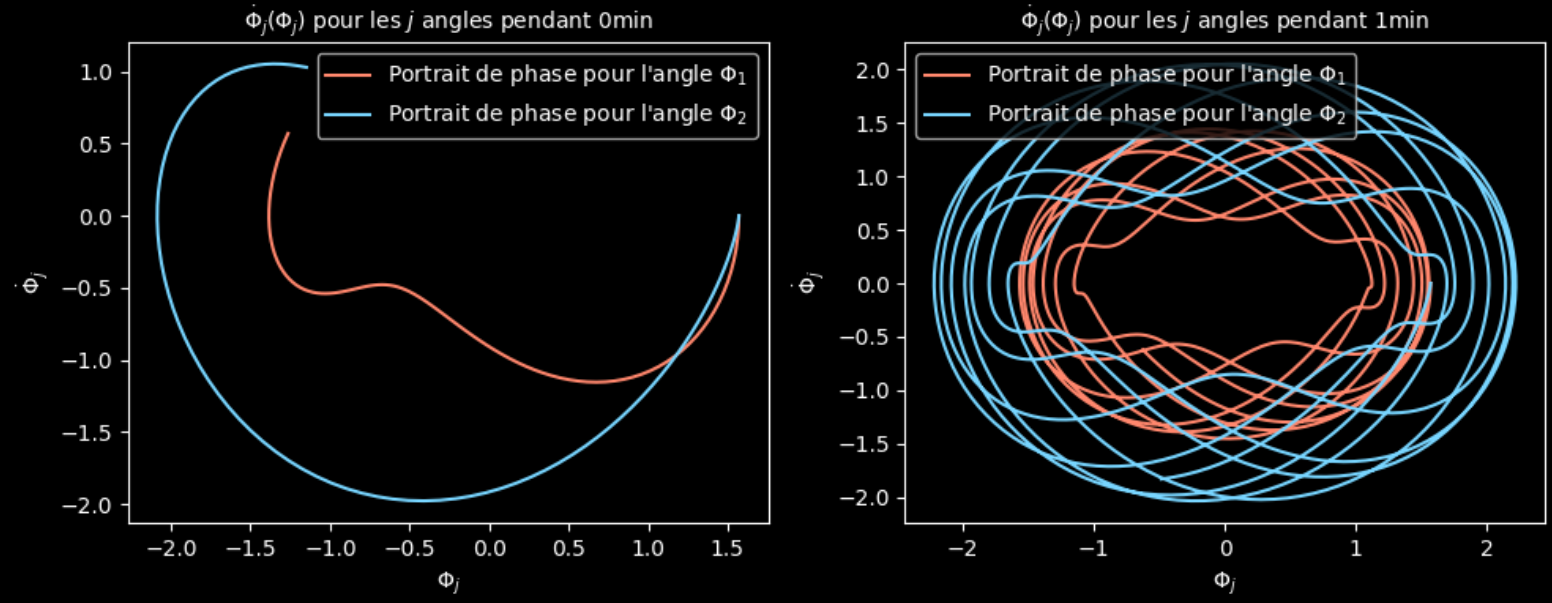



$\Phi_1(t)$ (rad)
Temps (s)
$\Phi_2(t)$ (rad)
Temps (s)
$\dot{\Phi}_1(t)$ (rad/s)
Temps (s)
$\dot{\Phi}_2(t)$ (rad/s)
Temps (s)

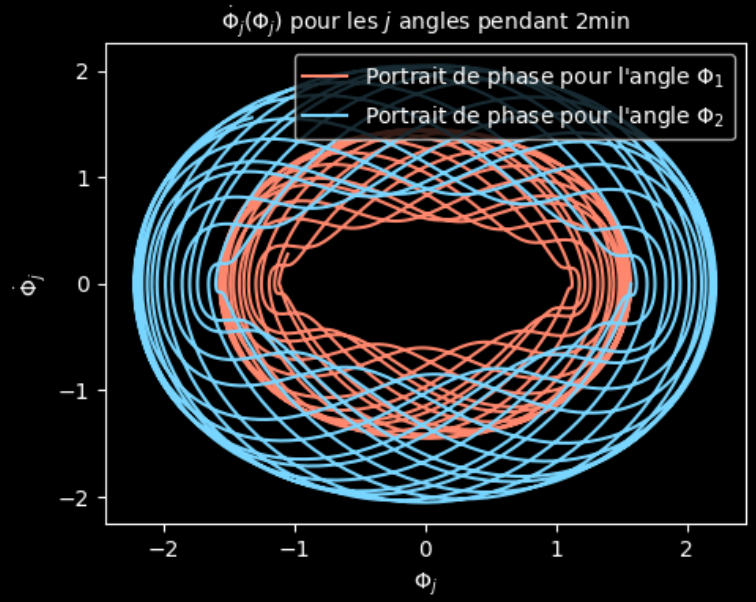
$$\frac{d^2\theta}{dt^2} + q\frac{d\theta}{dt} + \Omega^2 \sin\theta = 0$$

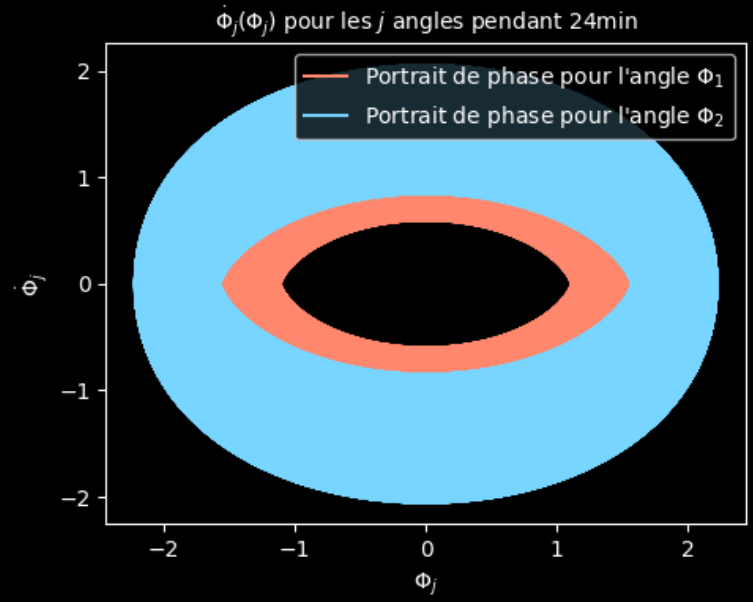


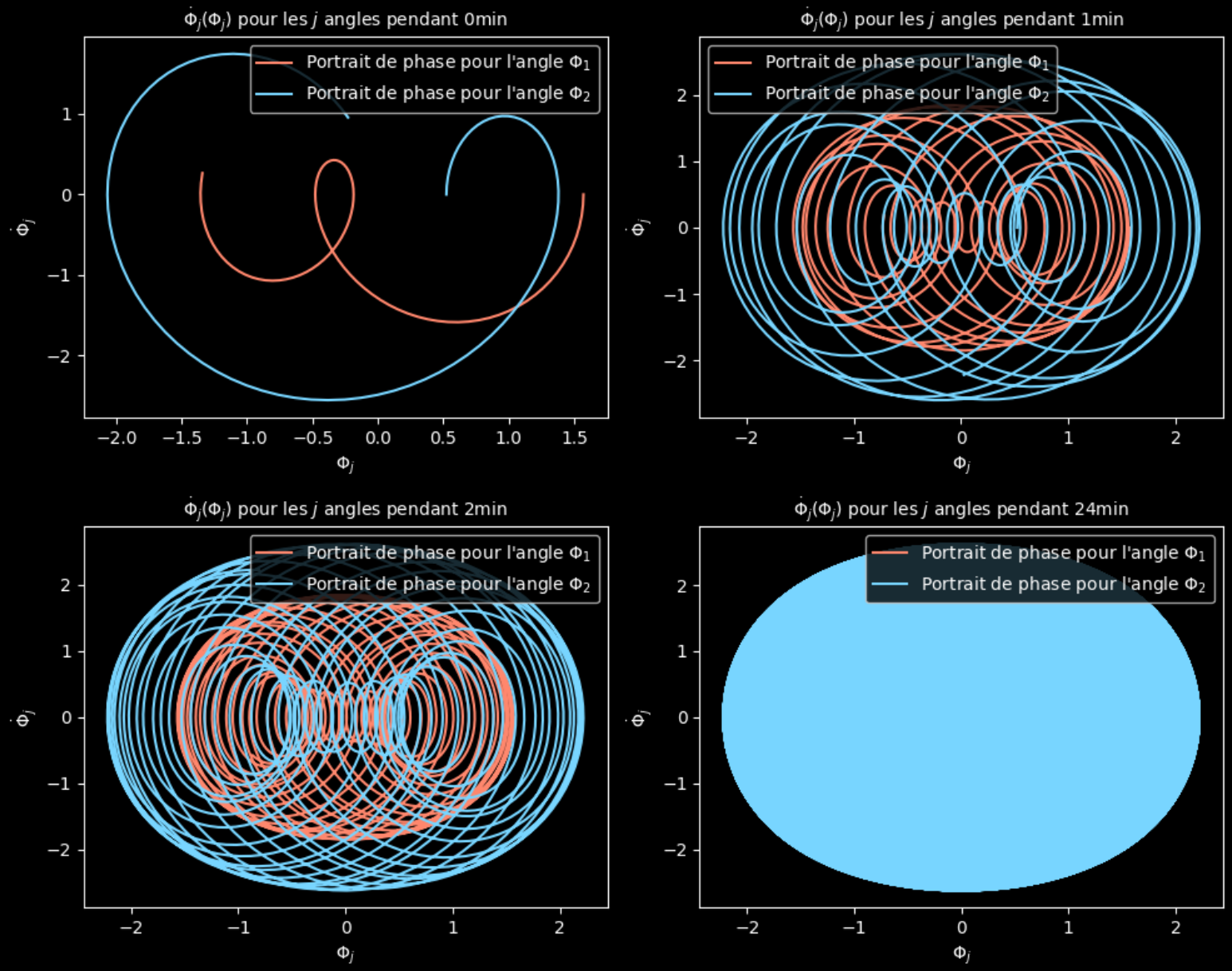



$\dot{\Phi}_j(\Phi_j)$ pour les j angles pendant 0min
Portrait de phase pour l'angle Φ_1
Portrait de phase pour l'angle Φ_2
$\dot{\Phi}_j$
Φ_j

$\dot{\Phi}_j(\Phi_j)$ pour les j angles pendant 1min
Portrait de phase pour l'angle Φ_1
Portrait de phase pour l'angle Φ_2
$\dot{\Phi}_j$
Φ_j


$\dot{\Phi}_j(\Phi_j)$ pour les j angles pendant 2min
Portrait de phase pour l'angle Φ_1
Portrait de phase pour l'angle Φ_2
$\dot{\Phi}_j$
Φ_j


$\dot{\Phi}_j(\Phi_j)$ pour les j angles pendant 24min
Portrait de phase pour l'angle Φ_1
Portrait de phase pour l'angle Φ_2
$\dot{\Phi}_j$
Φ_j

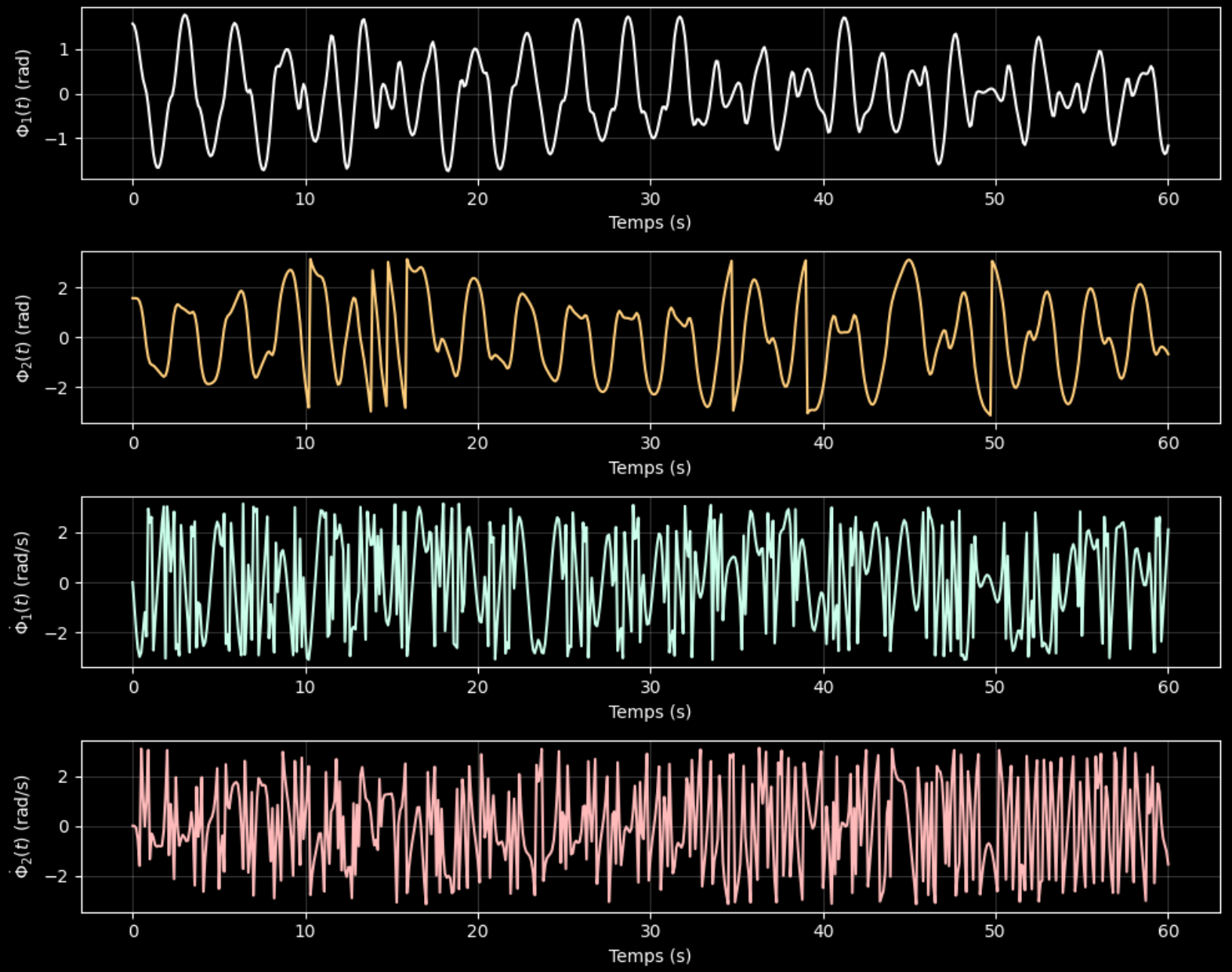

$\dot{\Phi}_j(\Phi_j)$ pour les j angles pendant 0min
$\dot{\Phi}_j(\Phi_j)$ pour les j angles pendant 1min
$\dot{\Phi}_j(\Phi_j)$ pour les j angles pendant 2min
$\dot{\Phi}_j(\Phi_j)$ pour les j angles pendant 24min
Portrait de phase pour l'angle Φ_1
Portrait de phase pour l'angle Φ_2
$\dot{\Phi}_j$
Φ_j

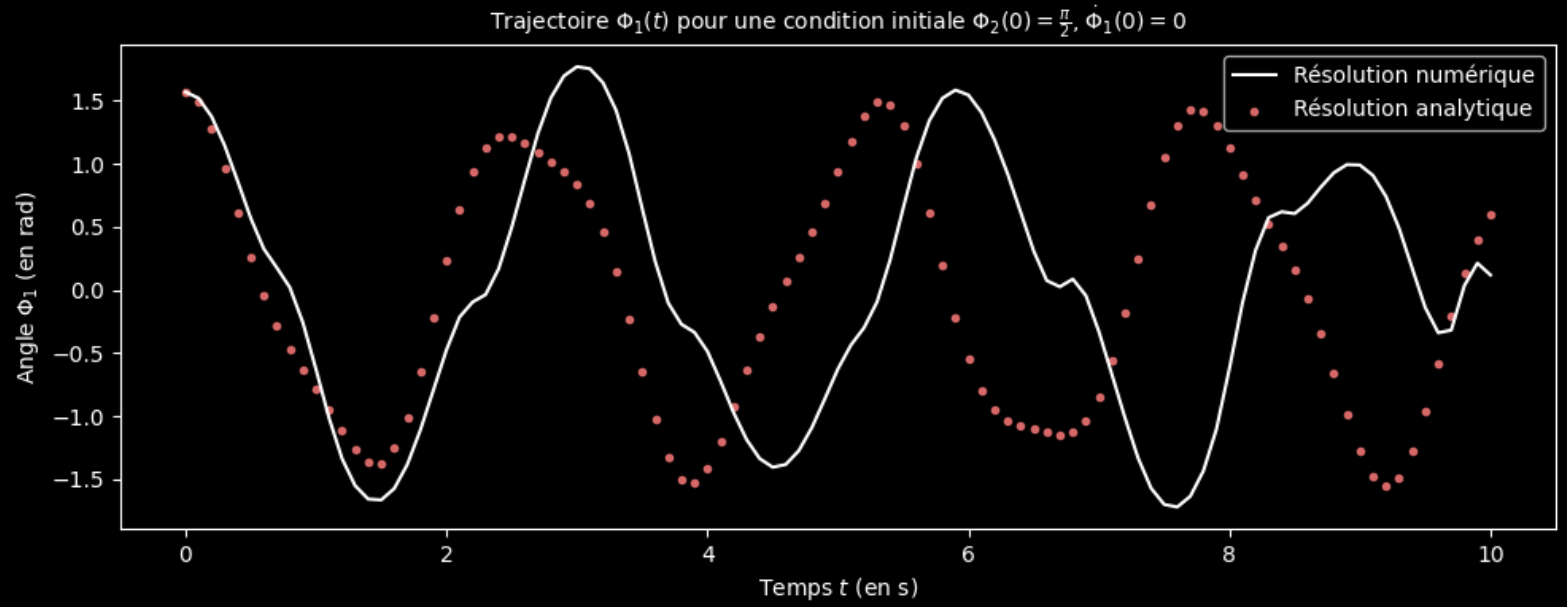
$$\begin{cases} [m_1 + m_2] l_1 \ddot{\Phi}_1 + m_2 l_2 \ddot{\Phi}_2 \cos(\Phi_1 - \Phi_2) + m_2 l_2 \dot{\Phi}_2^{\,2} \sin(\Phi_1 - \Phi_2) + [m_1 + m_2] g \sin(\Phi_1) = 0 \\ l_1 \ddot{\Phi}_1 \cos(\Phi_1 - \Phi_2) + l_2 \ddot{\Phi}_2 - l_1 \dot{\Phi}_1^{\,2} \sin(\Phi_1 - \Phi_2) + g \sin \Phi_2 = 0 \end{cases}$$

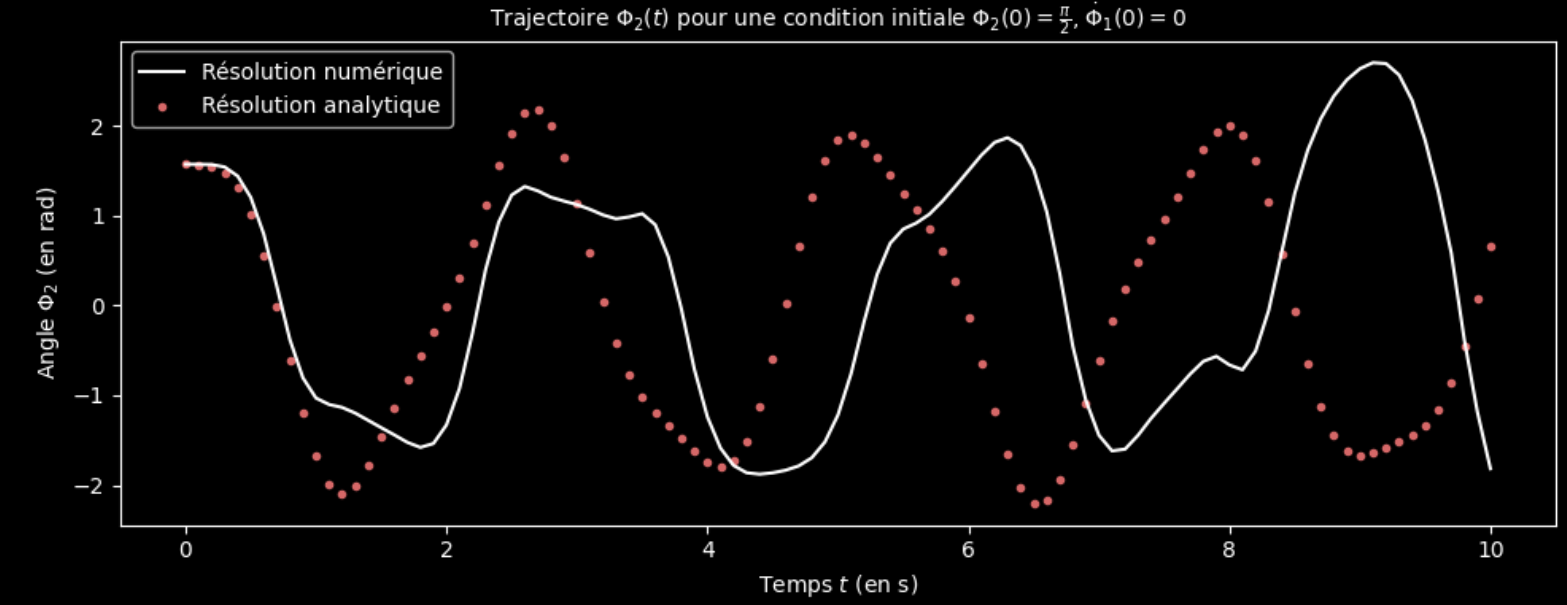
```python
def differential_equation_system_without_the_assumption_of_small_oscillations(u, t, m1, m2, l1, l2, g):

    # On récupère les valeurs des angles et de leurs vitesses angulaires
    Φ1, Φ2, angular_velocity_1, angular_velocity_2 = u

    # Pour alléger les lignes de codes qui viennent, on stocke les termes en cosinus et en sinus dans des variables
    cosine_term = np.cos(Φ1-Φ2)
    sine_term = np.sin(Φ1-Φ2)

    Φ1_dot = angular_velocity_1
    Φ2_dot = angular_velocity_2

    # Equation 1
    angular_velocity_1_dot = (m2*g*np.sin(Φ2)*cosine_term - m2*sine_term*
(l1*angular_velocity_1**2*cosine_term +
                                                                    l2*angular_velocity_2**2)
                              - (m1+m2)*g*np.sin(Φ1)) / l1 / (m1 + m2*sine_term**2)

    # Equation 2
    angular_velocity_2_dot = ((m1+m2)*(l1*angular_velocity_1**2*sine_term - g*np.sin(Φ2) +
g*np.sin(Φ1)*cosine_term)
                              + m2*l2*angular_velocity_2**2*sine_term*cosine_term) / l2 / (m1 +
m2*sine_term**2)

    return [Φ1_dot, Φ2_dot, angular_velocity_1_dot, angular_velocity_2_dot]
```

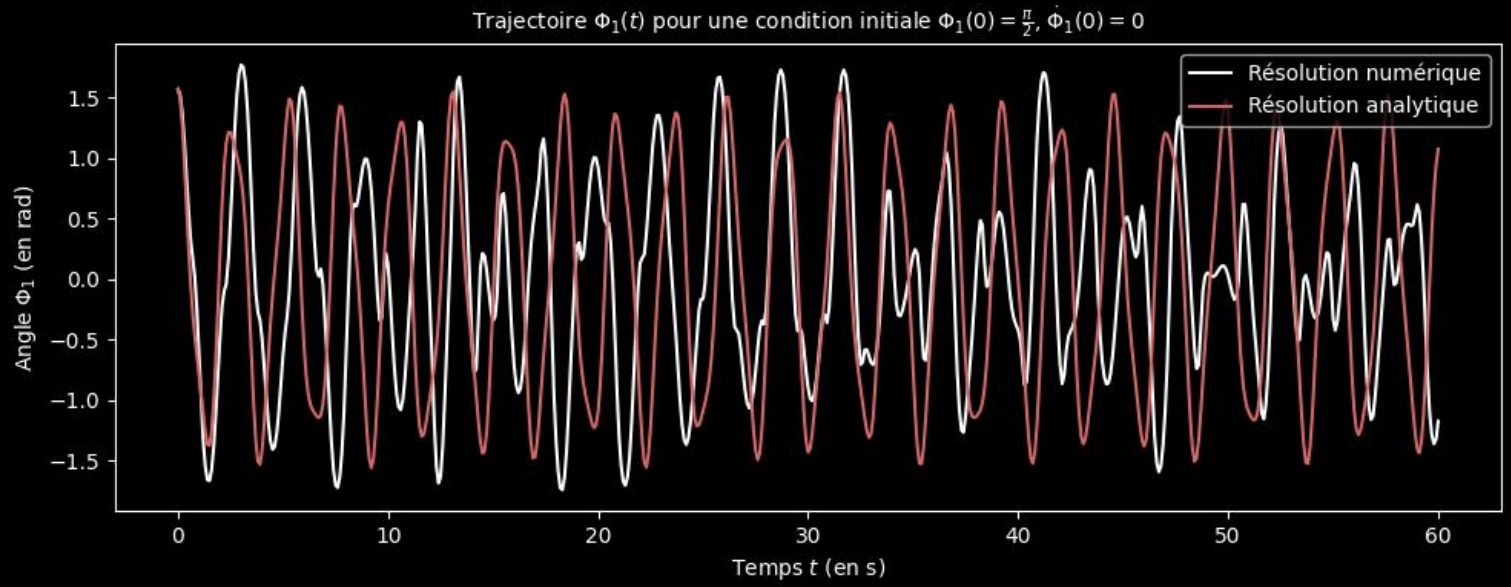
- ✓ Ajout de g, l_1, l_2, m_1, et m_2 en entrée

