

UNIVERSITÉ LIBRE DE BRUXELLES



ECOLE
POLYTECHNIQUE
DE BRUXELLES

TRAN-H101

RAPPORT DE PROJET

Projet Show Laser

Membres du groupe : HANSSENS Dimitri, JOUSTEN Maxence, LINDEMANN Theresa, MATIN Joel, NACHTERGAL Marine, NOUBAYO WOUASSI Yann Ketis, POLLIN Mano, STEINKUHLER Lucas

Chef de projet : KOMINO Alissa

Superviseur : DEBEIR Olivier

Année académique : 2020-2021

Table des matières

Introduction	5
1 Modèle mathématique	6
1.1 Détermination du modèle mathématique	6
1.1.1 Extrémité fixe	8
1.1.2 Extrémité libre	10
1.2 Protocole de détermination des paramètres	12
1.2.1 La pulsation ω	12
1.2.2 L'amplitude a	12
1.2.3 La position de repos l_0	13
1.2.4 Le déphasage ϕ	13
1.2.5 Le coefficient d'amortissement α	13
1.3 Le logiciel de tracking vidéo	15
1.4 Vérification avec Excel	16
2 Systèmes masse-ressort	18
2.1 Les différents systèmes masse-ressort	18
2.1.1 Système 1 : amortissement externe 1	18
2.1.2 Système 2 : amortissement interne	19
2.1.3 Système 3 : amortissement interne modifié	21
2.1.4 Système 4 : amortissement externe 2	22
2.2 Paramètres des systèmes	24
3 Vérification du modèle	25

4 Description du dispositif laser et démarche utilisée	29
5 Recherche des paramètres des galvanomètres	31
5.1 Paramètres des galvanomètres	31
5.1.1 Discussion des résultats	33
5.1.2 Vérification de l'allure des courbes de résonance	35
5.2 Détermination de l'amortissement des galvanomètres	36
5.2.1 Comportement des systèmes à amortissement interne et externe	36
5.2.2 Application aux galvanomètres du projet	37
6 Résultats finaux des figures à tracer	40
6.1 Cercle	40
6.2 Ellipse	41
6.3 Hypotrochoïde	41
6.4 Hypotrochoïde nuage	42
6.5 Figure dynamique	42
6.6 Lemniscate de Bernoulli	43
6.7 Carré	43
7 Programmation	44
7.1 Fonctions de transfert	44
7.2 Fonctions de calcul pour les équations paramétriques	45
7.3 Généralisation à toute forme complexe	46
7.4 Observations des résultats et corrections	47
7.5 Logiciel final	51
Analyse de l'équipe - SWOT	52
Analyse des matériaux	55
Conclusion	57
Annexes	59

A Méthodes de détermination des paramètres des galvanomètres	60
A.1 Première méthode - largeur de la courbe de résonance ($\delta\omega^*$)	60
A.2 Seconde méthode - l'ordonnée à l'origine ($ X _s$)	61
A.3 Troisième méthode - protocole des paramètres	62
B Équations des figures	63
B.1 Cercle	63
B.2 Ellipse	63
B.3 Hypotrochoïde	63
B.4 Figure dynamique	65
B.5 Lemniscate de Bernoulli	67
B.6 Carré	67
C Transfert	71
D Geometry	73
E Read_XML	78
F Optimisation	83
G Interface	89
H Main	91

Abstract

Nombre de mots : 8 445 mots

Les phénomènes oscillatoires ont une grande importance dans la conception et la réalisation de nombreuses constructions. De plus, ils sont omniprésents dans la nature. Les ingénieur.e.s sont donc souvent amené.e.s à étudier leur dynamique et prédire leur mouvement.

Les étudiant.e.s en première année de bachelier d'ingénieur civil ont réalisé.e.s un projet show laser qui repose sur la compréhension de ces phénomènes oscillatoires. En effet, le but final du projet est la projection d'images sur un écran à l'aide d'un dispositif muni de deux galvanomètres et d'un faisceau laser. Ce dispositif présente une dynamique analogue à celle de systèmes oscillatoires. Un modèle mathématique qui décrit la dynamique de ces systèmes a dû être établi et vérifié. Il a ensuite pu être appliqué aux galvanomètres et a permis de prédire leur dynamique.