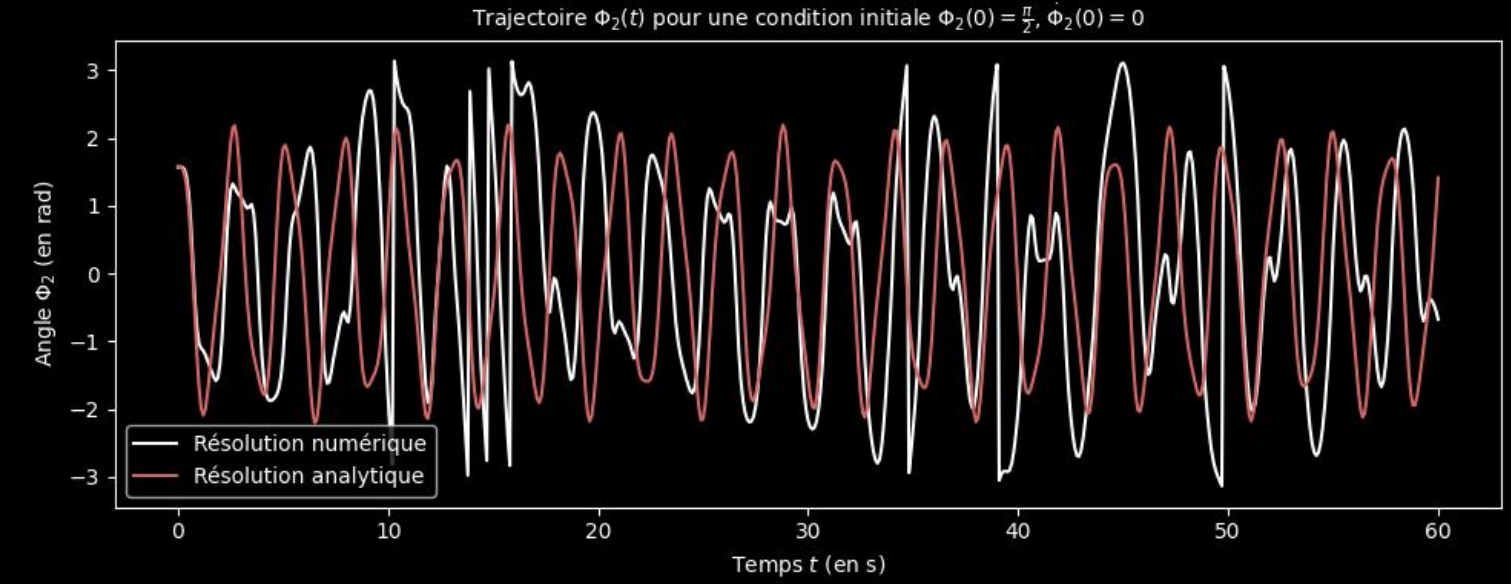
```python
def rk4_coupled_without_the_assumption_of_small_oscillations(start, end, step, v_0,
                                                             derivated,
                                                             order, m1, m2, l1, l2, g):

    # Création du tableau temps
    interval = end - start                      # Intervalle
    num_points = int(interval / step) + 1       # Nombre d'éléments
    t = np.linspace(start, end, num_points)     # Tableau temps t

    # Initialisation du tableau v
    v = np.empty((order, num_points))

    # Condition initiale (toutes les lignes à la colonne 0)
    v[:, 0] = v_0

    # Boucle for
    for i in range(num_points - 1):
        d1 = np.array(derivated(v[:, i], t[i], m1, m2, l1, l2, g))
        d2 = np.array(derivated(v[:, i] + step / 2 * d1, t[i] + step / 2, m1, m2, l1, l2, g))
        d3 = np.array(derivated(v[:, i] + step / 2 * d2, t[i] + step / 2, m1, m2, l1, l2, g))
        d4 = np.array(derivated(v[:, i] + step * d3, t[i] + step, m1, m2, l1, l2, g))
        v[:, i + 1] = v[:, i] + step / 6 * (d1 + 2 * d2 + 2 * d3 + d4)

    # Argument de sortie
    return t, v
```

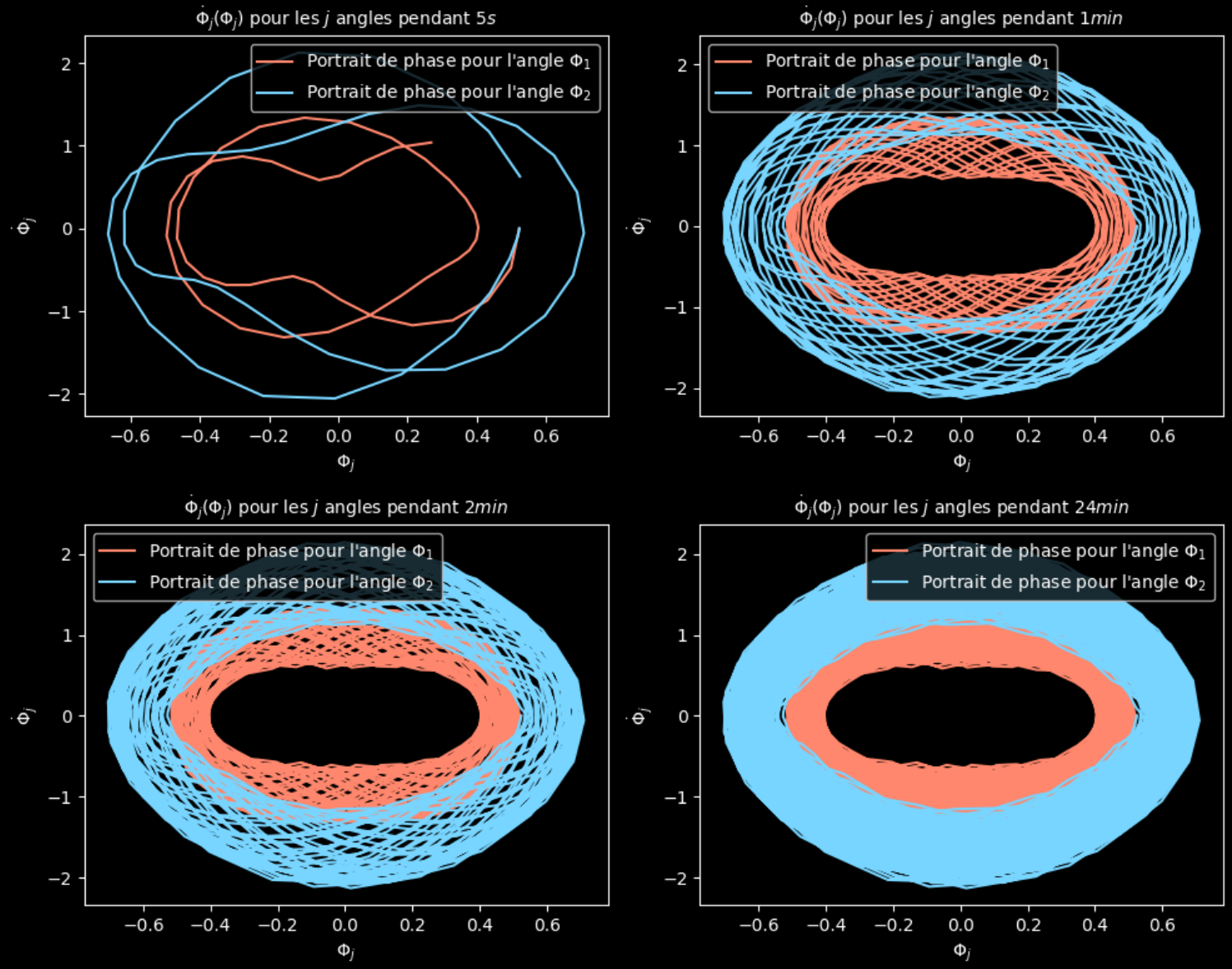

$\Phi_1(t)$ (rad)
Temps (s)
$\Phi_2(t)$ (rad)
Temps (s)
$\dot{\Phi}_1(t)$ (rad/s)
Temps (s)
$\dot{\Phi}_2(t)$ (rad/s)
Temps (s)

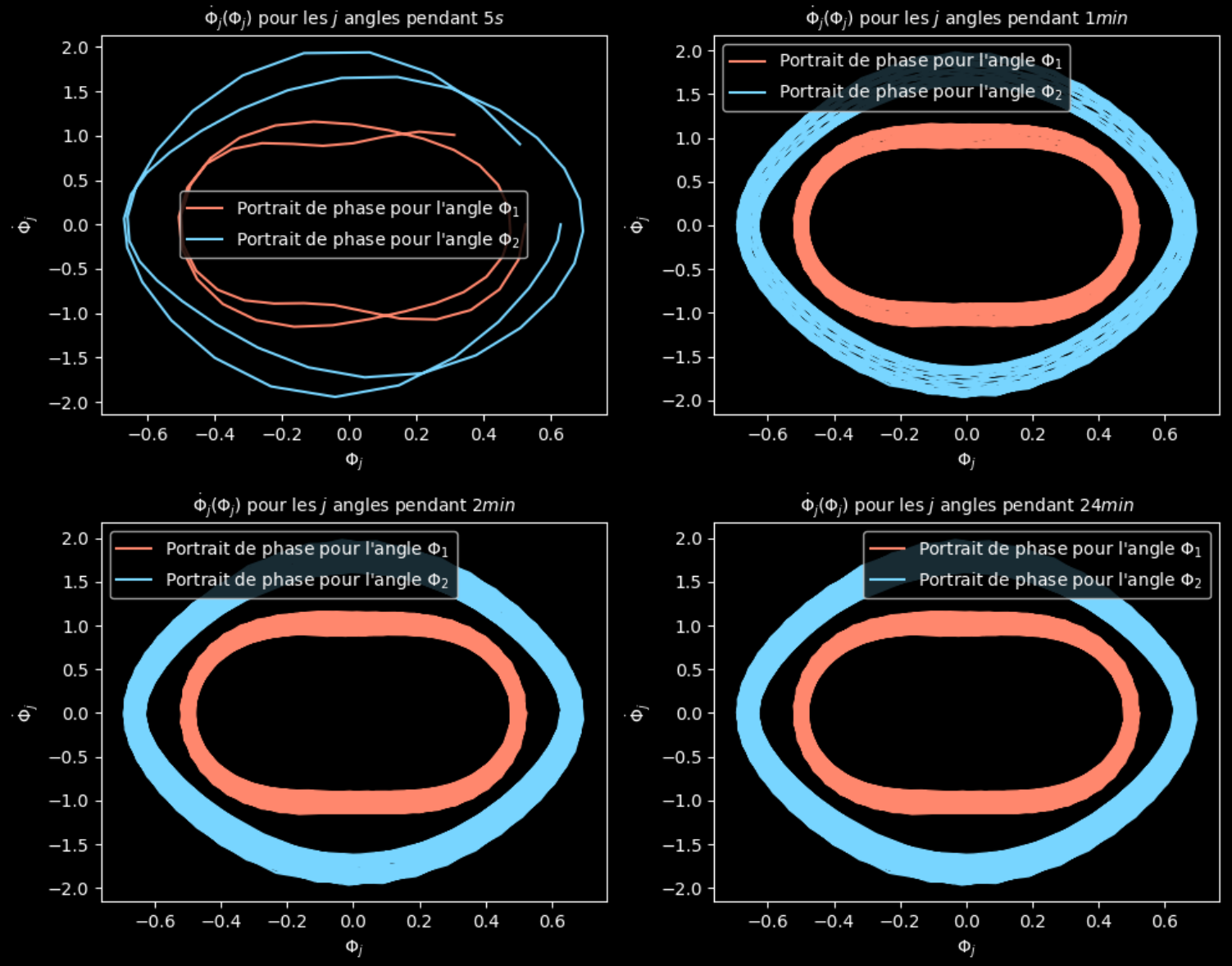

Trajectoire $\Phi_1(t)$ pour une condition initiale $\Phi_2(0) = \frac{\pi}{2}$, $\dot{\Phi}_1(0) = 0$
Résolution numérique
Résolution analytique
Angle Φ_1 (en rad)
Temps t (en s)

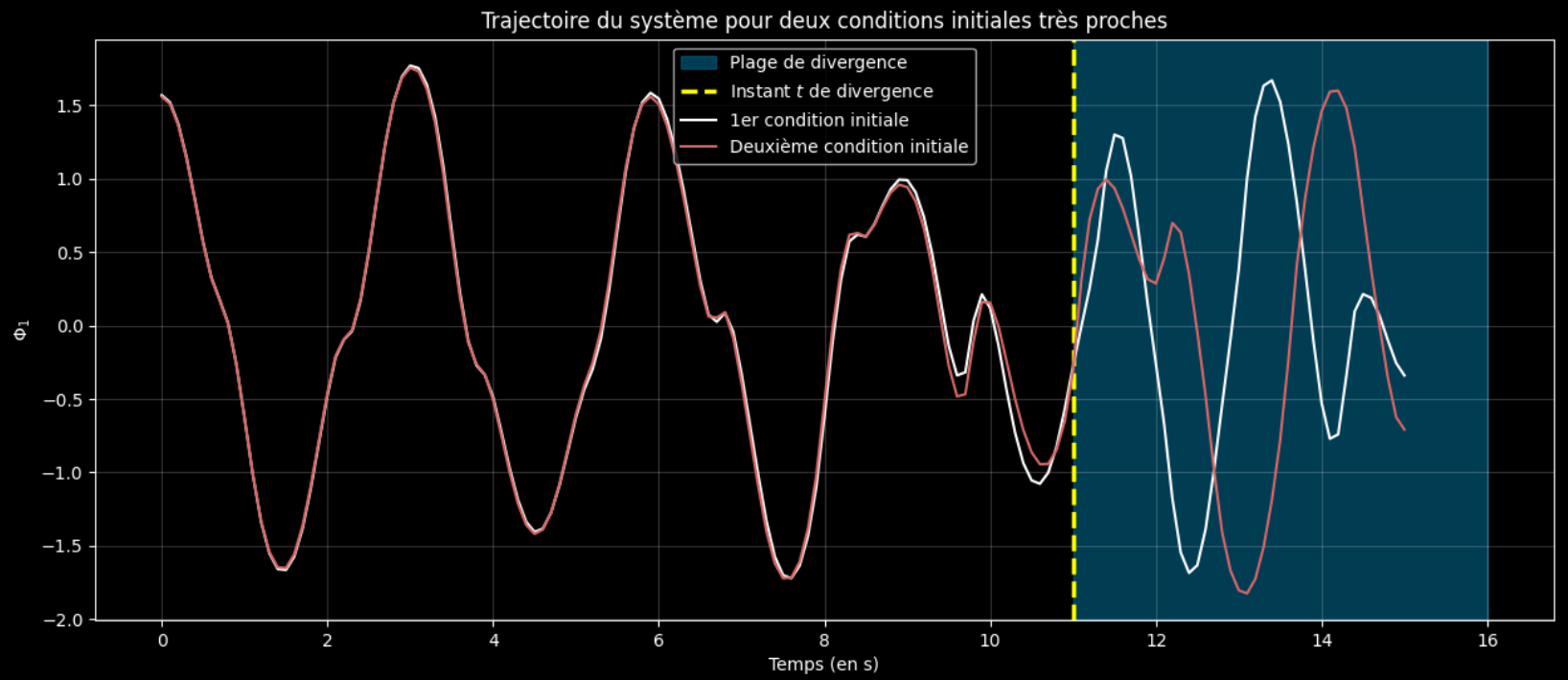

Trajectoire $\Phi_2(t)$ pour une condition initiale $\Phi_2(0) = \frac{\pi}{2}$, $\dot{\Phi}_1(0) = 0$
Résolution numérique
Résolution analytique
Angle Φ_2 (en rad)
Temps t (en s)


Trajectoire $\Phi_1(t)$ pour une condition initiale $\Phi_1(0) = \frac{\pi}{2}$, $\dot{\Phi}_1(0) = 0$
Résolution numérique
Résolution analytique
Angle Φ_1 (en rad)
Temps t (en s)

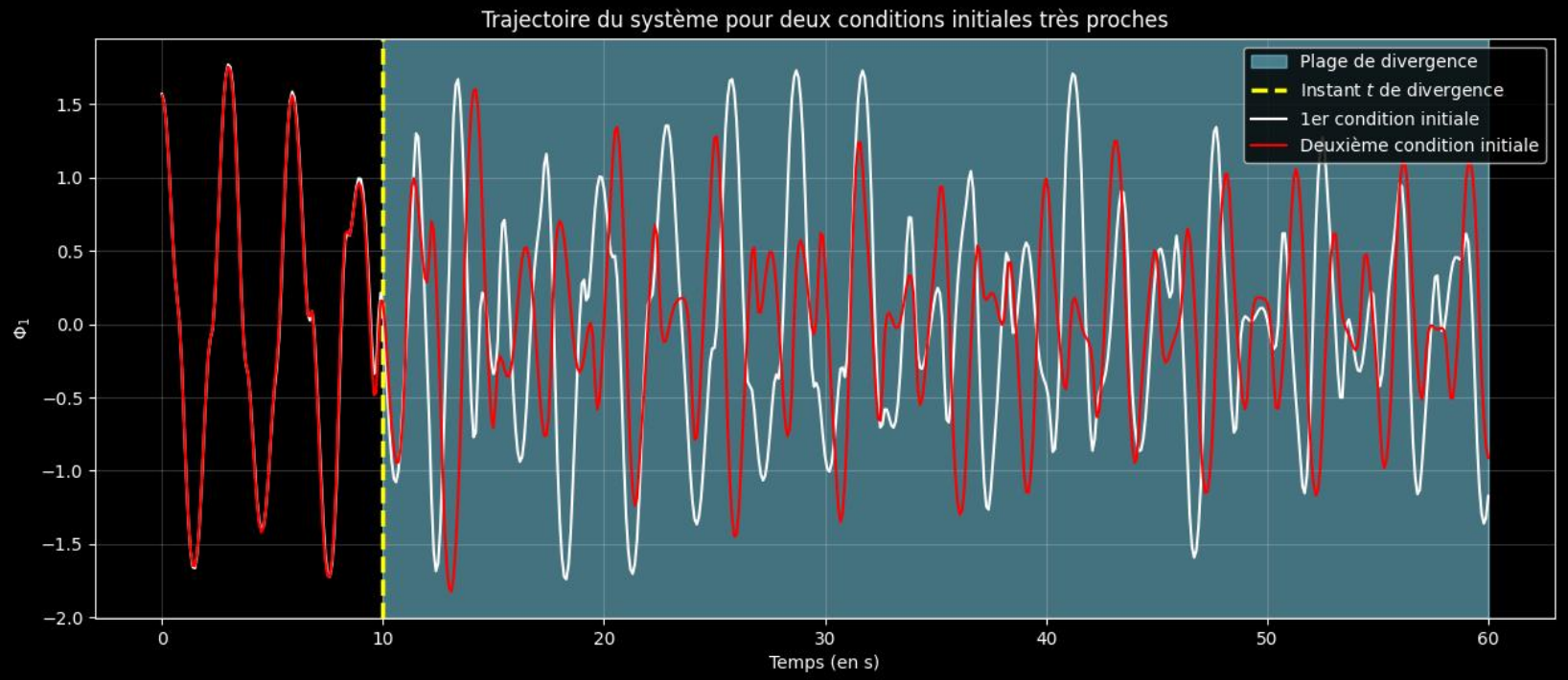

Trajectoire $\Phi_2(t)$ pour une condition initiale $\Phi_2(0) = \frac{\pi}{2}$, $\dot{\Phi}_2(0) = 0$
Résolution numérique
Résolution analytique
Angle Φ_2 (en rad)
Temps t (en s)


$\dot{\Phi}_j(\Phi_j)$ pour les j angles pendant $5s$
$\dot{\Phi}_j(\Phi_j)$ pour les j angles pendant $1min$
$\dot{\Phi}_j(\Phi_j)$ pour les j angles pendant $2min$
$\dot{\Phi}_j(\Phi_j)$ pour les j angles pendant $24min$
Portrait de phase pour l'angle Φ_1
Portrait de phase pour l'angle Φ_2
$\dot{\Phi}_j$
Φ_j

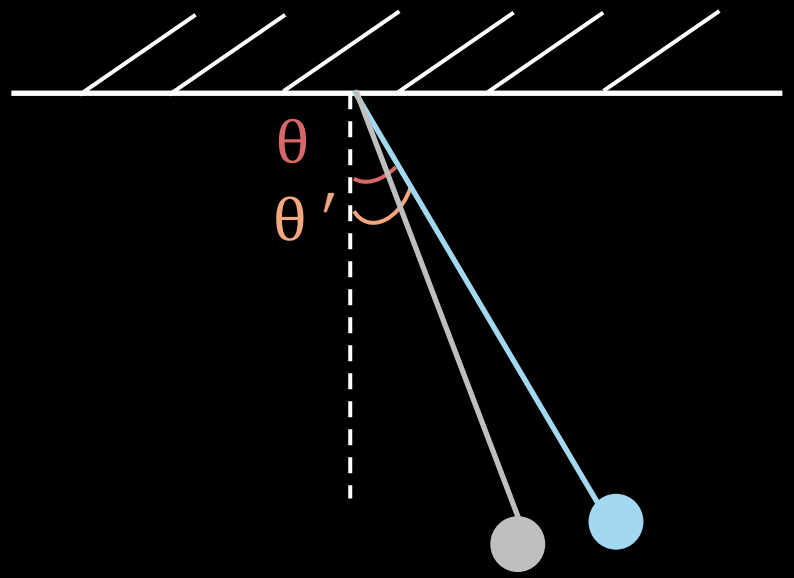

$\dot{\Phi}_j(\Phi_j)$ pour les j angles pendant $5s$
$\dot{\Phi}_j(\Phi_j)$ pour les j angles pendant $1min$
$\dot{\Phi}_j(\Phi_j)$ pour les j angles pendant $2min$
$\dot{\Phi}_j(\Phi_j)$ pour les j angles pendant $24min$
Portrait de phase pour l'angle Φ_1
Portrait de phase pour l'angle Φ_2
$\dot{\Phi}_j$
Φ_j
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6



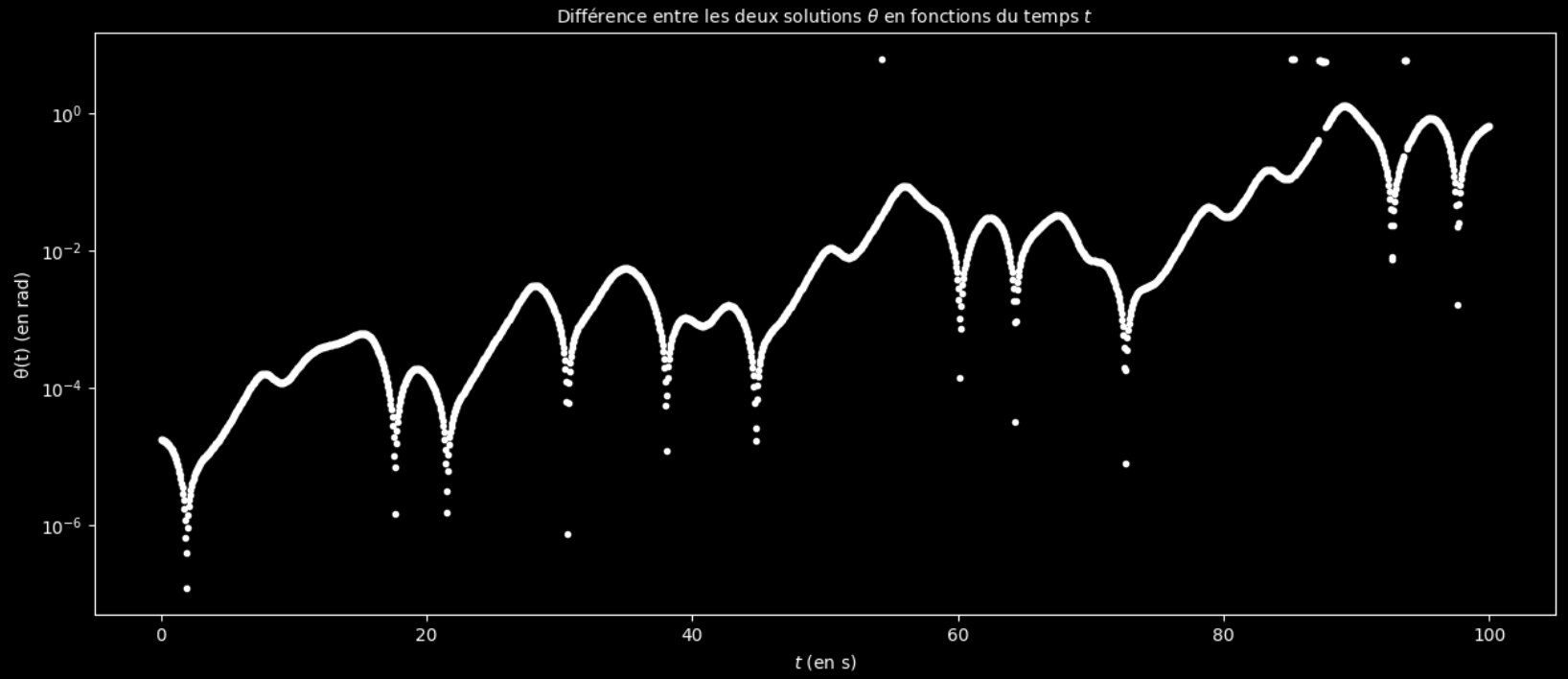
- On peut représenter la trajectoire pour deux conditions initiales très proches
- Informe à quelle moment les trajectoires ont diverger
- N'informe pas 'à quelle intensité' elle diverge

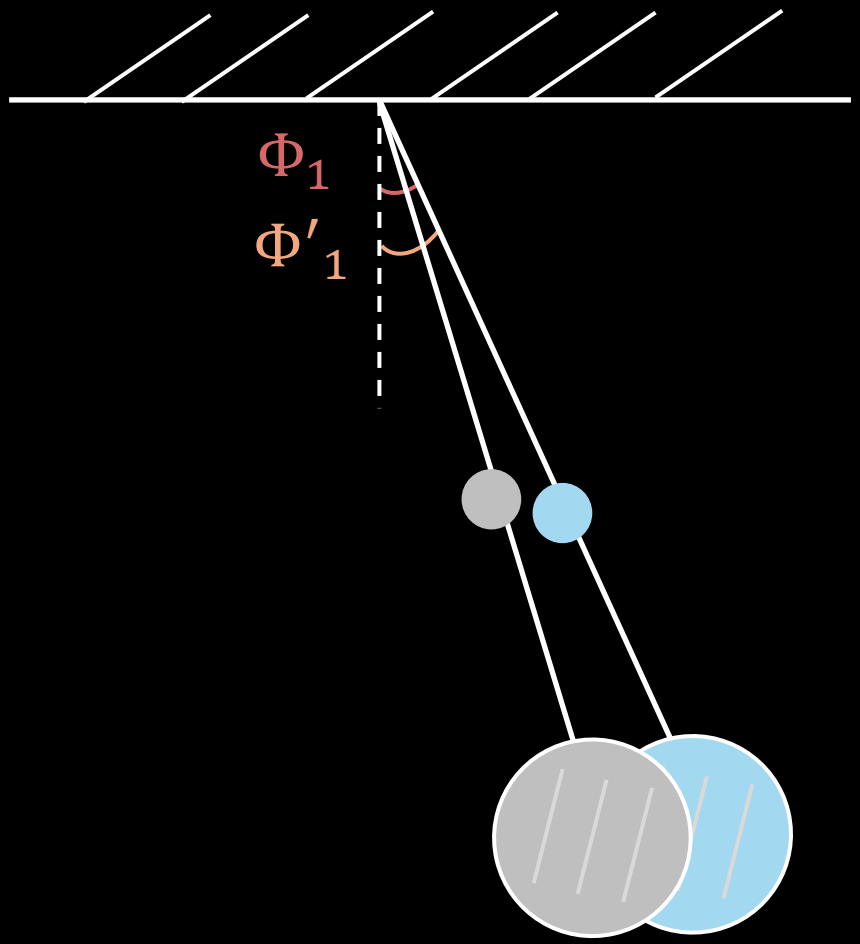


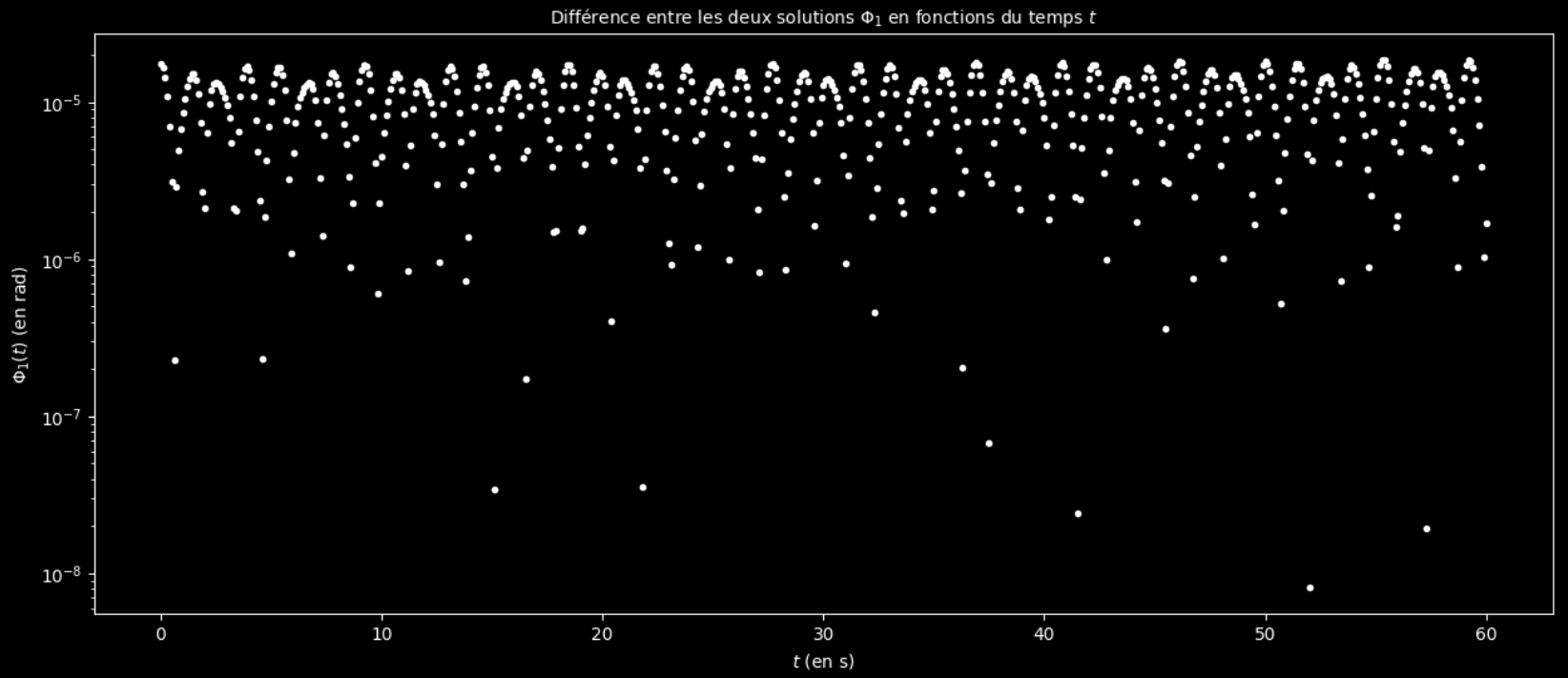
- On peut représenter la trajectoire pour deux conditions initiales très proches
- Informe à quelle moment les trajectoires ont diverger
- N'informe pas 'à quelle intensité' elle diverge



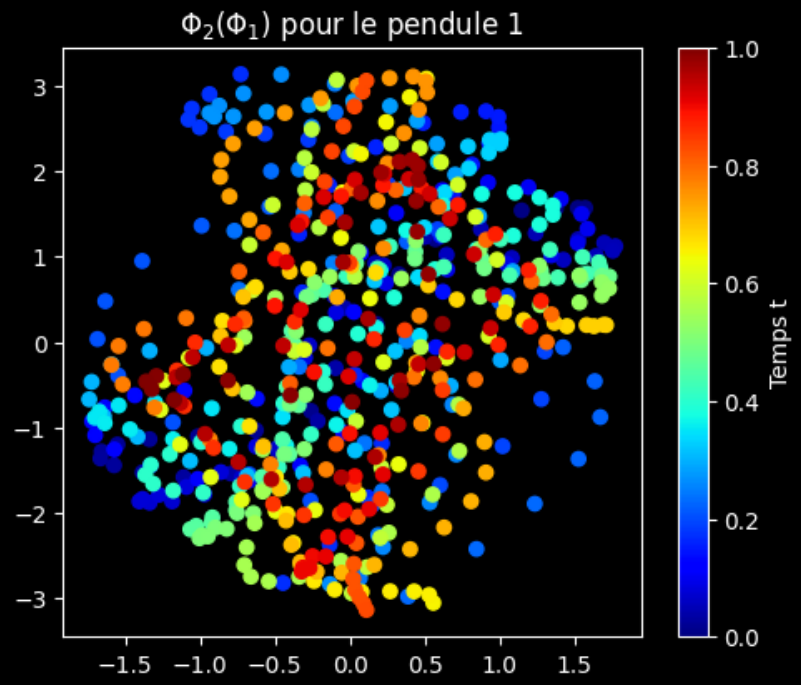
$$|\theta' - \theta| = f(t)$$

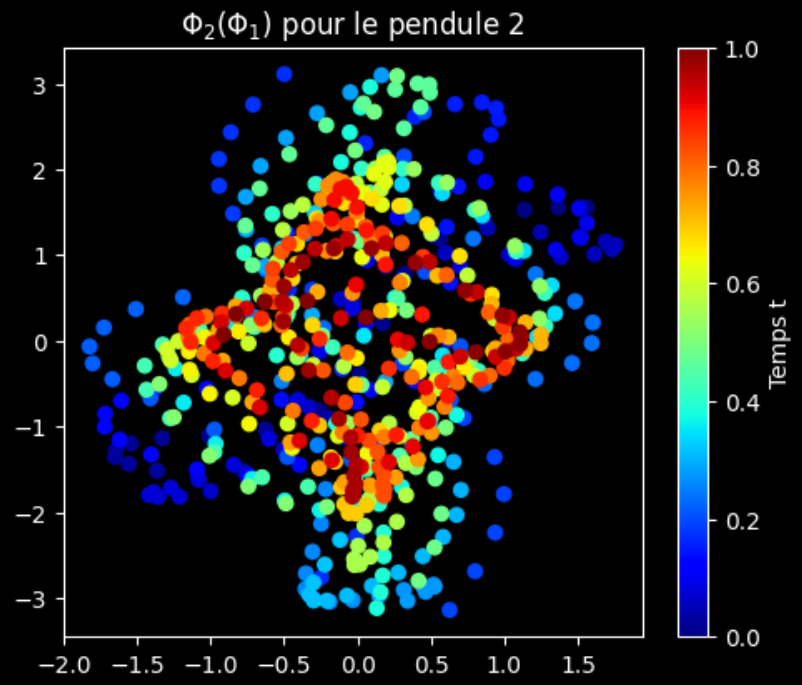



Φ_1
Φ'_1


Différence entre les deux solutions Φ_1 en fonctions du temps t
10^{-5}
10^{-6}
10^{-7}
10^{-8}
$\Phi_1(t)$ (en rad)
0
10
20
30
40
50
60
t (en s)

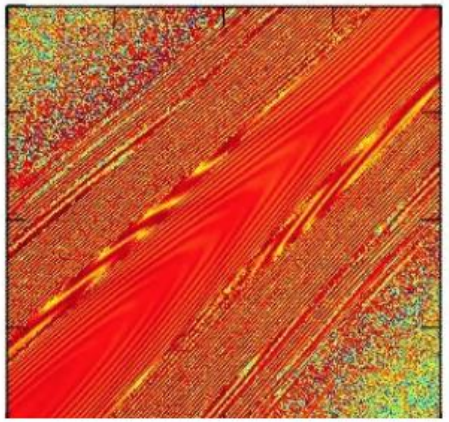
$$|\Phi'_1 - \Phi_1| = f(t)$$





- Les points de couleurs proches du bleu -> début de la stimulation
- Les points de couleurs proches du rouge -> fin de la stimulation
- Peu informatif

Le pendule double, un système chaotique



Noémie Jaquier Ma2-3

Travail de maturité 2009-2010
Gymnase intercantonal de la Broye

Supervisé par M. E. Jaquet

Noémie Jaquier, 2009-2010

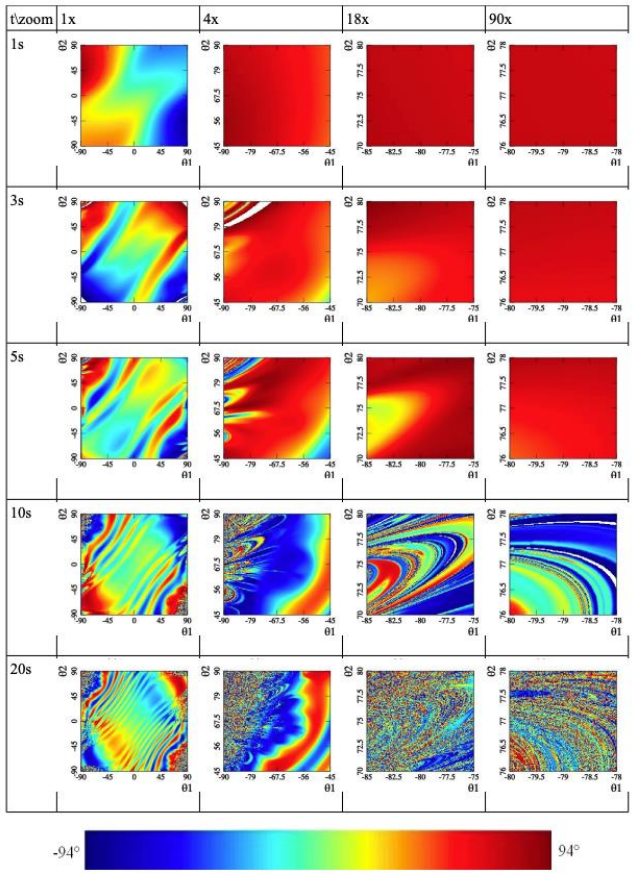


Figure 12: Représentation de θ_1 pour différents temps en fonction des paramètres initiaux

21

Analyse qualitative du chaos

En répétant l'opération avec une grande quantité de sets de conditions initiales - dans l'idéal il en faudrait une infinité - il est possible d'approcher l'exposant maximum de chaque temps t et ainsi de représenter l'exposant de Lyapunov maximal sur l'ensemble du système.

Par le biais d'un programme en C[7], 10'000 sets de conditions initiales ont été tiré au hasard et l'exposant de Lyapunov a été calculé pour chacun d'entre eux, pour un temps allant de 0 à 12 secondes. L'exposant de Lyapunov global[8] du système est représenté sur la figure 16. Ce graphique montre le caractère chaotique du pendule double, de par la valeur de l'exposant égale à la pente, constamment positive.

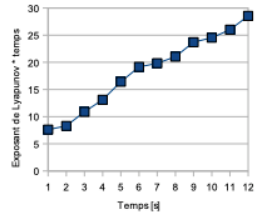


Figure 16: Représentation de l'exposant de Lyapunov (pente) en fonction du temps

On s'intéresse ensuite à l'exposant de Lyapunov en fonction de l'énergie globale du système. En effet, comme discuté au chapitre 6, la cinématique possible est fortement dépendante de l'énergie mécanique totale du pendule double.

La figure 17 représente ainsi la valeur de l'exposant de Lyapunov de chaque set de conditions initiales en fonction de l'énergie mécanique totale du système au temps t=20s. On remarque alors nettement une dépendance de la valeur de l'exposant à l'énergie.

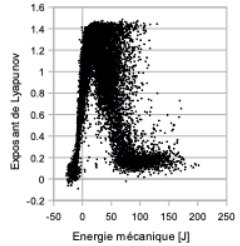
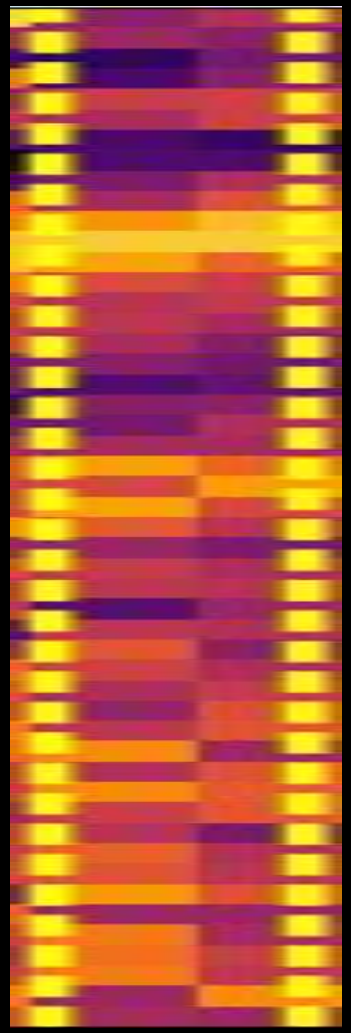


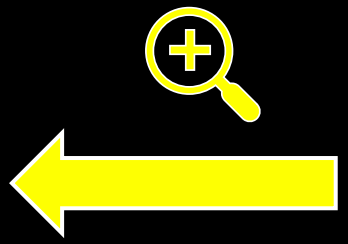
Figure 17 : Évaluation de l'exposant de Lyapunov en fonction de l'énergie mécanique totale après 20 s

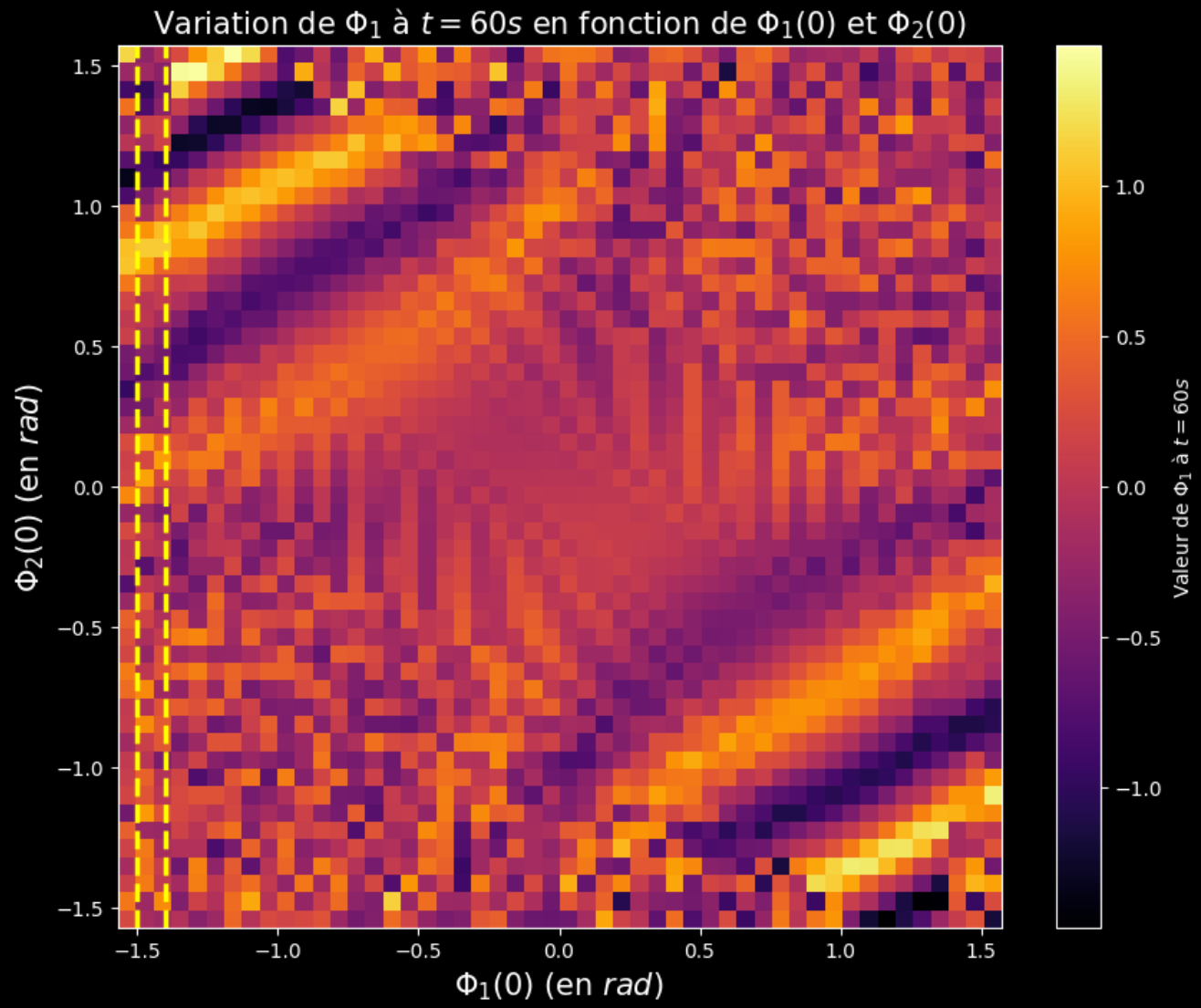
7 cf. Annexe 5
8 cf. Annexe 6

26

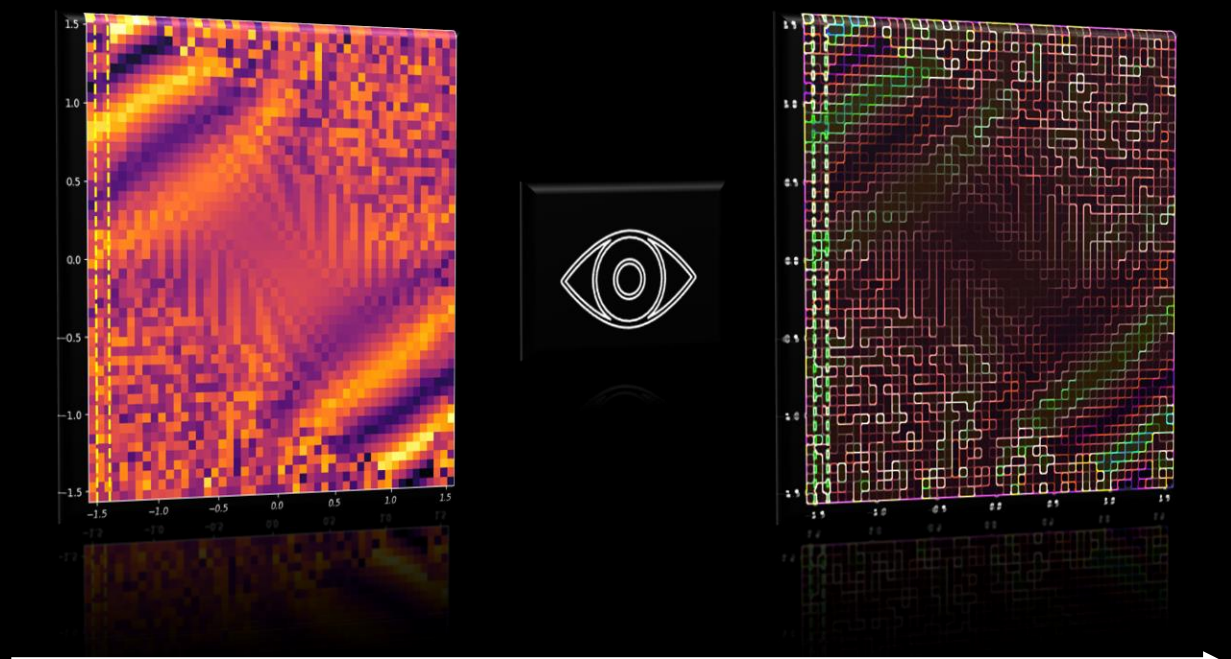
Analyse quantitative du chaos






Variation de Φ_1 à $t = 60s$ en fonction de $\Phi_1(0)$ et $\Phi_2(0)$
$\Phi_2(0)$ (en *rad*)
$\Phi_1(0)$ (en *rad*)
Valeur de Φ_1 à $t = 60s$
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5

```python
# Définition des bornes et du nombre de valeurs
num_values = 120
phi1_0_values = np.linspace(-np.pi / 2, np.pi / 2, num_values)
phi2_0_values = np.linspace(-np.pi / 2, np.pi / 2, num_values)

# Fonction pour mettre à jour le tracé en fonction du temps
def update(frame):
    plt.cla()  # Efface le tracé précédent
    t = frame / 10  # Conversion du numéro de trame en temps
    phi1_at_t = np.zeros((num_values, num_values))
    # Boucle pour les valeurs de Phi1(0) et Phi2(0)
    for i, phi1_0 in enumerate(phi1_0_values):
        for j, phi2_0 in enumerate(phi2_0_values):
            # Appel de la fonction rk4_coupled_without_the_assumption_of_small_oscillations
            t_, v = rk4_coupled_without_the_assumption_of_small_oscillations(0, t, 0.1, np.array([phi1_0,
phi2_0, 0, 0]),

differential_equation_system_without_the_assumption_of_small_oscillations,
                                                                    4, 1, 1, 1, 1, 9.81)
            # Stockage de la valeur de Phi1 à t
            phi1_at_t[i, j] = v[0][-1]
    # Création de la grille de couleur
    plt.imshow(phi1_at_t, extent=[-np.pi / 2, np.pi / 2, -np.pi / 2, np.pi / 2], cmap='inferno')
    plt.xlabel('Phi1(0)')
    plt.ylabel('Phi2(0)')
    plt.title(f'Variation de Phi1 à t = {t}s en fonction de Phi1(0) et Phi2(0)')

# Initialisation de la figure et de l'axe
fig, ax = plt.subplots(figsize=(10, 8))

# Création de l'animation
ani = FuncAnimation(fig, update, frames=np.arange(0, 601, 10), interval=200)

# Affichage de l'animation
HTML(ani.to_html5_video())
```



$$\lambda = \frac{1}{N \cdot [t_{\text{end}} - t_{\text{start}}]} \sum_{i=1}^{N} \sum_{j=1}^{M} \ln \left[\frac{|\delta x_i(t_j)|}{|\delta x_i(t_{j-1})|} \right]$$

```python
def calculate_lyapunov_exponent(start, end, step, v_0, derivated, order, m1, m2, l1, l2, g,
num_trajectories=10):
    # On initialise l'exposant
    lyapunov_sum = 0.0

    # On boucle sur le nombre de trajectoire
    for _ in range(num_trajectories):
        # On définit une perturbation de l'ordre de 10^-6
        perturbation = 1e-6 * np.random.randn(order)
        # On ajoute cette perturbation à la trajectoire que l'on veut perturber
        perturbed_v0 = v_0 + perturbation
        # On résoud les équations différentielles pour la trajectoire perturbée.
        # On prend soin de mettre un indice muet '_' pour le temps puisque l'on veut récupérer le vecteur
d'état du système et pas le temps
        _, perturbed_trajectory = rk4_coupled_without_the_assumption_of_small_oscillations(start, end,
step, perturbed_v0, derivated, order, m1, m2, l1, l2, g)
        # On fait de même pour la trajectoire non perturbée
        _, unperturbed_trajectory = rk4_coupled_without_the_assumption_of_small_oscillations(start, end,
step, v_0, derivated, order, m1, m2, l1, l2, g)
        # On calcule l'écart entre les trajectoires
        divergence = np.linalg.norm(perturbed_trajectory - unperturbed_trajectory, axis=0)
        # Puis on effectue la somme des logarithme de ces écarts
        lyapunov_sum +=np.sum(np.log(divergence[1:]/divergence[:-1]))
    # On calcule enfin l'exposant de Lyapunov que l'on retourne
    lyapunov_exponent = lyapunov_sum / (num_trajectories * (end-start))
    return lyapunov_exponent
```