Pour finir, un logiciel complexe qui calcule les signaux d'entrée à envoyer au dispositif a été réalisé pour obtenir les différentes figures voulues.

Introduction

Cette année, le projet de bac 1 à l'école Polytechnique de Bruxelles consiste à réaliser un show laser. Pour cela, nous avons à notre disposition deux galvanomètres contrôlables à distance, tous deux munis d'un miroir sur lesquels défléchi un faisceau laser. Afin de réaliser des figures avec ce dispositif, il faut imprimer aux galvanomètres une vitesse de rotation suffisamment élevée pour que l'oeil puisse voir les images souhaitées. Or, l'inertie des miroirs due à leur masse et leur vitesse de rotation provoque une force sur la tige de torsion, ce qui engendre un décalage entre la position du miroir souhaitée et le cadre mobile du galvanomètre.

Les galvanomètres forment un système complexe. Il a donc été utile de commencer par étudier la dynamique de systèmes masse-ressort et d'appliquer ensuite cette étude aux galvanomètres. Nous travaillerons sur quatre systèmes masse-ressort différents, chacun avec un amortissement et une disposition qui lui est propre. Dans l'étude de ces systèmes, nous allons déterminer leurs paramètres. Ceux-ci vont être utiles dans la validation du modèle mathématique établi.

Par la suite, nous trouverons les paramètres des deux galvanomètres et déterminerons les différents signaux d'entrée à leur envoyer pour obtenir les formes attendues. Ces signaux seront calculés à l'aide de programmes écrits en langage python.

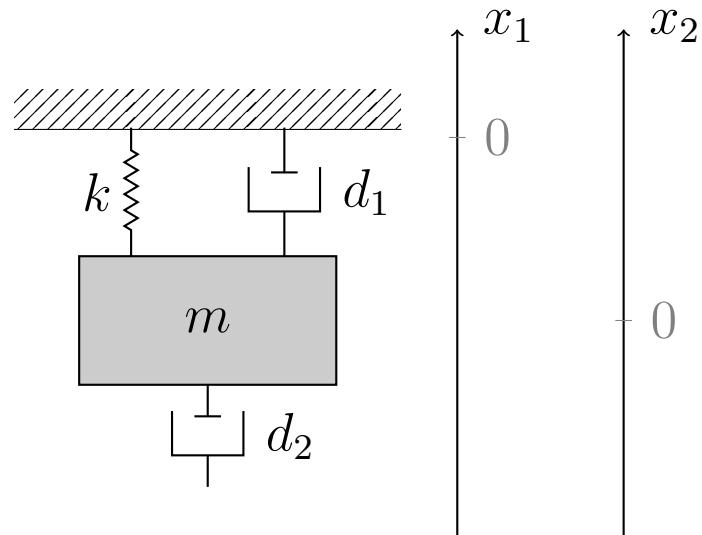
Le but de ce rapport est de détailler les méthodes par lesquelles nous sommes passé.e.s pour arriver au résultat final.

Chapitre 1

Modèle mathématique

1.1 Détermination du modèle mathématique

Nous allons commencer par établir un modèle mathématique. Nous avons adopté une modélisation qui est valable pour chacun de nos systèmes malgré leurs différences. Notons que ce qui suit s'inspire librement du contenu des cours d'Analyse I [4], de Physique générale II [11], de Mécanique rationnelle I [5] et de la vidéo ressource de A. Deraemaeker.



Modélisation du système masse-ressort

Considérons x_1 la position de l'extrémité supérieure du ressort, x_2 la position de la masse m concentrée en un point fixe, k la constante de raideur du ressort, d_1 le coefficient de frottement

interne et d_2 le coefficient de frottement externe. Ici, quatre forces agissent sur la masse, mais considérons uniquement celles dirigées dans le sens opposé à celui de l'axe x_2 . De fait, la force de pesanteur G n'a pour effet qu'une translation sur le système en lui-même, elle peut donc être éliminée. Voici le diagramme du corps libre de la masse :