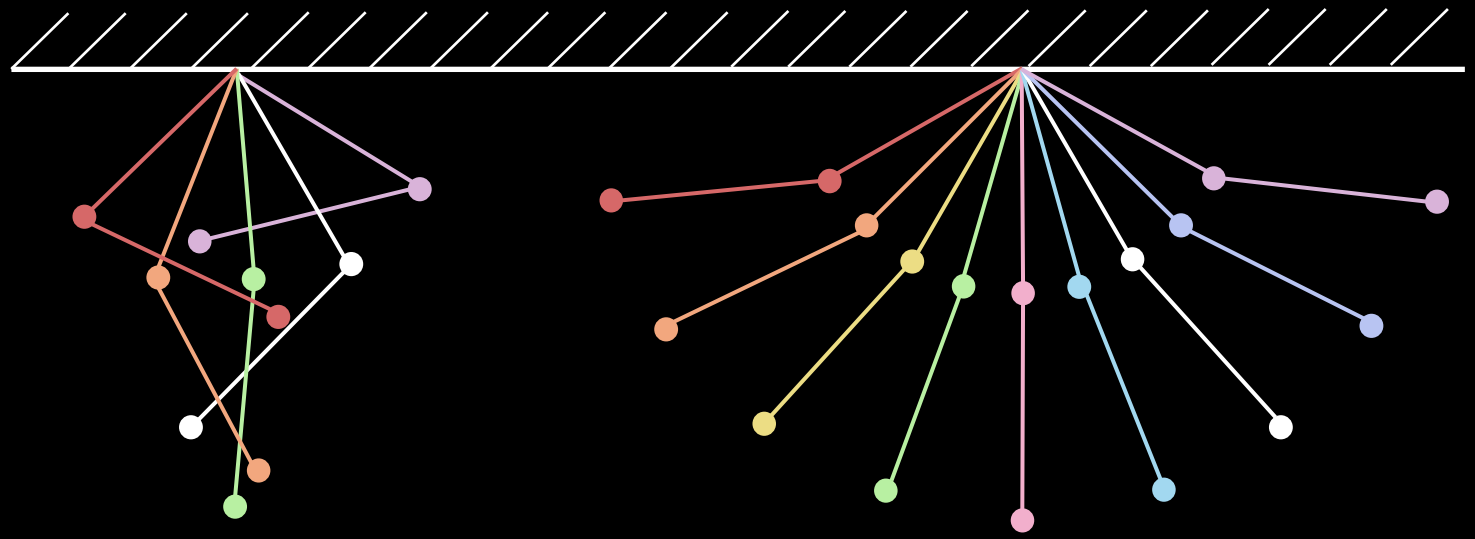


Figure 3.6- Modes propres du pendule double. A gauche $V_+(t)$, et à droite $V_-(t)$.

```python
def find_resonance_frequency_of_the_double_pendulum(λ_1, λ_2, ω0):
    eigenpulsation_1 = np.sqrt(λ_1)/(2*np.pi)
    eigenpulsation_2 = np.sqrt(λ_2)/(2*np.pi)
    eigenfrequency_1 = eigenpulsation_1*ω0
    eigenfrequency_2 = eigenpulsation_2*ω0
    return np.array([eigenpulsation_1, eigenpulsation_2, eigenfrequency_1, eigenfrequency_2])
```

U.S. DEPARTMENT OF INTERIOR
GEOLOGICAL SURVEY

OBSERVATIONS AND MODELING

OF

SEISMIC BACKGROUND NOISE

Jon Peterson

Open-File Report 93-322

This report is preliminary and has not been reviewed for conformity with U.S. Geological Survey editorial standards. Any use of trade names is for descriptive purposes only and does not imply endorsement by the U.S. Geological Survey.

Albuquerque, New Mexico

1993

Jon Peterson, 1993

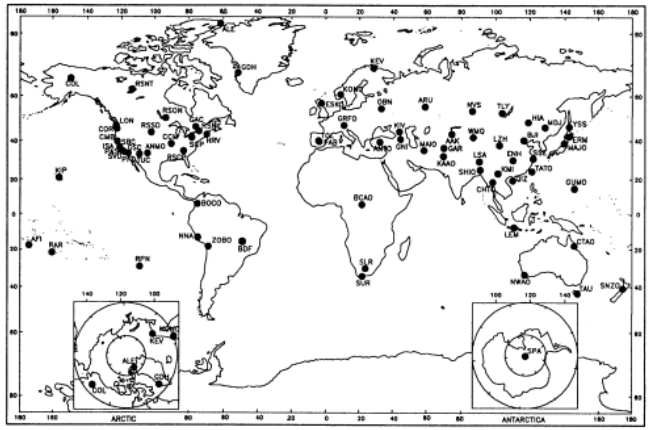


Figure 1.--Map showing the approximate locations of the stations used in this report.

THE DATABASE

Stations

Seismic data from 75 stations have been used in this study. All but one of the stations are part of what will be generically referred to as the global seismograph network, and all of the data recorded at the network stations, with the exception of some test data, are available to anyone through either the USGS or IRIS data management centers. The codes, coordinates, type of data system, and vault and site characteristics, if known, are listed in Table 1. A few of the data systems are no longer in operation, and one of the stations listed, ALQ, is a test system at ASL that operates intermittently. Several of the stations have been the subjects of previous background noise studies, generally in greater scope and detail than reported in this study. An analysis of seismic noise recorded at the five RSTN stations has been published by Rodgers and others (1987) and Breding (1987). Given (1990) has published a detailed report on seismic noise recorded at three of the IRIS/IDA sites listed in Table 1 (ARU, KIV, OBN).

Constitution d'une des plus grandes bases de données au monde

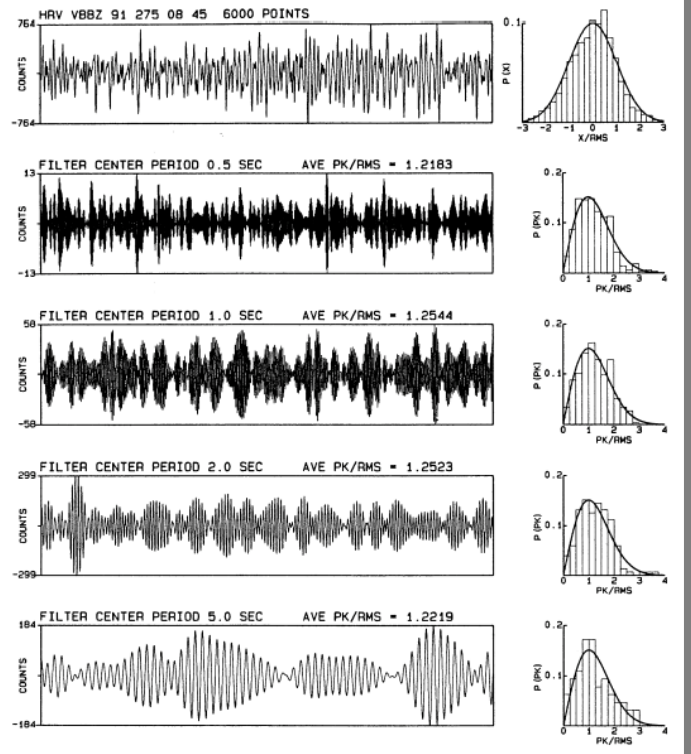
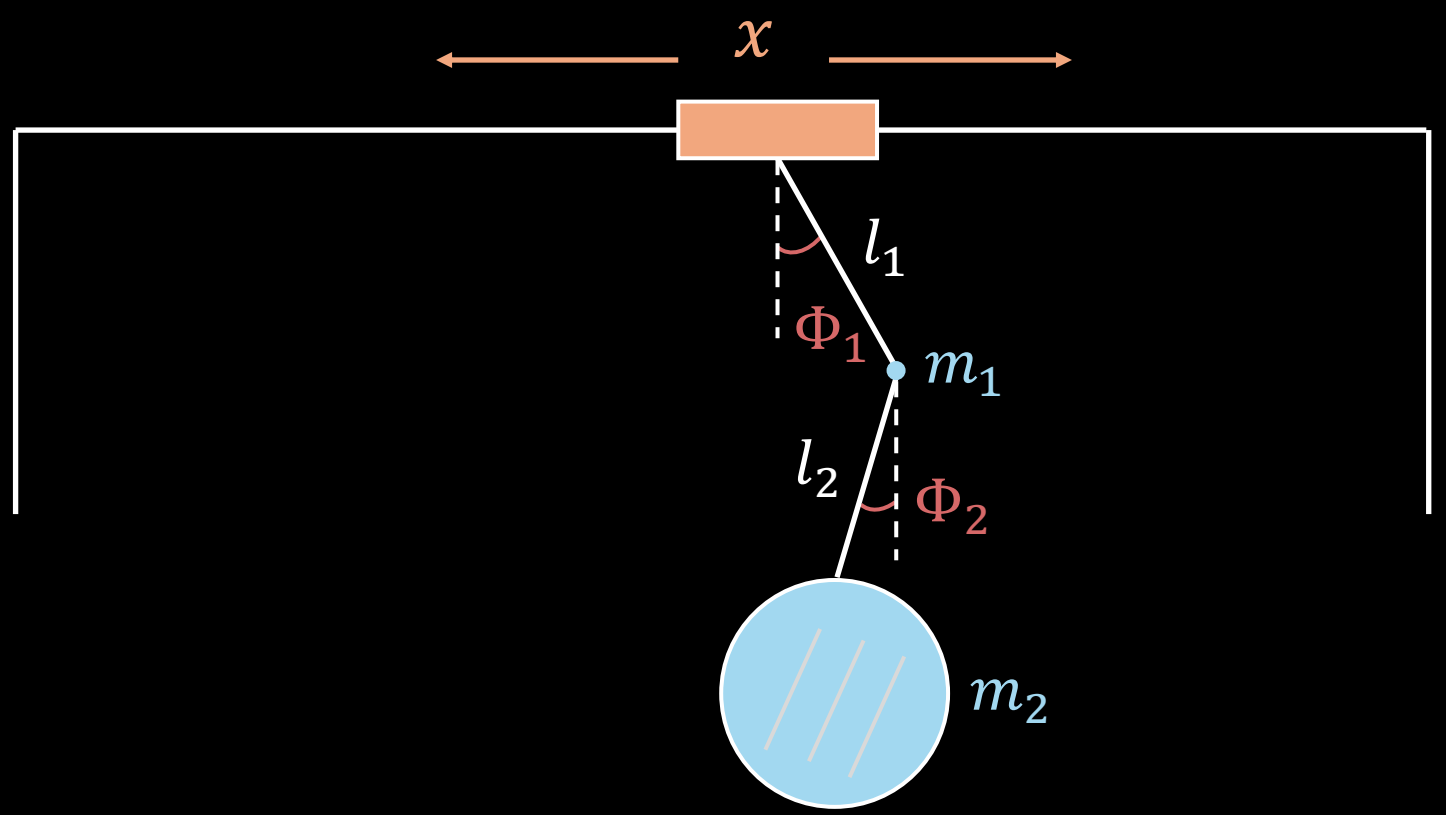
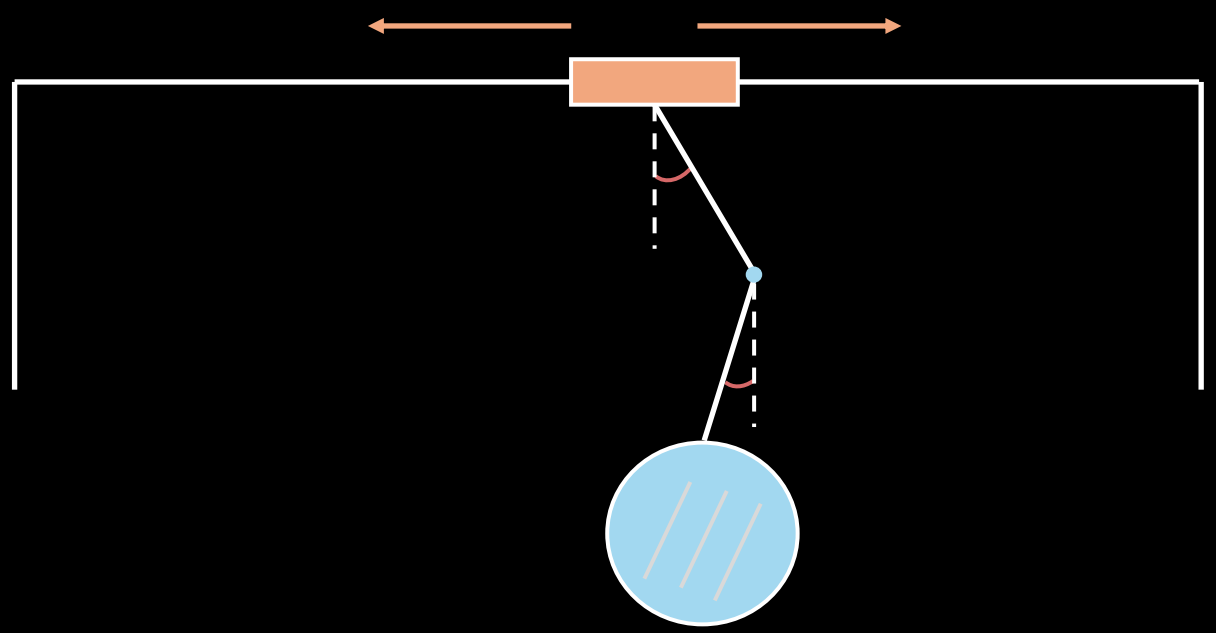


Figure 16.--An original VBBZ waveform at the top and four filtered waveforms derived from the original using 1/3-octave filters at the center periods listed. The distribution of sample amplitudes in the original time series is plotted to the right of the original waveform together with a Gaussian probability density function. The distributions of individual peak values in the filtered waveforms are shown plotted to the right of each the waveforms together with Rayleigh probability density functions. The ratios of average peak values to the RMS amplitude vary from 1.2183 to 1.2544. The theoretical value is 1.253.

Modèles de bruits sismiques (NLNM) et (NHNM)


x
l_1
Φ_1
m_1
l_2
Φ_2
m_2



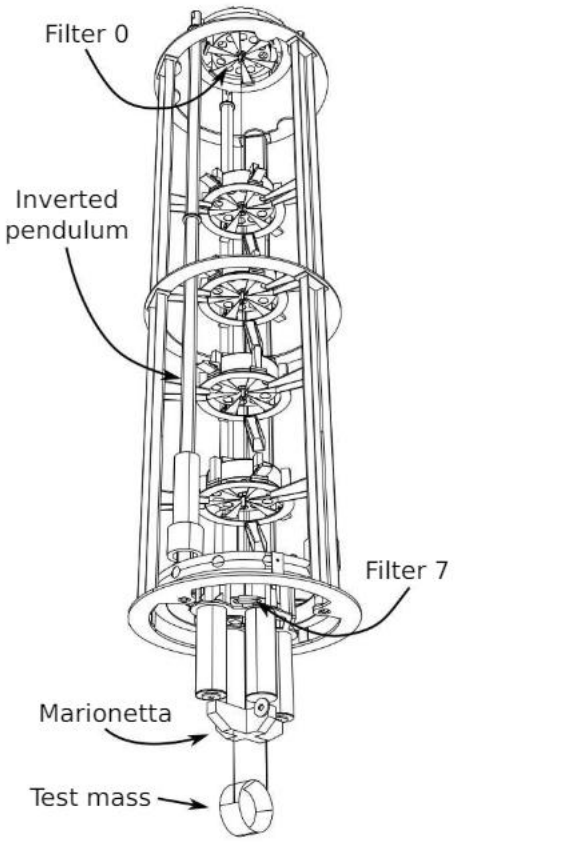
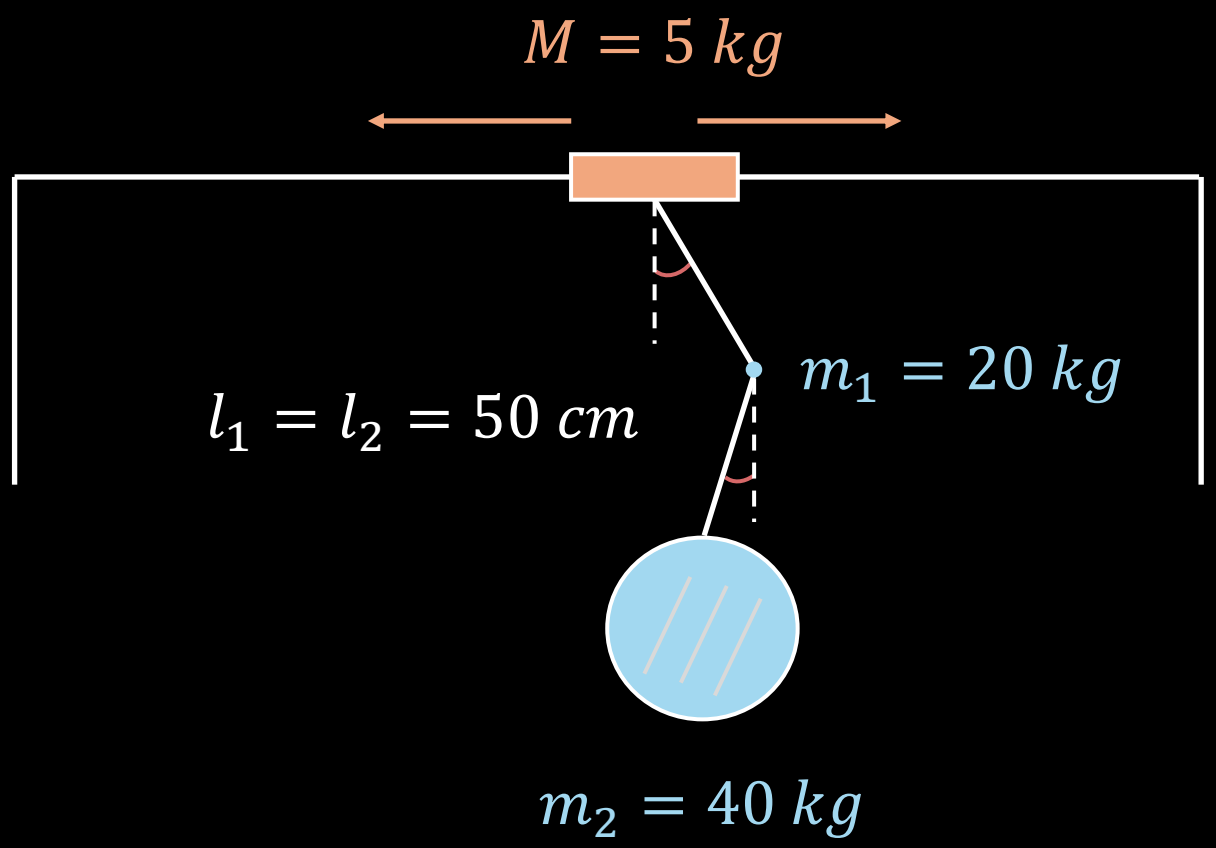
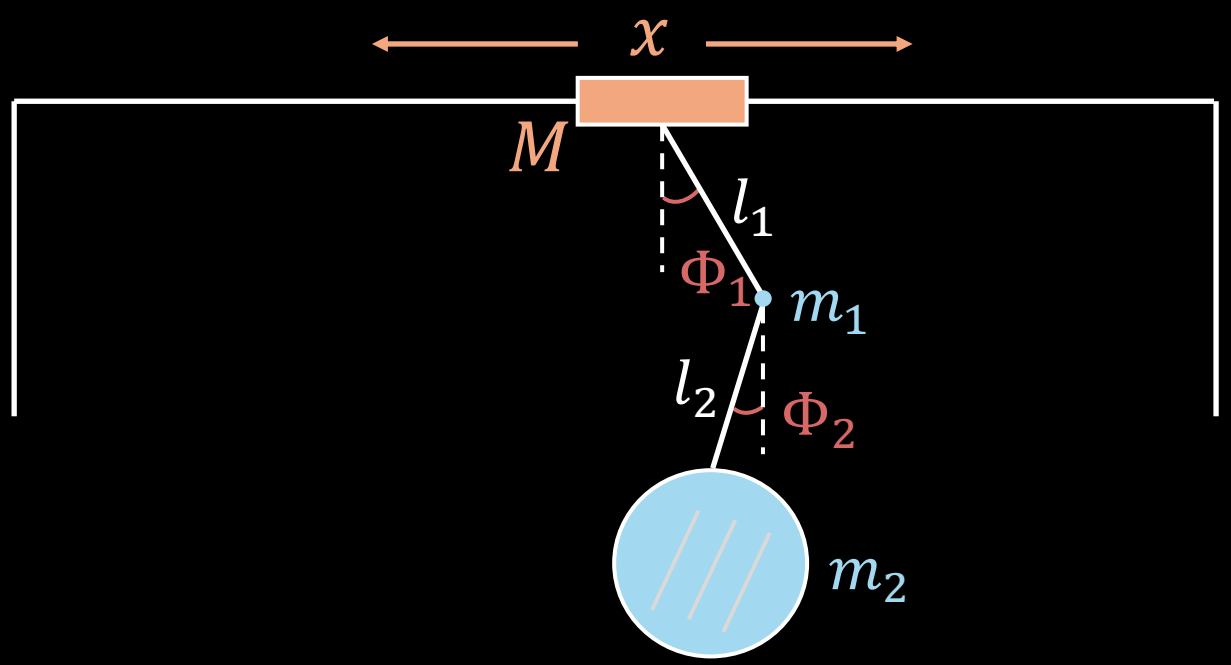


FIGURE 3.1: Mechanical scheme of the Superattenuator. Its aim is to isolate the test masses from the ground motion.