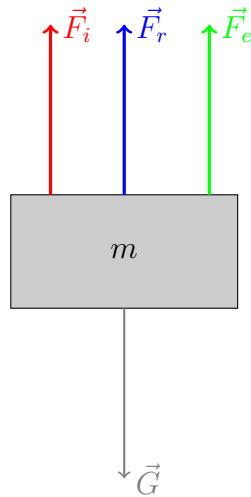


Diagramme du corps libre de la masse

- La **force de rappel** due au ressort, dépendante de l'élongation de ce dernier par rapport à sa longueur libre au repos telle que

$$\vec{F}_r = k(x_2 - x_1)1_{\vec{x}_1}$$

- La **force d'amorti interne** proportionnelle à la vitesse d'élongation du ressort telle que

$$\vec{F}_i = d_1(\dot{x}_2 - \dot{x}_1)1_{\vec{x}_1}$$

- La **force d'amorti externe** proportionnelle à la vitesse d'élongation de la masse telle que

$$\vec{F}_e = d_2(\dot{x}_2)1_{\vec{x}_1}$$

Par le principe fondamental de la dynamique établi par Newton, la résultante des forces appliquées sur la masse s'exprime comme suit :

$$F_R = m\ddot{x}_2$$

Or, par ce qui précède :

$$\vec{F}_R = \vec{F}_i + \vec{F}_e + \vec{F}_r$$

Ceci conduit à l'équation différentielle suivante :

$$m\ddot{x}_2 = -k(x_2 - x_1) - d_1(\dot{x}_2 - \dot{x}_1) - d_2\dot{x}_2$$

En divisant par la masse m , en posant $\beta = \frac{d_1}{2m}$ et $\alpha = \frac{d_2}{2m}$ qui sont respectivement les coefficients d'amortissement interne et externe [3] et $\omega_0 = \sqrt{k/m}$ la pulsation propre de l'oscillateur, on obtient :

$$\ddot{x}_2 + 2\beta(\dot{x}_2 - \dot{x}_1) + 2\alpha\dot{x}_2 + \omega_0^2(x_2 - x_1) = 0$$

1.1.1 Extrémité fixe

En considérant que l'extrémité est fixe, c'est-à-dire $x_1 = 0$, nous obtenons l'équation suivante que nous allons résoudre :

$$\ddot{x}_2 + 2(\alpha + \beta)\dot{x}_2 + \omega_0^2 x_2 = 0$$

Cette équation différentielle est linéaire, du second ordre et à coefficients constants. Suivons la méthode de résolution proposée dans la partie 6.6.2 du cours d'Analyse I [4]. Il faut cependant préciser qu'en fonction du système, α ou β est nul.

$$Ae^{\lambda t}(\lambda^2 + 2\alpha\lambda + \omega_0^2) = 0$$

$$\lambda = -\alpha \pm \sqrt{\alpha^2 - \omega_0^2}$$

La suite du développement dépendra du signe de $\alpha^2 - \omega_0^2$. Dans la suite, seul le cas d'un amortissement faible sera considéré. Par conséquent : $\alpha < \omega_0$, comme l'indique la vidéo ressource fournie par l'université []. Dans le cas de l'amortissement fort, le système considéré n'oscillerait

plus. Le radicande étant dès lors négatif, utilisons les nombres complexes pour poursuivre le développement.

$$\lambda = -\alpha \pm i\sqrt{\omega_0^2 - \alpha^2}$$

$$\implies z_1(t) = Ae^{(-\alpha+i\sqrt{\omega_0^2-\alpha^2})t}, \quad z_2(t) = Ae^{(-\alpha-i\sqrt{\omega_0^2-\alpha^2})t}$$

Nous obtenons donc deux solutions à valeurs dans \mathbb{C} , avec A et B , deux coefficients réels.

Notons que $\Re z_1$ et $\Im z_1$ sont bien deux fonctions réelles linéairement indépendantes. Comme nous l'indique à nouveau le cours d'Analyse I [3], ces deux dernières fonctions étant solutions de l'équation, on sait que :

$$x_2(t) = Ae^{-\alpha t} \cos \sqrt{\omega_0^2 - \alpha^2} t + Be^{-\alpha t} \sin \sqrt{\omega_0^2 - \alpha^2} t$$

Remarquons que l'expression $\sqrt{\omega_0^2 - \alpha^2}$ donne la pulsation ω_d de l'oscillateur amorti.