*« Les deux masses supérieures sont en métal, pesant chacune **20 kg**, tandis que les troisième et quatrième sont des cylindres de verre de silice vierge pesant chacun **40 kg** ».* California Institute of Technology.



- $U = m_1 g l_1 \cos\Phi_1 + m_2[l_1 \cos\Phi_1 + l_2 \cos\Phi_2]$
- T
$= \frac{1}{2} M \dot{x}^2$
$+ \frac{1}{2} m_1 \left[\left(\dot{x} + l_1 \dot{\Phi}_1 \cos(\Phi_1)\right)^2 + \left(l_1 \dot{\Phi}_1 \sin(\Phi_1)\right)^2\right]$
$+ \frac{1}{2} m_2 \left[\left(\dot{x} + l_1 \dot{\Phi}_1 \cos(\Phi_1) + l_2 \dot{\Phi}_2 \cos(\Phi_2)\right)^2 + \left(l_1 \dot{\Phi}_1 \sin(\Phi_1) + l_2 \dot{\Phi}_2 \sin(\Phi_2)\right)^2\right]$
- $\mathcal{L} = T - U$
- $\begin{cases} \frac{d}{dt}\left[\frac{\partial \mathcal{L}}{\partial \dot{x}}\right] - \frac{\partial \mathcal{L}}{\partial x} = 0 \\ \frac{d}{dt}\left[\frac{\partial \mathcal{L}}{\partial \dot{\Phi}_1}\right] - \frac{\partial \mathcal{L}}{\partial \Phi_1} = 0 \\ \frac{d}{dt}\left[\frac{\partial \mathcal{L}}{\partial \dot{\Phi}_2}\right] - \frac{\partial \mathcal{L}}{\partial \Phi_2} = 0 \end{cases}$ • • •

Système de Cramer

$$\mathrm{A} \cdot \ddot{\mathrm{X}}(t) = \mathrm{B}$$

- $$\mathrm{A} = \begin{bmatrix} M + m_1 + m_2 & l_1(m_1 + m_2)cos(\Phi_1) & l_2 m_2 cos(\Phi_2) \\ l_1(m_1 + m_2)cos(\Phi_1) & l^2{}_1(m_1 + m_2) & l_1 l_2 m_2 cos(\Phi_1 - \Phi_2) \\ l_2 m_2 cos(\Phi_2) & l_1 l_2 m_2 cos(\Phi_1 - \Phi_2) & m_2 l^2{}_2 \end{bmatrix}$$
- $$B = \begin{bmatrix} l_1(m_1 + m_2)\dot{\Phi}^2{}_1 sin(\Phi_1) + l_2 m_2 \dot{\Phi}^2{}_2 sin(\Phi_2) + \partial_x \mathcal{L} \\ l_1(m_1 + m_2)\dot{x}\dot{\Phi}_1 sin(\Phi_1) + l_1\, l_2 m_2 \dot{\Phi}_2 \left(\dot{\Phi}_1 - \dot{\Phi}_2\right) sin(\Phi_1 - \Phi_2) + \partial_{\Phi_1} \mathcal{L} \\ l_2 m_2 \dot{x}\dot{\Phi}_2 sin(\Phi_2) + l_1 l_2 m_2 \dot{\Phi}_1 \left(\dot{\Phi}_1 - \dot{\Phi}_2\right) sin(\Phi_1 - \Phi_2) + \partial_{\Phi_2} \mathcal{L} \end{bmatrix}$$
- $$\ddot{\mathrm{X}}(t) = \begin{bmatrix} \ddot{x} \\ \ddot{\Phi}_1 \\ \ddot{\Phi}_2 \end{bmatrix}$$

Règle de Cramer

$$\ddot{x}(t) = \frac{1}{detA}\begin{vmatrix} \boldsymbol{b_{11}} & a_{12} & a_{13} \\ \boldsymbol{b_{21}} & a_{22} & a_{23} \\ \boldsymbol{b_{31}} & a_{32} & a_{33} \end{vmatrix} \qquad \ddot{\Phi}_1(t) = \frac{1}{detA}\begin{vmatrix} a_{11} & \boldsymbol{b_{11}} & a_{13} \\ a_{21} & \boldsymbol{b_{21}} & a_{23} \\ a_{31} & \boldsymbol{b_{31}} & a_{33} \end{vmatrix}$$

$$\ddot{\Phi}_2(t) = \frac{1}{detA}\begin{vmatrix} a_{11} & a_{12} & \boldsymbol{b_{11}} \\ a_{21} & a_{22} & \boldsymbol{b_{21}} \\ a_{31} & a_{32} & \boldsymbol{b_{31}} \end{vmatrix}$$

```python
def specifies_the_differential_equations_of_the_inverted_double_pendulum(u, step, t_, dt):
    # On récupère les valeurs des angles et des vitesses angulaire
    # On récupère les valeurs du déplacement du double pendule, et de sa vitesse de déplacement
    x, dx, ϕ1, dϕ1, ϕ2, dϕ2 = u

    # On écrit les dérivées du lagrangien par rapport à la vitesse du déplacement du double pendule
    # On écrit les dérivées du lagrangien par rapport auM vitesses angulaires
    derivative_of_the_lagrangian_with_respect_to_dx = 0.0
    derivative_of_the_lagrangian_with_respect_to_dϕ1 = -(m1 + m2) * l1 * dϕ1 * dx * sin(ϕ1) + (m1 + m2) *
g * l1 * sin(
        ϕ1) - m2 * l1 * l2 * dϕ1 * dϕ2 * sin(ϕ1 - ϕ2)
    derivative_of_the_lagrangian_with_respect_to_dϕ2 = m2 * l2 * (
                g * sin(ϕ2) + l1 * dϕ1 * dϕ2 * sin(ϕ1 - ϕ2) - dx * dϕ2 * sin(ϕ2))

    # Pour alléger les lignes de codes qui suivent, on écrit les composantes de A dans des variables
    # Le premier chiffre indique la ligne et le second la colonne de la matrice
    component_11_of_A = M + m1 + m2
    component_12_of_A = (m1 + m2) * l1 * cos(ϕ1)
    component_13_of_A = m2 * l2 * cos(ϕ2)
    component_21_of_A = (m1 + m2) * l1 * cos(ϕ1)
    component_22_of_A = (m1 + m2) * l1 ** 2
    component_23_of_A = m2 * l1 * l2 * cos(ϕ1 - ϕ2)
    component_31_of_A = m2 * l2 * cos(ϕ2)
    component_32_of_A = m2 * l1 * l2 * cos(ϕ1 - ϕ2)
    component_33_of_A = m2 * l2 ** 2

    # On fait de même pour la matrice B
    component_11_of_B = (m1 + m2) * l1 * dϕ1 ** 2 * sin(ϕ1) + m2 * l2 * dϕ2 ** 2 * sin(ϕ2)
    component_21_of_B = (m1 + m2) * dx * dϕ1 * l1 * sin(ϕ1) + m2 * l1 * l2 * dϕ2 * (dϕ1 - dϕ2) * sin(ϕ1 -
ϕ2)
    component_31_of_B = m2 * dx * dϕ2 * l2 * sin(ϕ2) + m2 * l1 * l2 * dϕ1 * (dϕ1 - dϕ2) * sin(ϕ1 - ϕ2)

    # On peut maintenant écrire les matrices A et B
    A = np.array([[component_11_of_A, component_12_of_A, component_13_of_A],
                  [component_21_of_A, component_22_of_A, component_23_of_A],
                  [component_31_of_A, component_32_of_A, component_33_of_A]])

    B = np.array([component_11_of_B + derivative_of_the_lagrangian_with_respect_to_dx,
                  component_21_of_B + derivative_of_the_lagrangian_with_respect_to_dϕ1,
                  component_31_of_B + derivative_of_the_lagrangian_with_respect_to_dϕ2])

    # On calcule le déterminant de A avec la fonction linalg.det() de numpy
    determinant_of_matrix_A = np.linalg.det(A)

    # On créer une copie de la matrice pour ne pas affecter l'original
    first_copy_of_matrix_A = np.copy(A)
    # On utilise la méthode de Cramer -> on remplace la première colonne de A par la matrice B
    first_copy_of_matrix_A[:, 0] = B
    # Première équation différentielle
    ddx = np.linalg.det(first_copy_of_matrix_A) / determinant_of_matrix_A

    # On itère le même processus pour obtenir l'accélération angulaire 1
    second_copy_of_matrix_A = np.copy(A)
    # Attention, on remplace cette fois-ci la deuxième colonne de A par la matrice B
    second_copy_of_matrix_A[:, 1] = B
    # Seconde équation différentielle
    ddϕ1 = np.linalg.det(second_copy_of_matrix_A) / determinant_of_matrix_A

    # De même pour l'accélération angulaire de l'angle 2
    third_copy_of_matrix_A = np.copy(A)
    # On remplace la troisième colonne de A par B
    third_copy_of_matrix_A[:, 2] = B
    # Troisième équation différentielle
    ddϕ2 = np.linalg.det(third_copy_of_matrix_A) / determinant_of_matrix_A

    return [dx, ddx, dϕ1, ddϕ1, dϕ2, ddϕ2]
```

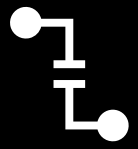
$\frac{\partial \mathcal{L}}{\partial x}, \frac{\partial \mathcal{L}}{\partial \Phi_1}, \frac{\partial \mathcal{L}}{\partial \Phi_2}$

Composantes de la matrice A

Composantes de la matrice B

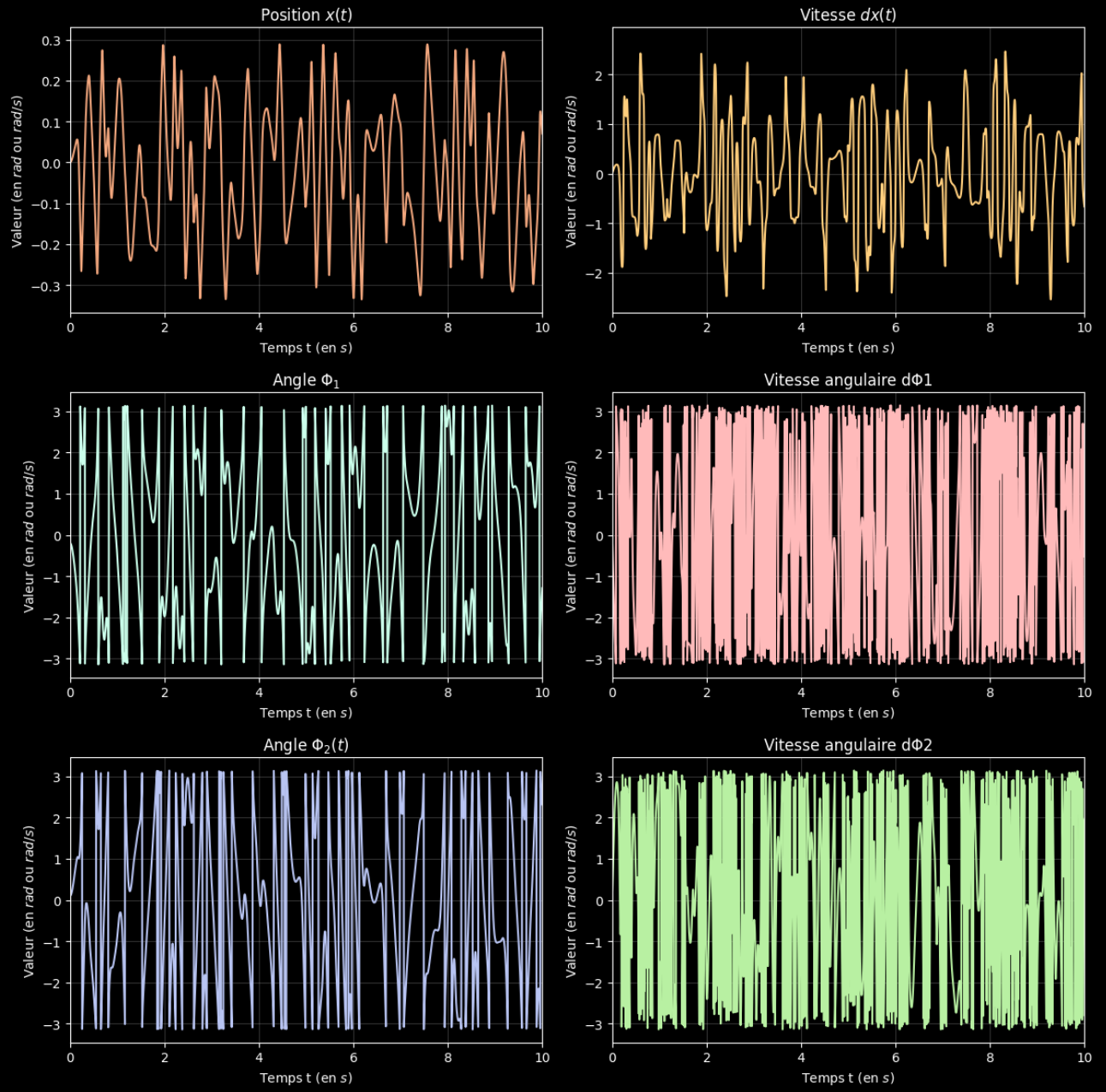
Règle de Cramer

Séparation de la fonction de résolution et de la méthode rk4

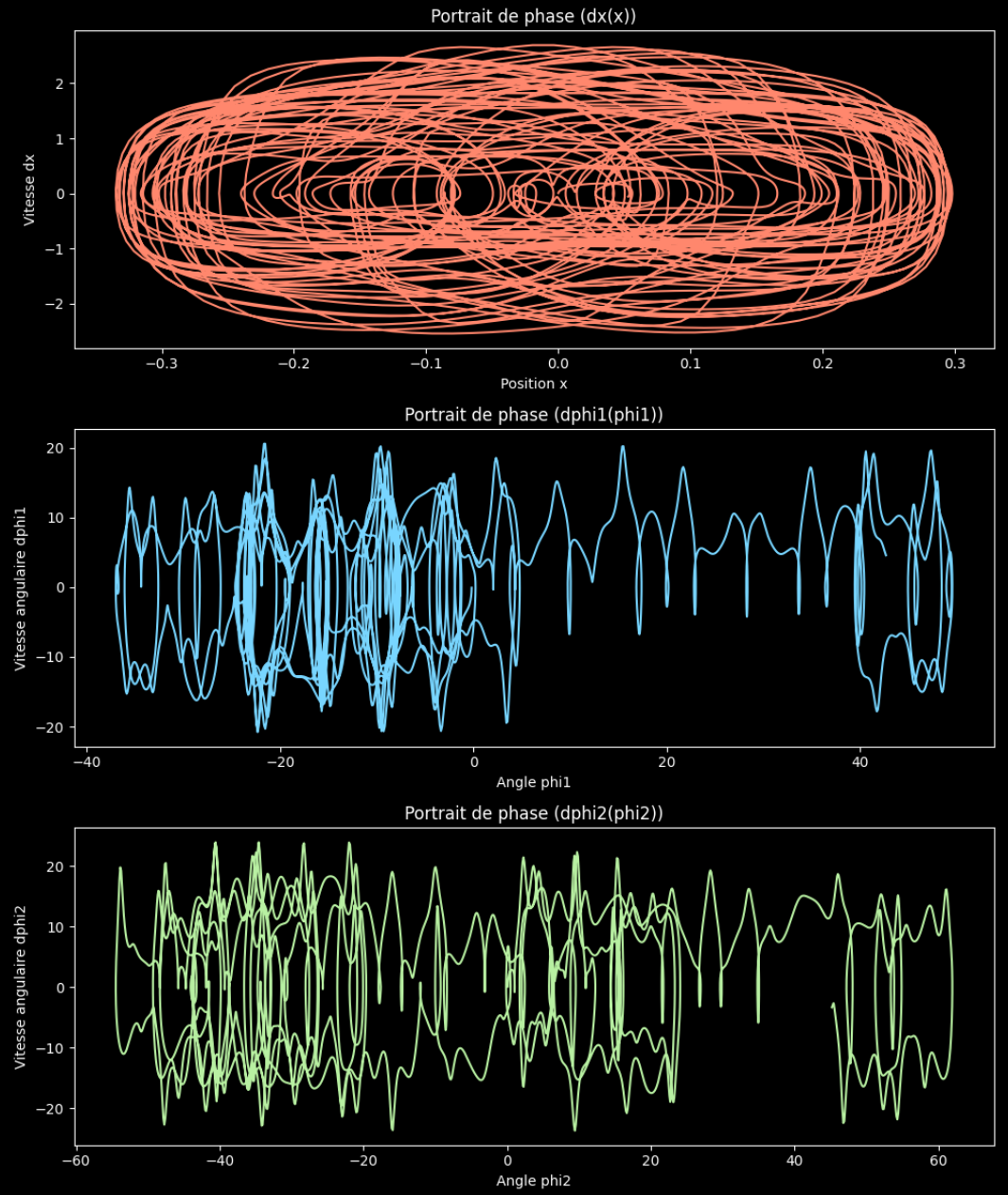


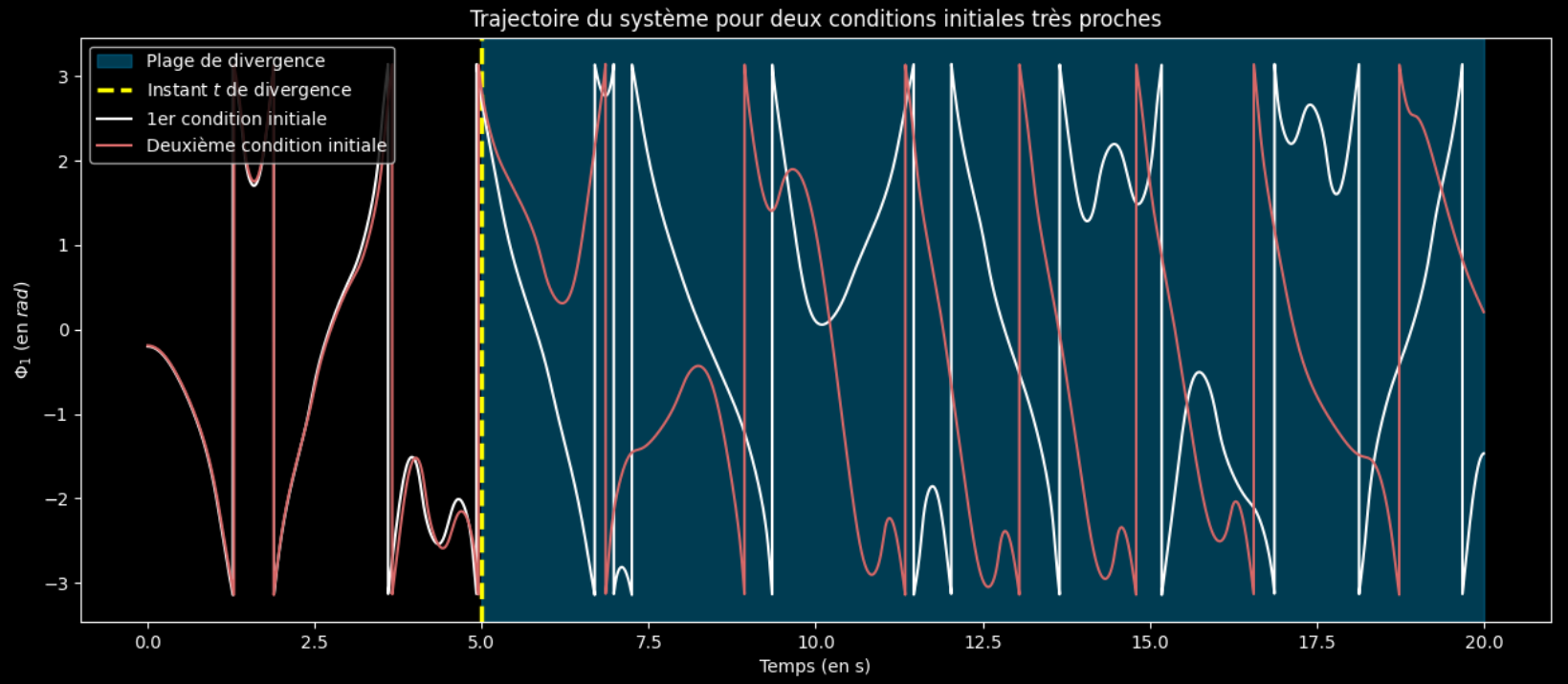
```python
def solve(v0, t, solution_method, derivative):
    # dt -> écart entre deux valeurs de temps consécutives (ici entre la première et la seconde)
    dt = t[1] - t[0]
    # On initialise le vecteur d'état initial du système
    u = [v0]
    # Pour chaque valeur de t associé à son indice step
    for step, t in enumerate(t):
        # On ajoute le résultat qui sera renvoyé par rd4
        u.append(solution_method(u[-1], step, t, dt, derivative))
    # On retourne le vecteur d'état du système
    return np.array(u)


def integrate_rk4(u, step, t, dt, derivative):
    d1 = derivative(u, step, t, dt)
    d2 = derivative([v + d * dt / 2 for v, d in zip(u, d1)], step, t, dt)
    d3 = derivative([v + d * dt / 2 for v, d in zip(u, d2)], step, t, dt)
    d4 = derivative([v + d * dt for v, d in zip(u, d3)], step, t, dt)
    v = [v + (d1_ + 2 * d2_ + 2 * d3_ + d4_) * dt / 6 for v, d1_, d2_, d3_, d4_ in zip(u, d1, d2, d3,
d4)]
    return v
```

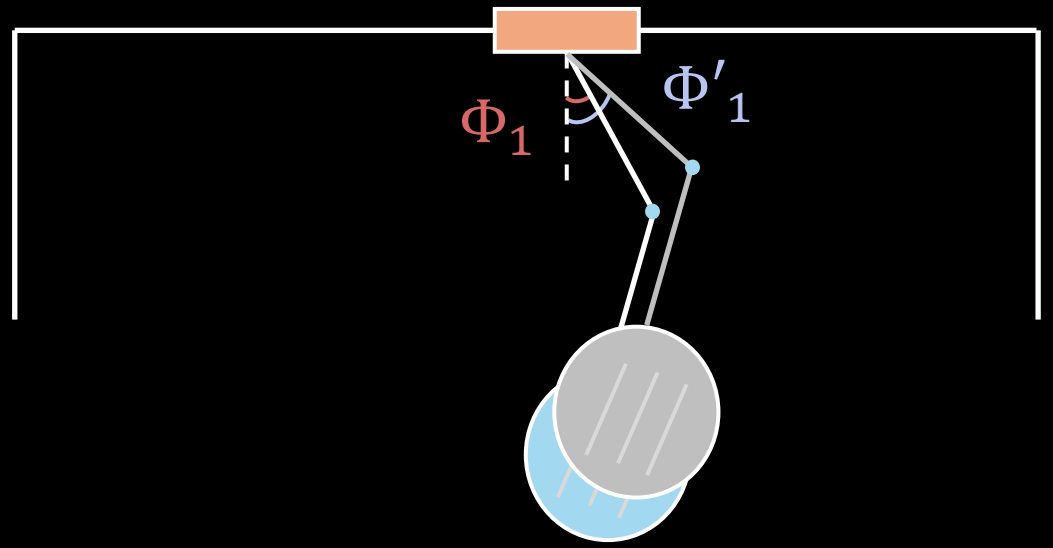
III/ Le double pendule inversé



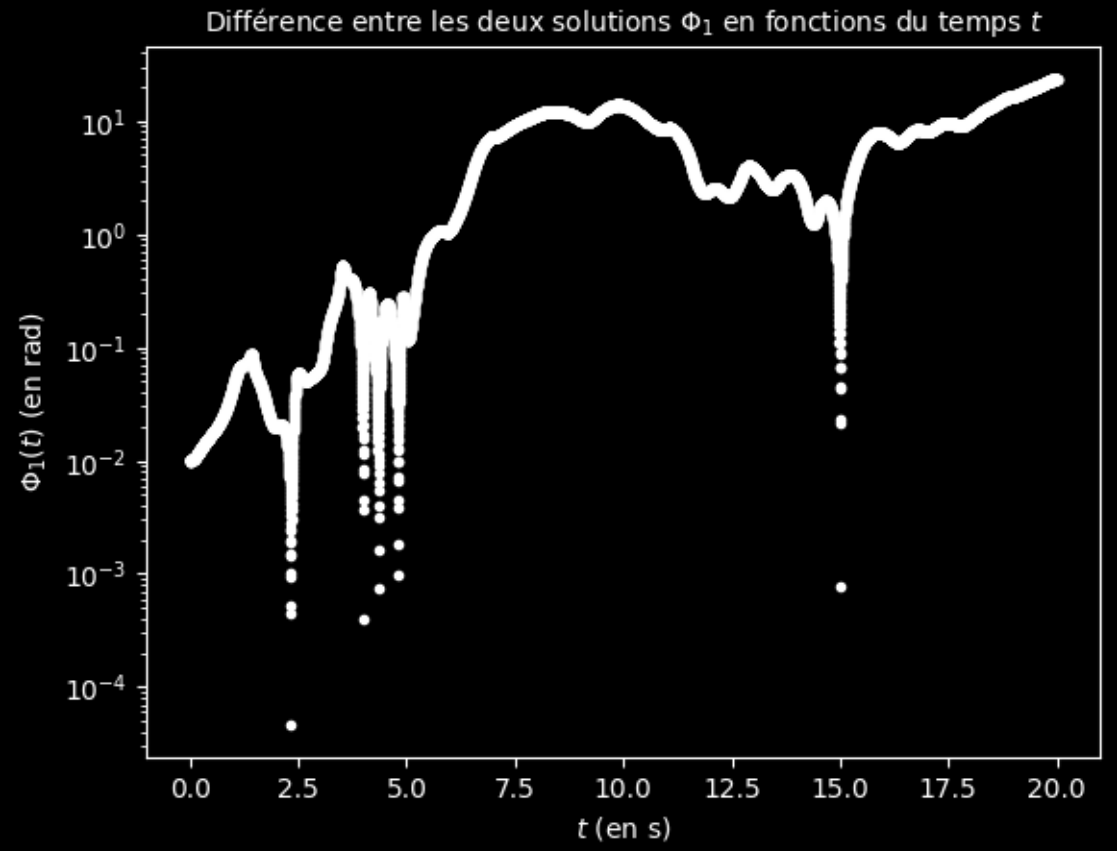
III/ Le double pendule inversé

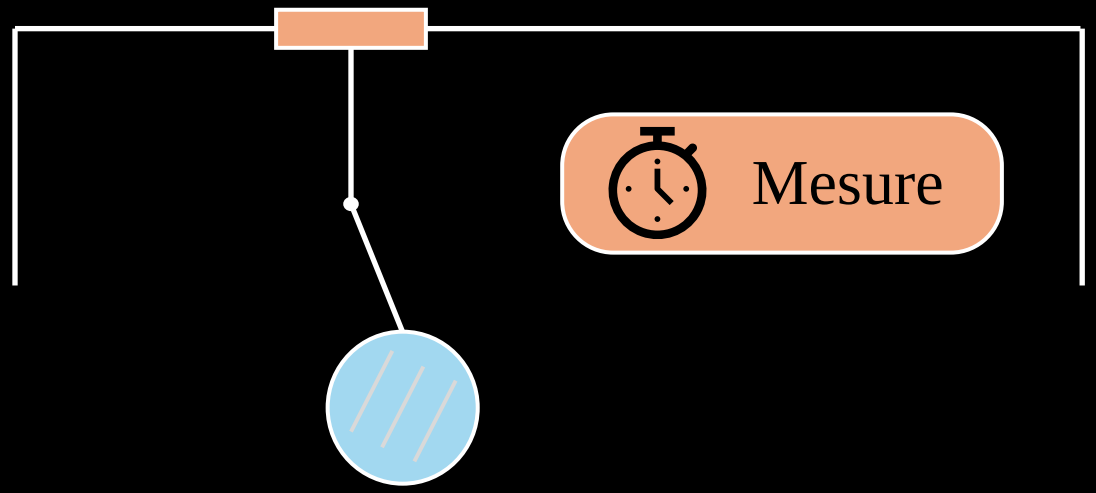


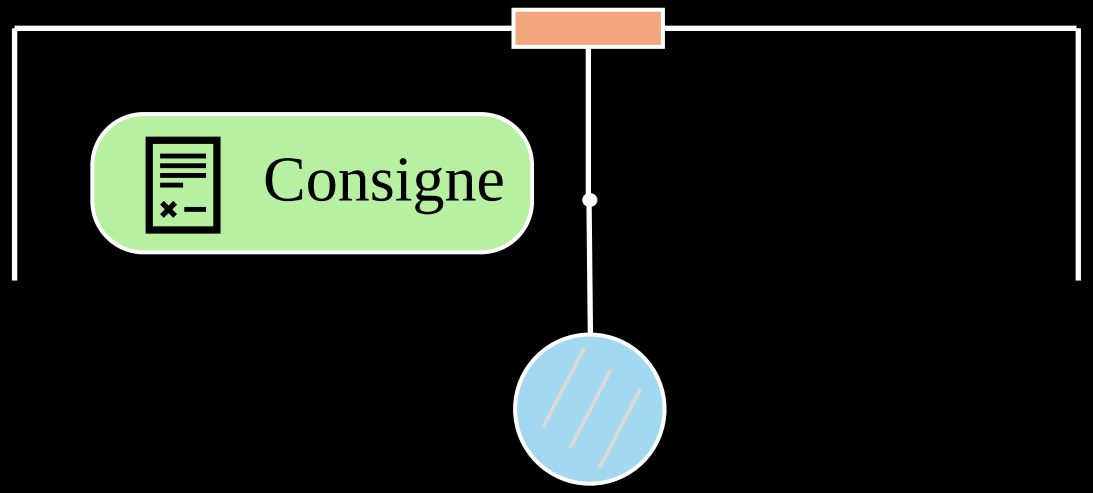

Trajectoire du système pour deux conditions initiales très proches
Plage de divergence
Instant t de divergence
1er condition initiale
Deuxième condition initiale
Φ1 (en rad)
Temps (en s)
3
2
1
0
−1
−2
−3
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


Φ_1
Φ'_1

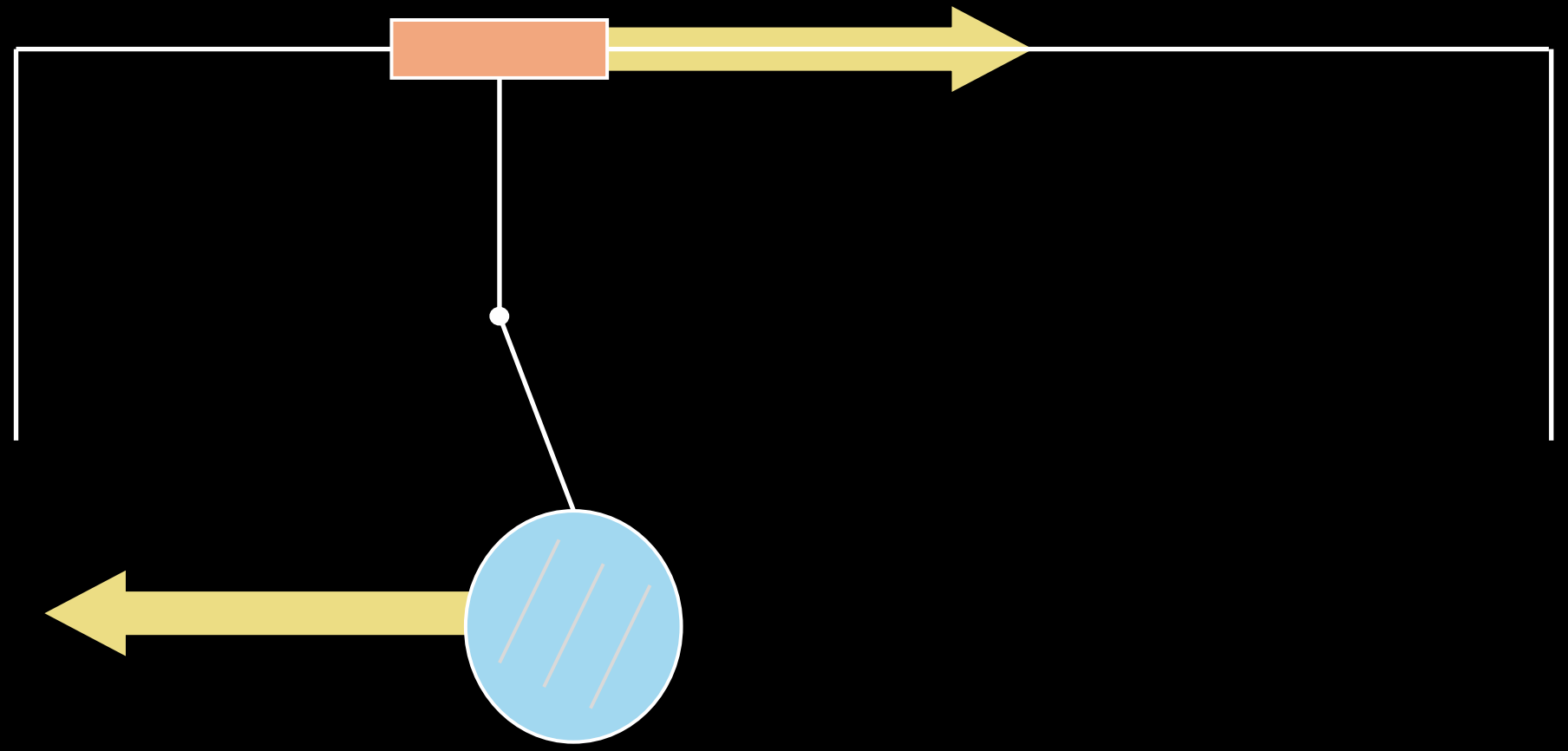
$$|\Phi'_1 - \Phi_1| = f(t)$$

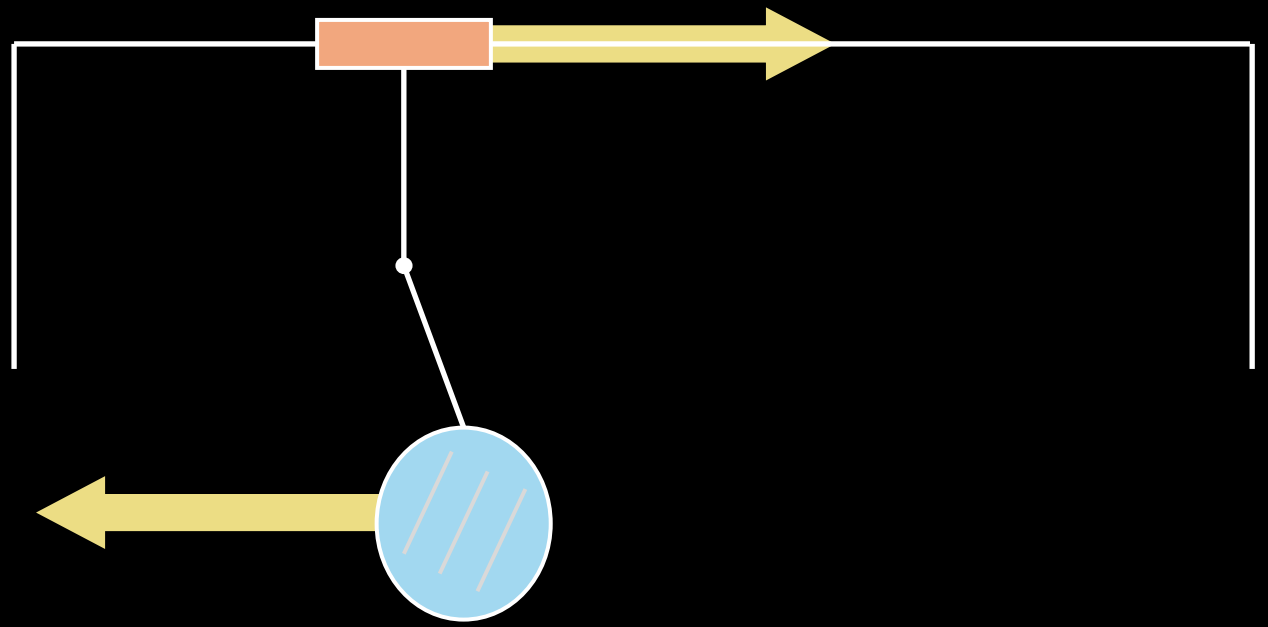

Différence entre les deux solutions Φ_1 en fonctions du temps t
$\Phi_1(t)$ (en rad)
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
t (en s)





$$\text{Commande} = f(\underbrace{\text{Consigne} - \text{Mesure}}_{\text{Erreur}})$$





- Les variables sont corrélées entre elles
- Il faut donc une commande qui synchronise les variables
  ⇒ **Coordination multi-axes**

CONTRIBUTIONS TO THE THEORY OF OPTIMAL CONTROL

By R. E. Kalman

1. Introduction

The purpose of this paper is to give an account of recent research on a classical problem in the theory of control: the design of linear control systems so as to minimize the integral of a quadratic function evaluated along motions of the system. This problem dates back in its modern form to Wiener and Hall at about 1943 ([1], [2]). In spite of its relatively long history, the problem has never been formulated rigorously from a mathematical point of view. Even the most up-to-date expositions of the subject (see, e.g., [3]) are inaccessible to the mathematician due to the lack of precisely stated conditions and results.

The problem is quite broad, and there are many unsettled questions. This paper will be concerned with only the simplest case, the so-called *regulator problem*. For other aspects of the problem, the reader may consult [4]–[8] which, while devoted primarily to questions of theoretical engineering, contain precise mathematical results.

The conventional theory of the regulator problem is based largely on Fourier and Laplace transforms. By contrast, the approach of this paper is direct and uses the well-known theory of ordinary differential equations. We have also drawn on Lyapunov's theory of stability. While our earlier treatments ([5], [6], [8]) followed the point of view of dynamic programing, here we utilize classical tools of the calculus of variations, in particular the Hamilton-Jacobi equation.

The principal contribution of the paper lies in the introduction and exploitation of the concepts of *controllability* and *observability* ([8]), with the aid of which we give, for the first time, a complete theory of the regulator problem. In particular, we prove existence and stability theorems for the regulator problem and study in some detail stability properties of the matrix Riccati equation, which arises as a special case of the Hamilton-Jacobi equation.

A careful discussion of the conceptual aspects of the control problems has been included as an aid to persons not familiar with the field of control. Some mathematical arguments, in particular the review of the calculus of variations, are more leisurely than usual in order to render the paper reasonably self-contained.

2. Notation and terminology

We use standard vector-matrix notation, with the following conventions: small Greek letters are scalars; small Latin letters are vectors, capitals are matrices. The unit matrix is I. Exceptions: i, j, m, n, p are integers; t, E, L, V are scalars; ϕ, ψ, ξ are vectors. The inner product is $[x, y]$. The transpose of a matrix is denoted by the prime. The norm is $\| x \| = [x, x]^{\frac{1}{2}}$. The norm $\| A \|$ of a matrix A is sup $\| Ax \|$ over $\| x \| = 1$. Special conventions: $\| x \|_P^2 = [x, Px]$ where P is

Reprinted with permission from *Boletin de la Sociedad Matematica Mexicana*, R. E. Kalman, "Contributions to the Theory of Optimal Control," Vol. 5, 1960, pp. 102–119.

Rudolf Kalman, 1964

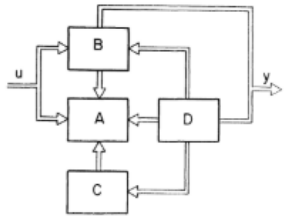


Figure 4.

parts of the canonical structure "subsystems." For constant systems this difficulty does not arise. More generally, we have:

Theorem 6. *For a periodic or analytic linear dynamical system* (2.1–2) *the dimension numbers* $n_A, \cdots, n_D$ *are constants, and the canonical decomposition is continuous with respect to* t.

An illustration of the canonical structure theorem is provided by

Example 2. Consider the constant system defined by

$$F = \begin{bmatrix} -3 & -3 & 0 & 1 \\ 26 & 36 & -3 & -25 \\ 30 & 39 & -2 & -27 \\ 30 & 43 & -3 & -32 \end{bmatrix},$$

$$G = \begin{bmatrix} 3 & 3 \\ -2 & -1 \\ 0 & 0 \\ 0 & 1 \end{bmatrix},$$

and

$$H = [-5 \quad -8 \quad 1 \quad 5].$$

We introduce new coördinates by letting $\bar{x} = Tx$, where

$$T = \begin{bmatrix} 2 & 3 & 0 & -2 \\ 1 & 1 & 0 & -1 \\ -2 & -3 & 0 & 3 \\ -6 & -9 & 1 & 6 \end{bmatrix},$$

and

$$T^{-1} = \begin{bmatrix} 0 & 3 & 1 & 0 \\ 1 & -2 & 0 & 0 \\ 3 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}.$$

With respect to these new coördinates the system matrices assume the

Introduction de la matrice de gain

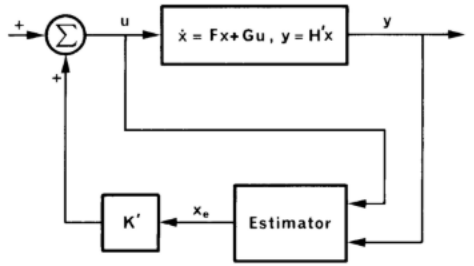


Figure 7.1-2 Use of estimator in implementing a control law.

only an estimate of $K'x$, this being merely a linear function or collection of linear functions of the states, rather than the full state vector x.)

2. It should function in the presence of noise. Preferably, it should be possible to optimize the action of the estimator in a noisy environment—to ensure that the noise has the least possible effect when the estimator is used in connection with controlling a system.

As we shall show, these properties are both more or less achievable. Estimators designed to behave *optimally in certain noise environments* turn out to consist of linear, finite-dimensional systems, if the system whose states are being estimated is linear and finite-dimensional. Moreover, if the dimensions of the system whose states are being estimated and the estimator are the same, and if the system whose states are being estimated is time-invariant and the associated noise is stationary, the estimator is time-invariant. Also, all these properties hold irrespective of the number of inputs and outputs of the system whose states are being estimated.

A further interesting property of such estimators is that their design is independent of the associated optimal feedback law K' shown in Fig. 7.1-2, or of any performance index used to determine K'. Likewise, the determination of K' is independent of the presence of noise, and certainly independent of the particular noise statistics. Yet, use of the control law $u = K'x_e$ turns out to be not merely approximately optimal, but exactly optimal for a modified form of performance index. This point is discussed at length in Chapter 8.

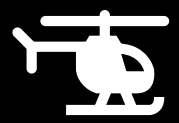
If satisfactory rather than optimal performance in the presence of noise is acceptable, two simplifications can be made. First, the actual computational procedure for designing an estimator (at least for scalar output systems) becomes far less complex. Second, it is possible to simplify the structure of the estimator, if desired. In other words, the designer can stick with the same estimator structure as is used

Commande linéaire quadratique LQR

Point A

Décision optimale ?

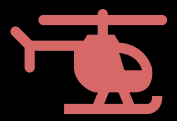
Point B

	Temps	Prix
	20 min	20 €
	5 min	500 €
	60 min	0 €

Point A

Décision optimale ?

Point B

	Temps	Prix
	20 min	20 €
	5 min	500 €
▶ 	60 min	**0 €**

Point A

Décision optimale ?

Point B

	Temps	Prix
	20 min	20 €
▶	**5 min**	500 €
	60 min	0 €

$$J(Q,R) = min\{Q \times temps + R \times prix\}$$

	Q	Temps	R	Prix	J
▶ 	1	× 20 min	2	× 20 €	= 60
	1	× 5 min	2	× 500 €	= 1005
	1	× 80 min	2	× 0 €	= 80

Temps → Performance

Prix → Effort mécanique fournie

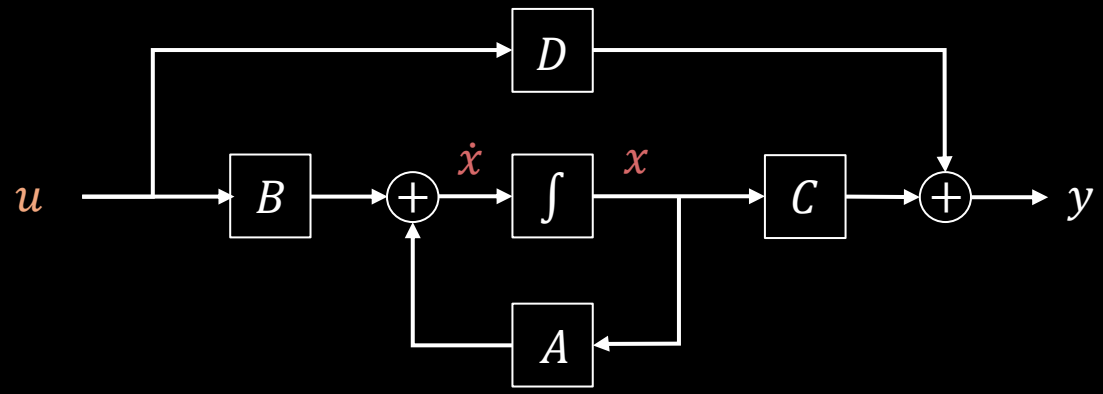
Moyen de transport → Combinaison linéaire de $\boldsymbol{Q}$ et $\boldsymbol{R}$

Représentation d'état

$$\begin{cases} \dot{x} = A\,x + B\,u \\ y = C\,x + D\,u \end{cases}$$

Etats internes

Entrée externe



$$M = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & (M + m_1 + m_2) & l_1(m_1 + m_2) & l_2 m_2 \\ 0 & 0 & 0 & l_1(m_1 + m_2) & l^2{}_1(m_1 + m_2) & l_1 l_2 m_2 \\ 0 & 0 & 0 & l_2 m_2 & l_1 l_2 m_2 & m_2 l^2{}_2 \end{bmatrix} \begin{matrix} \dot{x} \\ \dot{\Phi}_1 \\ \dot{\Phi}_2 \\ \ddot{x} \\ \ddot{\Phi}_1 \\ \ddot{\Phi}_2 \end{matrix}$$

$$N = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & l_1 g(m_1 + m_2) & 0 & 0 & 0 & 0 \\ 0 & 0 & l_2 g m_2 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} \dot{x} \\ \dot{\Phi}_1 \\ \dot{\Phi}_2 \\ \ddot{x} \\ \ddot{\Phi}_1 \\ \ddot{\Phi}_2 \end{matrix}$$

$$F = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \psi(x) \\ 0 \\ 0 \end{bmatrix} \begin{matrix} \dot{x} \\ \dot{\Phi}_1 \\ \dot{\Phi}_2 \\ \ddot{x} \\ \ddot{\Phi}_1 \\ \ddot{\Phi}_2 \end{matrix}$$

$$\boldsymbol{A = M^{-1}N}$$

$$\boldsymbol{B = M^{-1}F}$$

Composantes de la matrice M

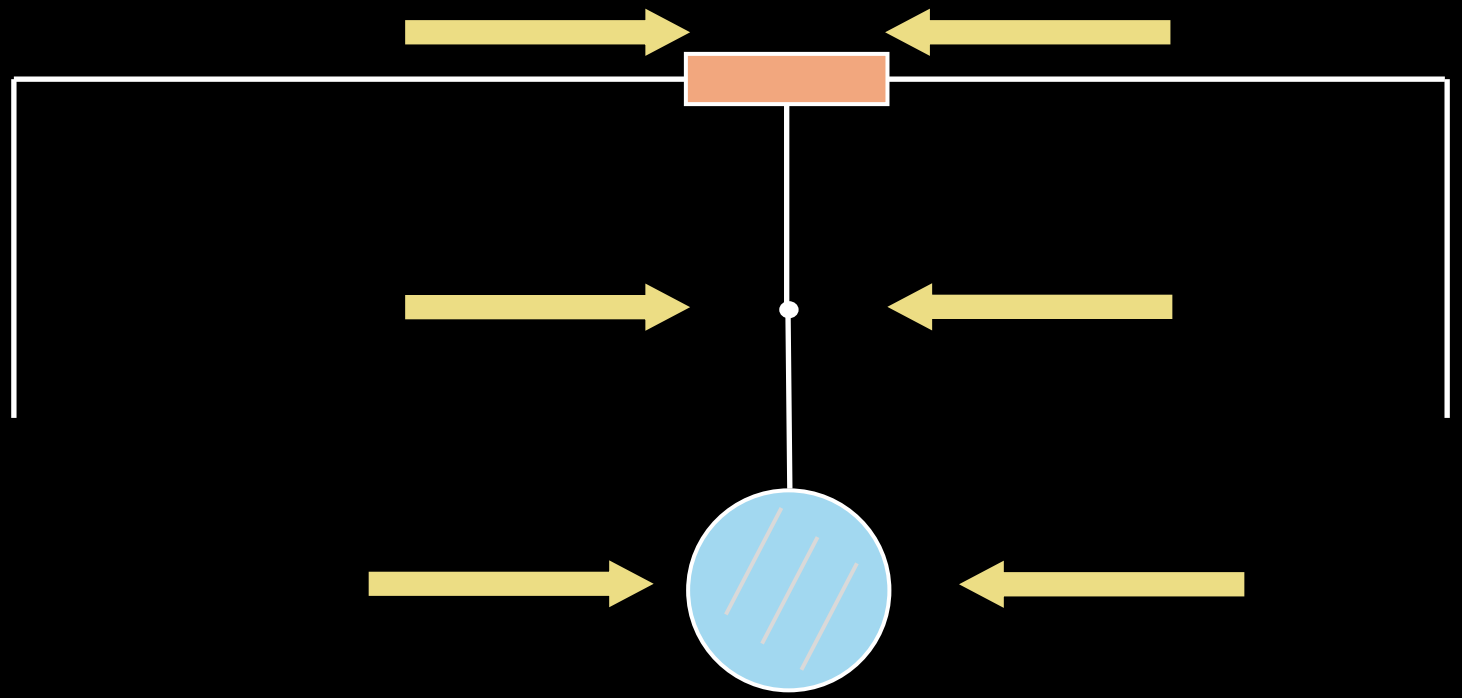
Composantes de la matrice *N*

$$A = M^{-1}N$$

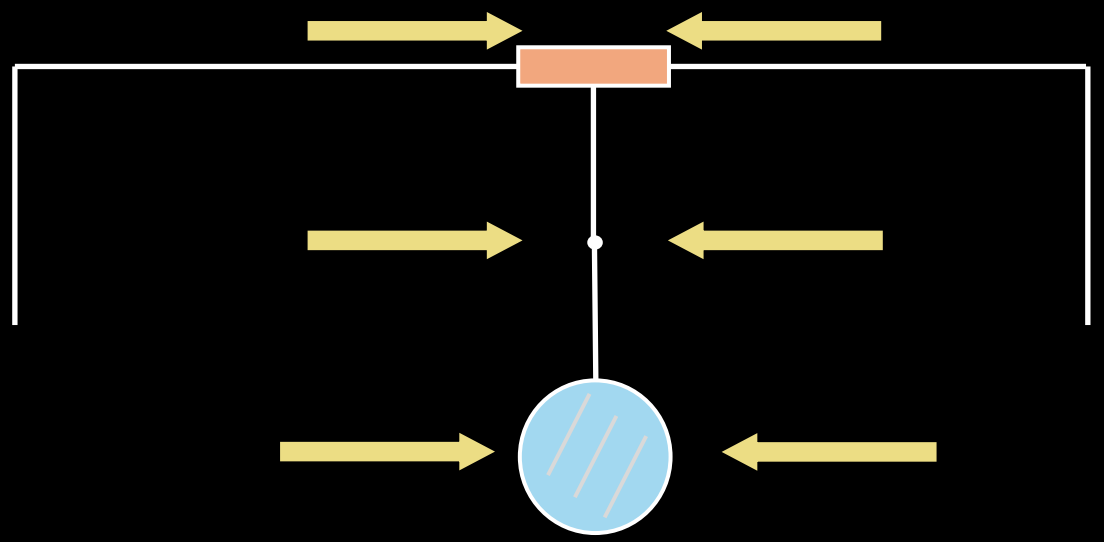
$$B = M^{-1}F$$

```python
def obtained_the_matrix_A_and_B(M, m1, m2, l1, l2):
    # Pour alléger les lignes de codes qui suivent, on écrit les lignes de M dans des variables
distinctes
    line_1_of_M = [1, 0, 0, 0, 0, 0]
    line_2_of_M = [0, 1, 0, 0, 0, 0]
    line_3_of_M = [0, 0, 1, 0, 0, 0]
    line_4_of_M = [0, 0, 0, M + m1 + m2, l1 * (m1 + m2), l2 * m2]
    line_5_of_M = [0, 0, 0, l1 * (m1 + m2), l1 ** 2 * (m1 + m2), l1 * l2 * m2]
    line_6_of_M = [0, 0, 0, l2 * m2, l1 * l2 * m2, l2 ** 2 * m2]

    # On peut maintenant écrire la matrice M
    M = np.array([
        line_1_of_M,
        line_2_of_M,
        line_3_of_M,
        line_4_of_M,
        line_5_of_M,
        line_6_of_M
    ], dtype="float64")
    # On fait de même pour N
    line_1_of_N = [0, 0, 0, 1, 0, 0]
    line_2_of_N = [0, 0, 0, 0, 1, 0]
    line_3_of_N = [0, 0, 0, 0, 0, 1]
    line_4_of_N = [0, 0, 0, 0, 0, 0]
    line_5_of_N = [0, (m1+m2)*l1*g, 0, 0, 0, 0]
    line_6_of_N = [0, 0, m2*l2*g, 0, 0, 0]
    # On peut maintenant écrire la matrice N
    N= np.array([
        line_1_of_N,
        line_2_of_N,
        line_3_of_N,
        line_4_of_N,
        line_5_of_N,
        line_6_of_N
    ], dtype="float64")
    # On écrit la matrice F
    F = np.array([[0, 0, 0, 1, 0, 0]], dtype="float64").T
    # On calcule l'inverse de M
    M_inverted = inv(M)
    # On calcule A et B avec M et N
    A = M_inverted @ N
    B = M_inverted @ F
    return A, B
```

Composantes de la matrice F

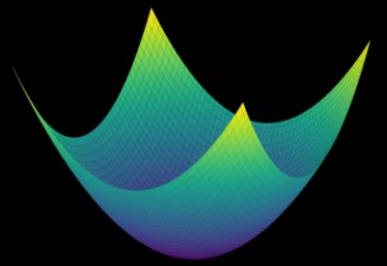


$$u = -Kx$$



$$J(Q, R) = \int_0^{\infty} [X^t(t).Q.X(t) + R.u^2(t)]$$

L'intégrande du **critère** est strictement convexe en u



⇒ Si il existe une commande optimale, alors elle est unique.

$$\boldsymbol{J(Q,\ R)} = \int_0^{\infty} [X^t(t).Q.X(t) + R.u^2(t)]$$

$$Q = \begin{bmatrix} \boldsymbol{Q_x} & 0 & 0 & 0 & 0 & 0 \\ 0 & \boldsymbol{Q_{\Phi_1}} & 0 & 0 & 0 & 0 \\ 0 & 0 & \boldsymbol{Q_{\Phi_2}} & 0 & 0 & 0 \\ 0 & 0 & 0 & \boldsymbol{Q_{\dot{x}}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \boldsymbol{Q_{\dot{\Phi}_1}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \boldsymbol{Q_{\dot{\Phi}_2}} \end{bmatrix}$$

$$R = [\boldsymbol{R_{11}}]$$

✓ Résolution de l'**équation de Ricatti**

✓ $K = R^{-1}B^{T}X$

```python
# On calcule la matrice de gain optimal K
def constructs_a_linear_quadratic_command(A, B, Q, R):
    # On résoud les équations de Riccati en temps continu
    X = np.array(scipy.linalg.solve_continuous_are(A, B, Q, R))
    # On calule K
    K = np.array(scipy.linalg.inv(R) @ (B.T @ X))
    # On calcule les valeurs propres et les vecteurs propres associées à K
    eigen_values, eigen_vectors = scipy.linalg.eig(A - B @ K)
    return K, X, eigen_values

def extract_K(A, B, Q, R):
    # On appelle la fonction responsable de la génération de LQR pour récupérer K
    K, M, eigen_values = constructs_a_linear_quadratic_command(A, B, Q, R)
    return K
```

$$H_s = \begin{bmatrix} \boldsymbol{A - BR^{-1}B^T Q} & \boldsymbol{-A} \\ \boldsymbol{-Q} & 0 \end{bmatrix}$$

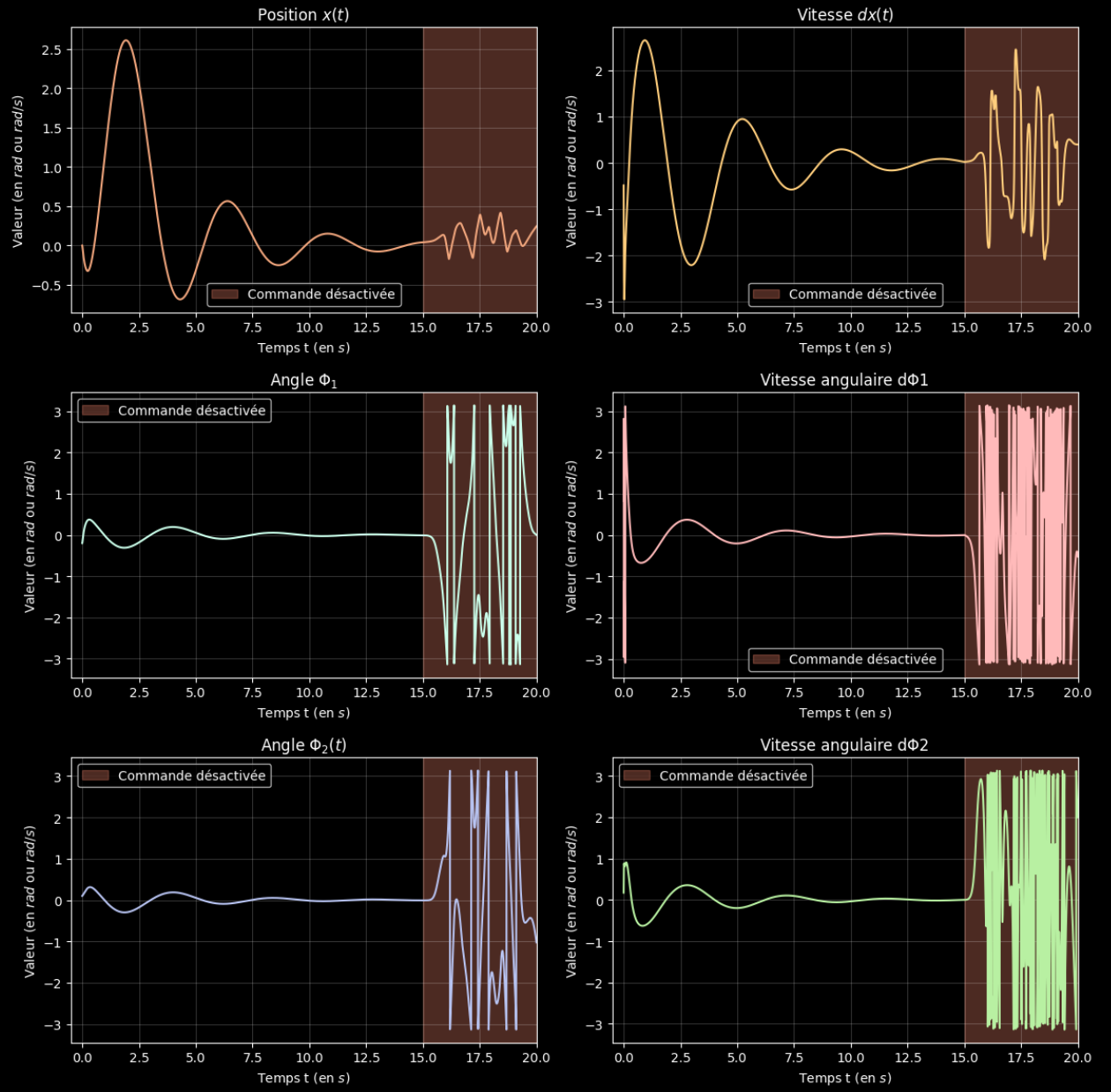
- ✓ H_s est une matrice de Hamilton $\left(H_s = -{H_s}^T\right)$
- ✓ H_s n'as pas de valeur propre imaginaire

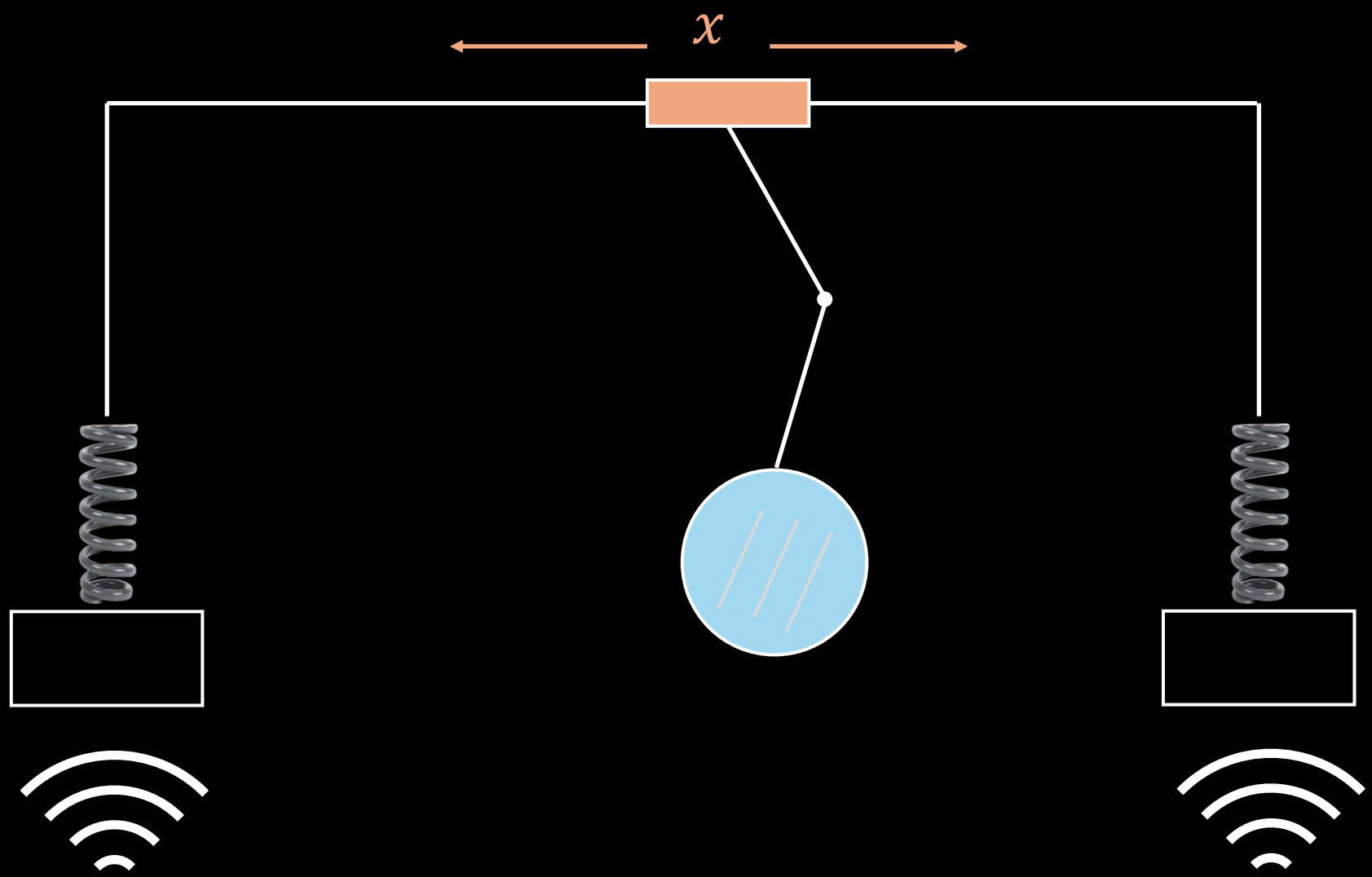
•
•
•

Il existe une unique solution aux équations de Riccati que l'on appelle **commande stabilisante**

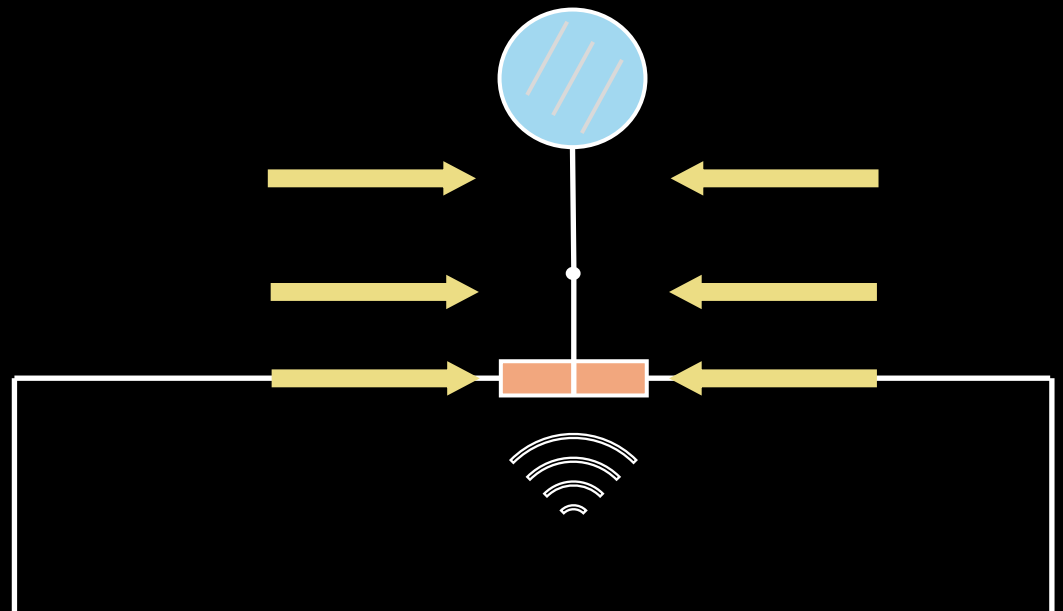
Pour $t < \tau$, la commande est activée.

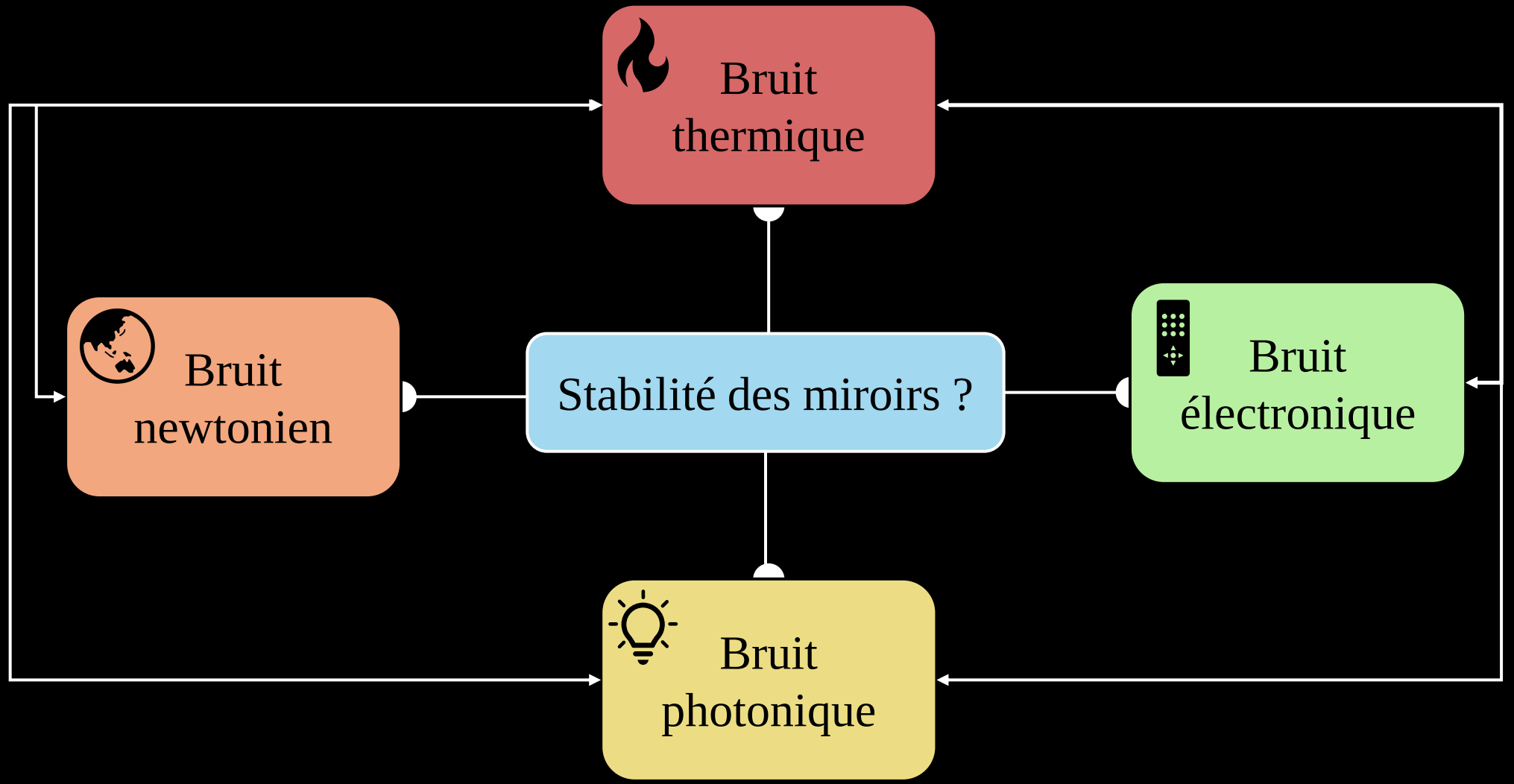
```python
def regulates_the_u_of_the_double_inverted_pendulum(u, t):
    # On écrit l'état du double pendule inversé donné par ses coordonnées généralisées
    x, dx, Φ1, dΦ1, Φ2, dΦ2 = u
    # Si t est < à ce seuil de désactivation de contrôle
    if t < disable_control_threshold:
        # Alors on retourne la commande de contrôle nécessaire pour réguler le système
            # Cette commande se trouve directement dans la fonction qui retourne l'état du système (voir
la cellule suivante),
            # donc retourner l'état revient à activer la commande
        _u = np.array([[x, dx, Φ1, dΦ1, Φ2, dΦ2]])
        return (-K @ _u.T)[0, 0]
    else:
        # Sinon on ne fait rien -> on retourne 0
        return .0
```

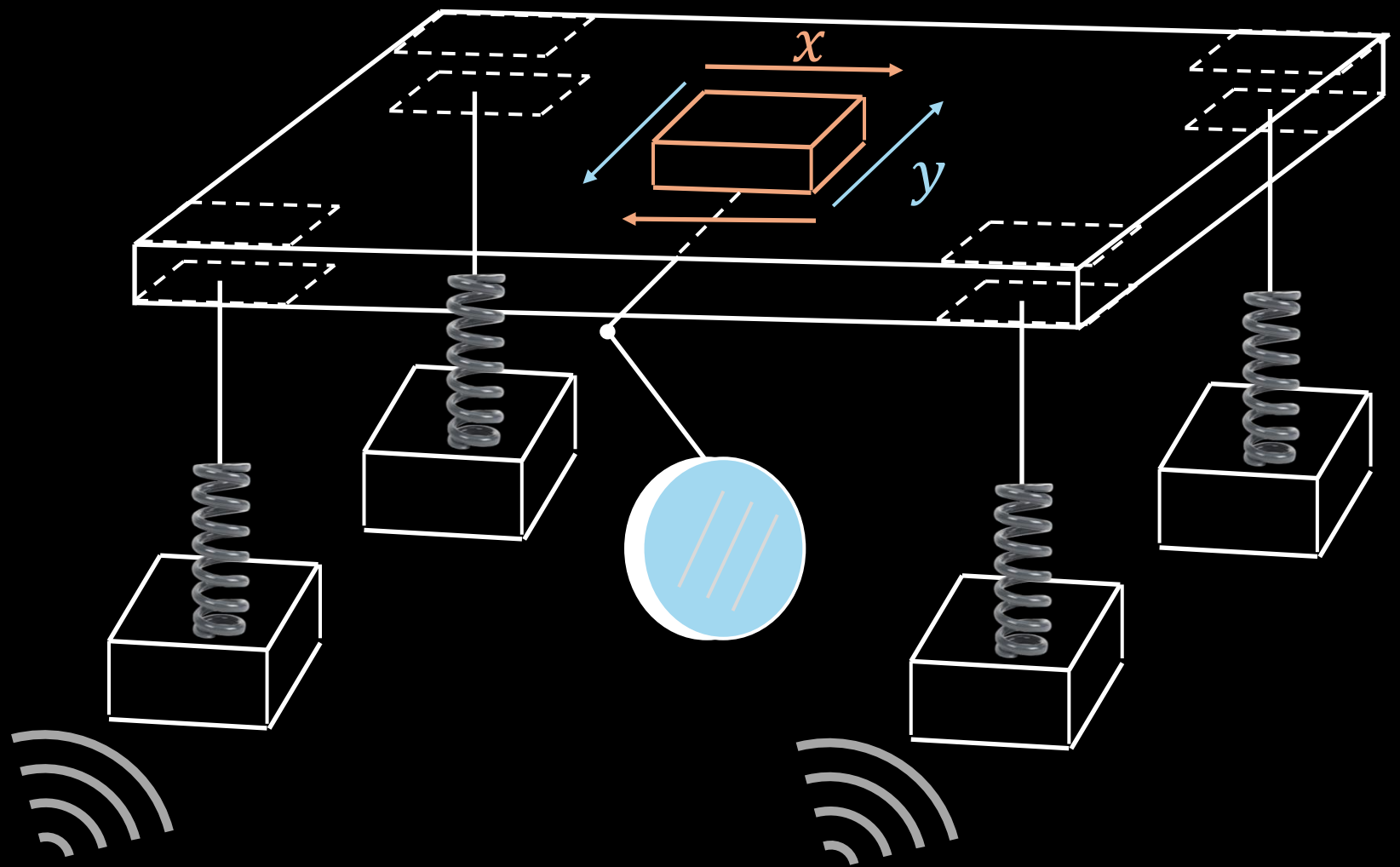

Position $x(t)$
Vitesse $dx(t)$
Angle Φ_1
Vitesse angulaire dΦ1
Angle $\Phi_2(t)$
Vitesse angulaire dΦ2
Commande désactivée
Valeur (en rad ou rad/s)
Temps t (en s)

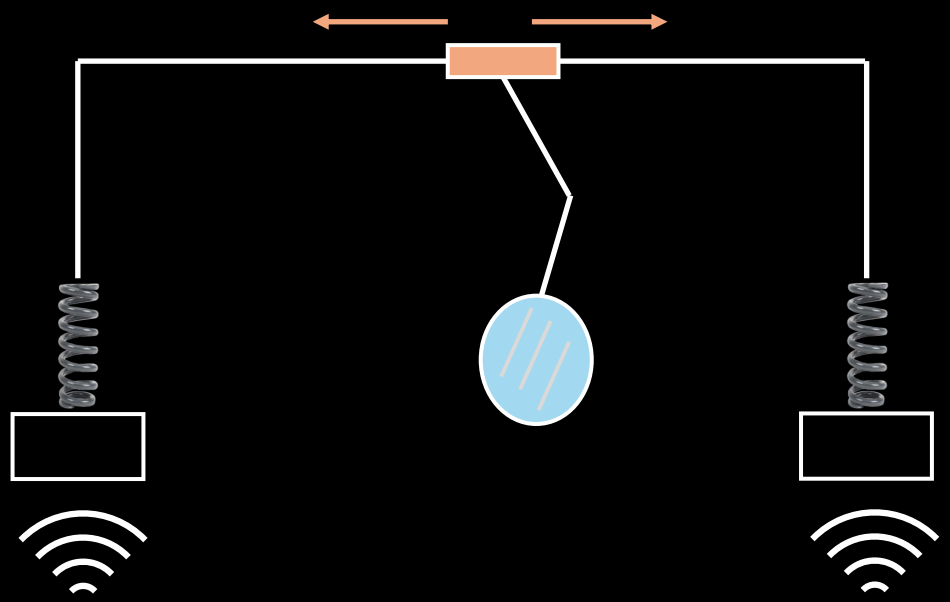

x

Le double pendule agit comme un système de **suspension actif** lorsqu'il est **commandé**

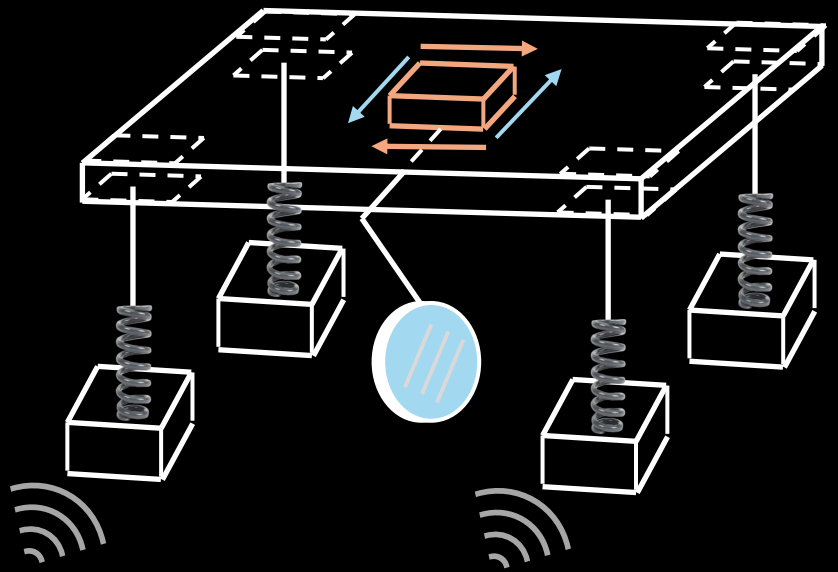



Bruit thermique
Bruit newtonien
Stabilité des miroirs ?
Bruit électronique
Bruit photonique

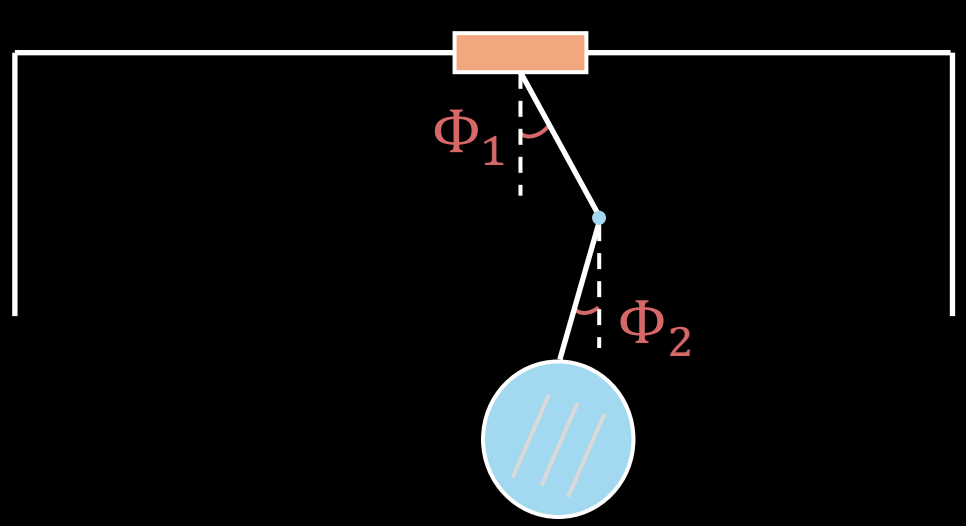

x
y

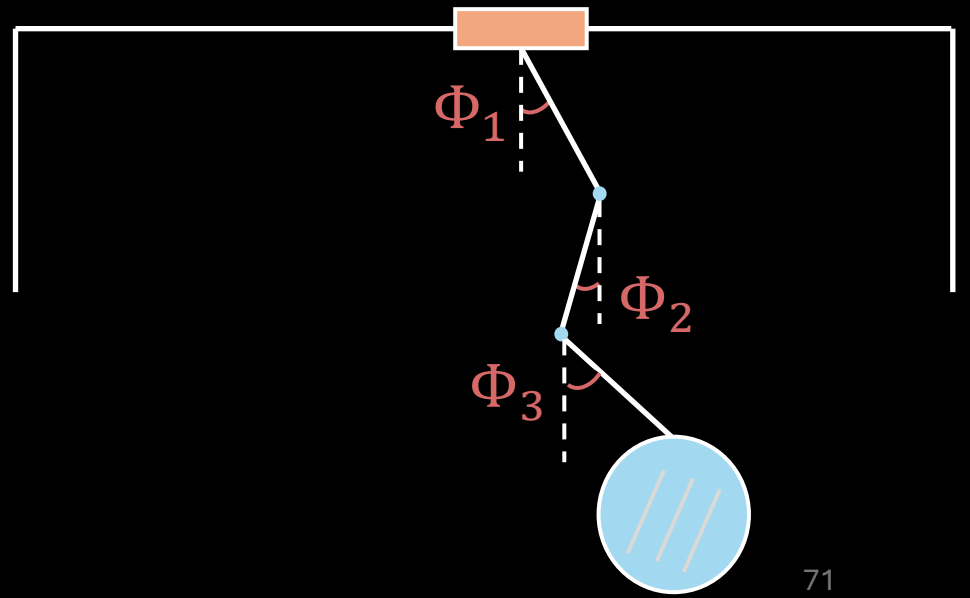


$$u = \begin{bmatrix} x \\ y \\ \Phi_1 \\ \Phi_2 \\ \dot{x} \\ \dot{y} \\ \dot{\Phi}_1 \\ \dot{\Phi}_2 \end{bmatrix}$$

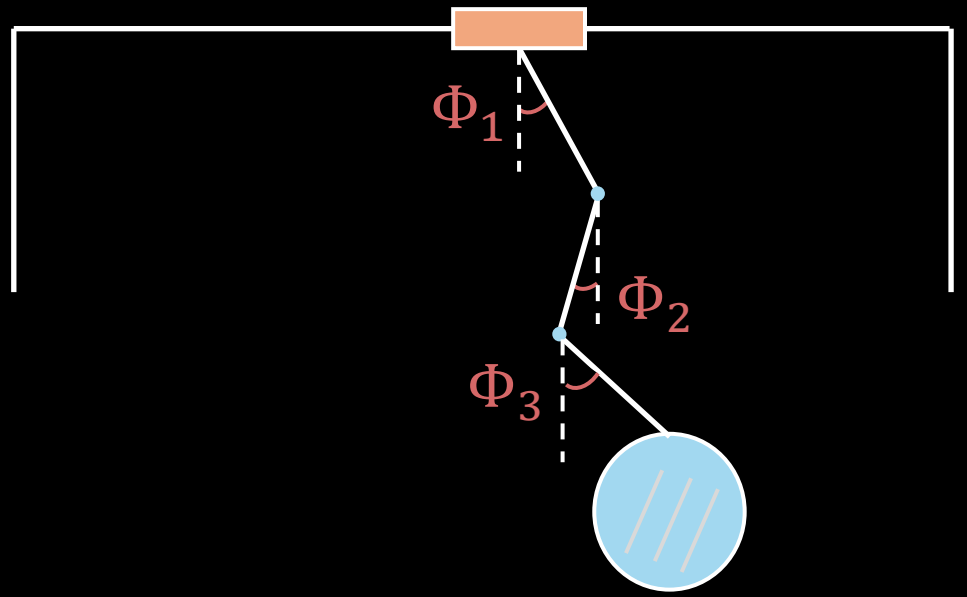


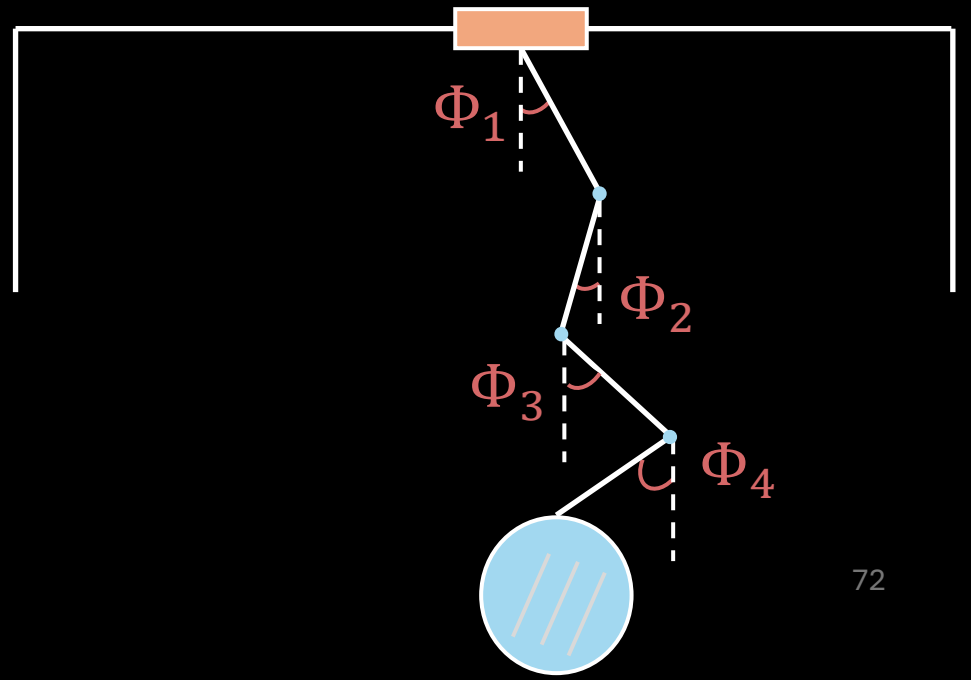
$$u = \begin{bmatrix} x \\ \Phi_1 \\ \Phi_2 \\ \boldsymbol{\Phi_3} \\ \dot{x} \\ \dot{\Phi}_1 \\ \dot{\Phi}_2 \\ \boldsymbol{\dot{\Phi}_3} \end{bmatrix}$$

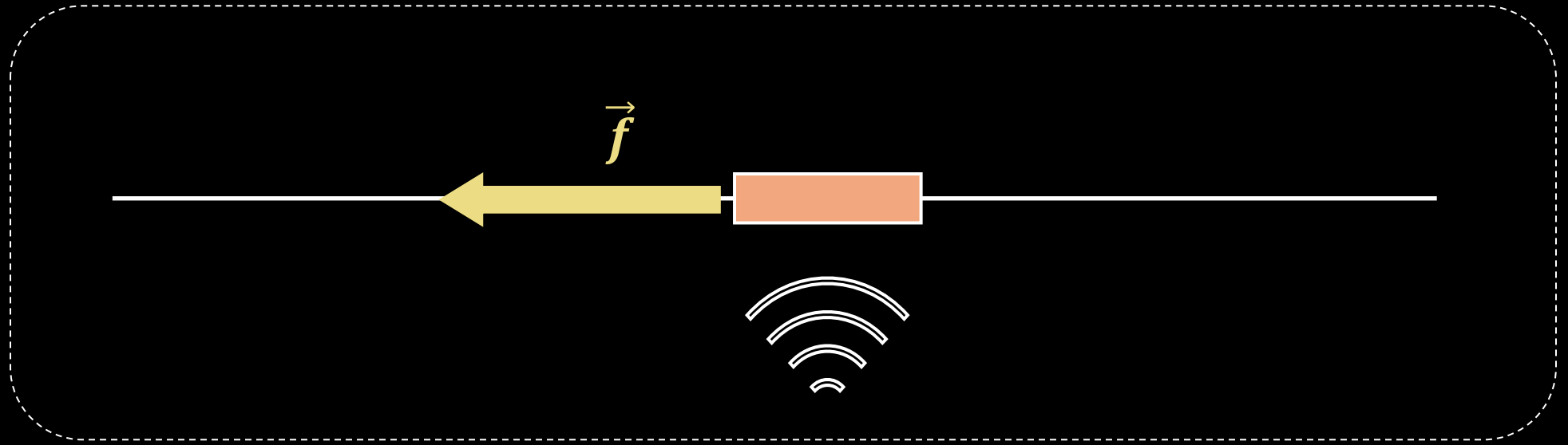


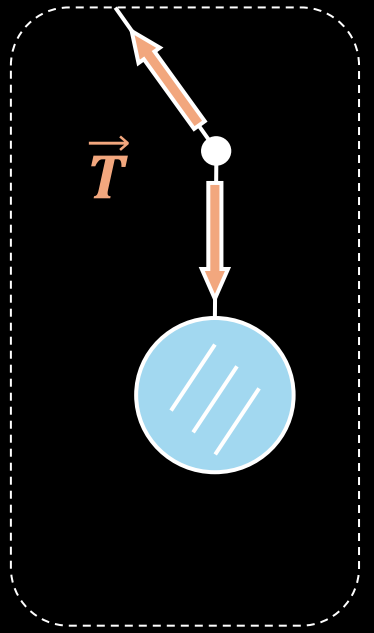


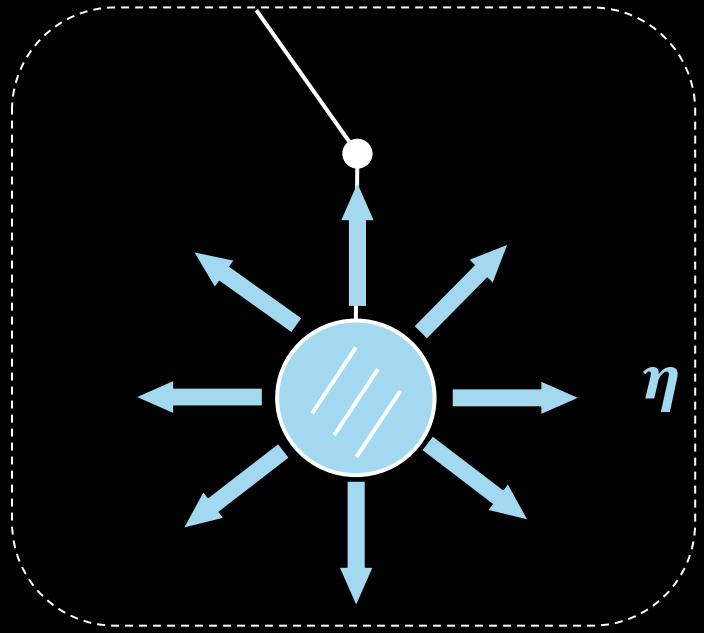
$$u = \begin{bmatrix} x \\ \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \boldsymbol{\Phi_4} \\ \dot{x} \\ \dot{\Phi}_1 \\ \dot{\Phi}_2 \\ \dot{\Phi}_3 \\ \boldsymbol{\dot{\Phi}_4} \end{bmatrix}$$



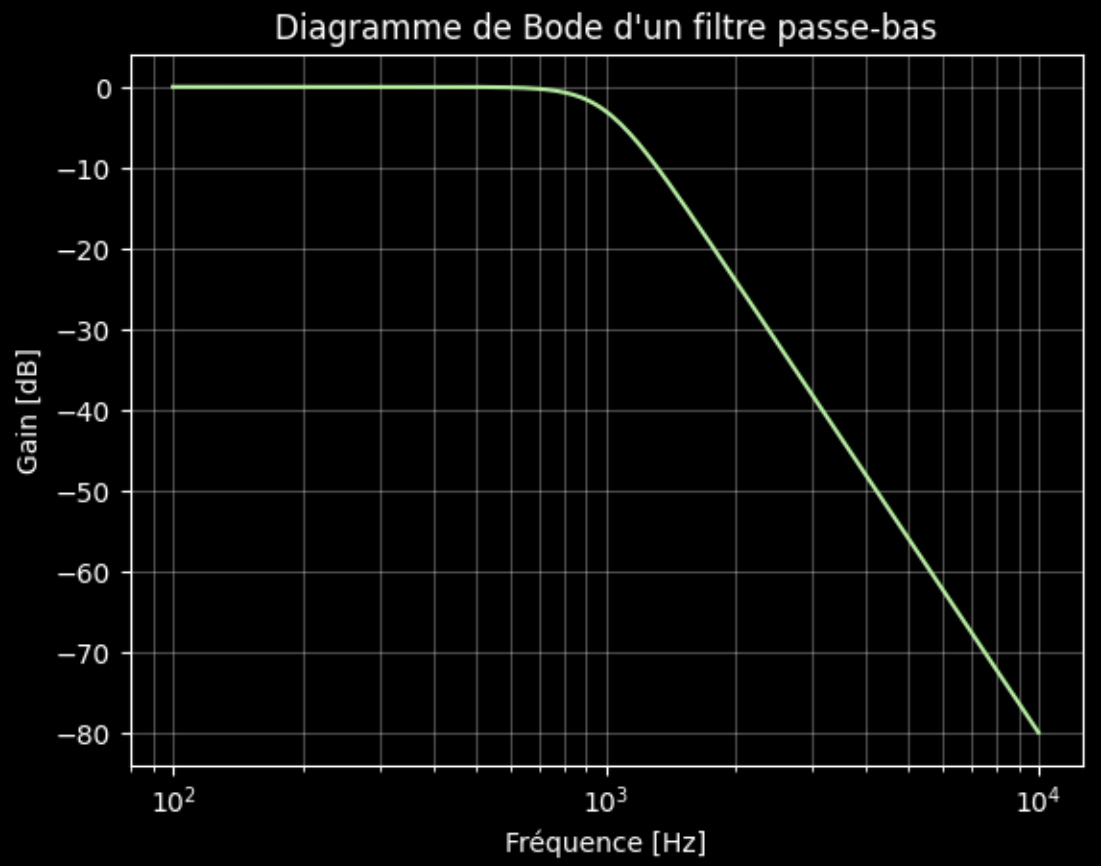


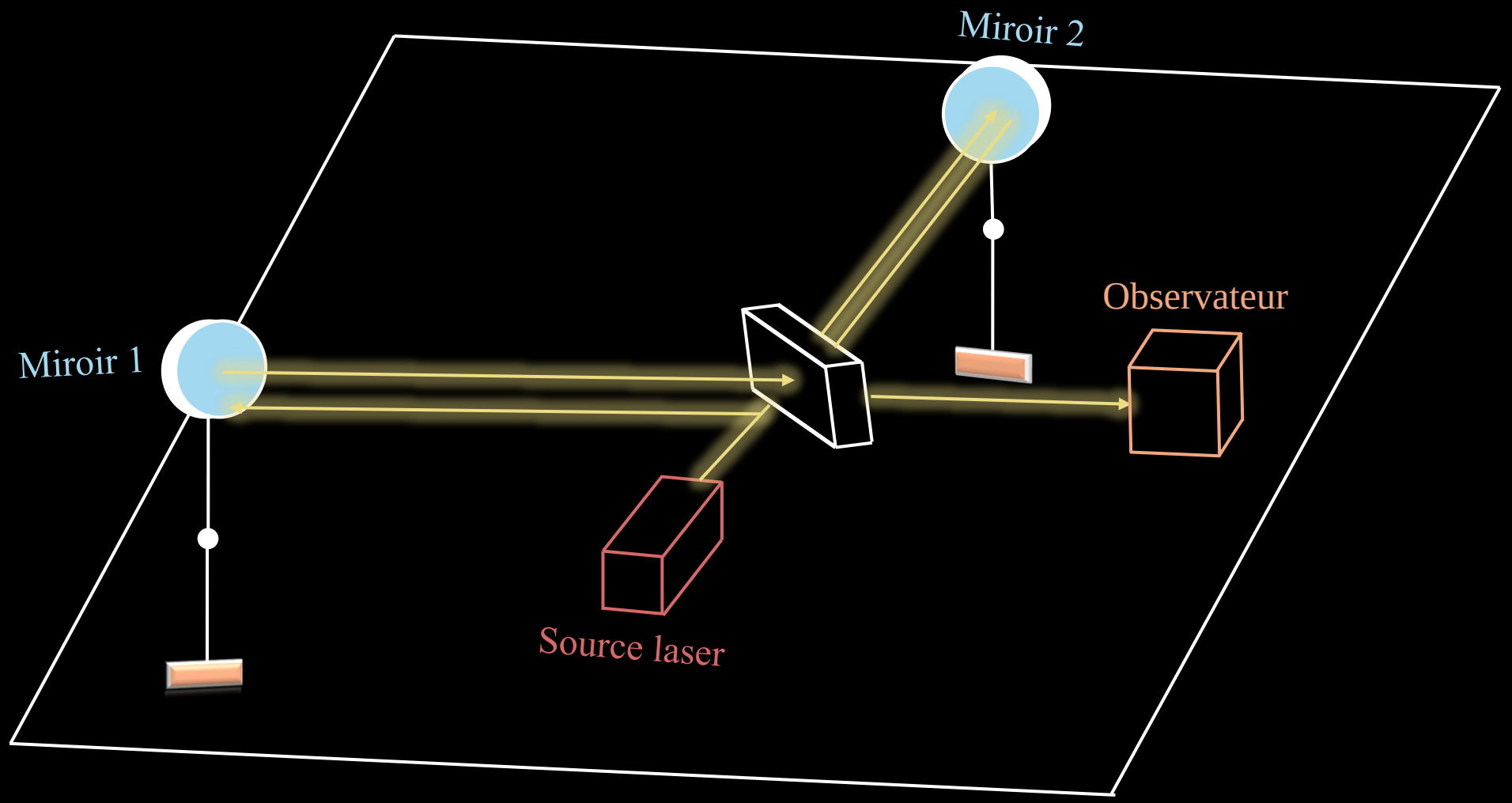

$\vec{f}$


$\vec{T}$


η

$$\boldsymbol{H}(j\omega) = \frac{x_o}{x_i} = \frac{K}{1 - \left[\frac{\omega}{\omega_0}\right]^2 + j\frac{\frac{\omega}{\omega_0}}{Q}}$$




Miroir 2
Miroir 1
Observateur
Source laser

Sources bibliographiques Valentin

- https://www.ligo.caltech.edu/page/ligo-technology
- https://pubs.usgs.gov/of/
- https://www.mdpi.com/2075-4434/10/1/20
- https://www.ligo.caltech.edu/page/ligo-technology?highlight=vacuum
- http://www.lemag.odns.fr/lemagbrunot/Exercices/Lependuledouble.pdf
- https://njaquier.ch/files/PenduleDouble_Jaquier.pdf
- https://perso.univ-lyon1.fr/marc.buffat/COURS/BOOK_OUTILSNUM_HTML/IntroIA/source/IApendule_double/IApendule_double_etu.html
- https://freddy.mudry.org/public/NotesApplications/na_pendinvc.pdf

Sources bibliographiques Ali