Pour trouver les valeurs des coefficients réels A et B , aidons-nous des conditions initiales du système, c'est-à-dire $x_2(0) = x_{20}$ et $\dot{x}_2(0) = \dot{x}_{20}$. Il suffit alors de calculer les valeurs de x_2 et de \dot{x}_2 lorsque $t = 0$ et nous trouvons :

$$A = x_{20}$$

$$B = \frac{\dot{x}_{20} + \alpha x_{20}}{\omega_d}$$

L'équation décrivant la position de la masse dans nos systèmes est donc la suivante :

$$x_2(t) = e^{-\alpha t} \left(x_0 \cos \omega_d t + \frac{\dot{x}_0 + \alpha x_0}{\omega_d} \sin \omega_d t \right) \quad (1.1)$$

L'équation peut également être écrite sous cette forme plus compacte qui se retrouve respectivement dans les chapitres 4 et 5 des cours de Mécanique rationnelle I [5] et de Physique générale II [11] :

$$x_2(t) = a \cos(\omega_d t + \phi) e^{-\alpha t} \quad (1.2)$$

où

$$\tan \phi = \frac{-\dot{x}_0 - \alpha x_0}{\omega_d}$$

$$a = \sqrt{x_0^2 + \left(\frac{\dot{x}_0 + \alpha x_0}{\omega_d} \right)^2}$$

1.1.2 Extrémité libre

À présent, considérons le cas où l'extrémité n'est pas fixe. Repartons alors de l'équation :

$$m\ddot{x}_2 = -k(x_2 - x_1) - d_1(\dot{x}_2 - \dot{x}_1) - d_2\dot{x}_2$$

La position de la masse x_2 peut être exprimée en fonction de la position de l'extrémité supérieure x_1 .

Utilisons dès lors les phaseurs, comme expliqué dans le cours de Physique générale II [11] pour représenter x_1 et x_2 . Soit

$$\xi_1(t) = x_{10}e^{i\omega t}, \quad \xi_2(t) = x_{20}e^{i\omega t}$$

$$x_1(t) = \text{Re}[\underline{X}_1 e^{i\omega t}], \quad x_2(t) = \text{Re}[\underline{X}_2 e^{i\omega t}]$$

Précisons que la pulsation ω est la même pour x_1 et x_2 car les deux points sont soumis à la même fréquence aussi bien dans notre excitation du système masse-ressort que dans le fonctionnement du galvanomètre. Ensuite, séparons notre étude du système, en un cas où l'amortissement est externe ($d_1 = 0$) et un autre où l'amortissement est interne ($d_2 = 0$).

1) $d_1 = 0$

$$\ddot{x}_2 + \frac{k}{m}x_2 - \frac{k}{m}x_1 + \frac{d_2}{m}\dot{x}_2 = 0$$

$$\ddot{x}_2 + 2\alpha\dot{x}_2 + \omega_0^2x_2 - \omega_0^2x_1 = 0$$

$$\dot{x}_2 = \text{Re}[i\omega\underline{X}_2 e^{i\omega t}]$$

$$\ddot{x}_2 = \text{Re}[-\omega^2\underline{X}_2 e^{i\omega t}]$$

$$\text{Re}[-\omega^2\underline{X}_2 e^{i\omega t} + 2\alpha i\omega\underline{X}_2 e^{i\omega t} + \omega_0^2\underline{X}_2 e^{i\omega t}] = \text{Re}[\omega_0^2\underline{X}_1 e^{i\omega t}]$$

$$\text{Re}[e^{i\omega t}(\underline{X}_2[-\omega^2 + 2\alpha i\omega + \omega_0^2])] = \text{Re}[e^{i\omega t}(\underline{X}_1\omega_0^2)]$$

$$\underline{X}_1 = \frac{-\omega^2 + 2\alpha i\omega + \omega_0^2}{\omega_0^2}\underline{X}_2$$

Désormais, considérons l'amortissement interne :

2) $d_2 = 0$

$$\ddot{x}_2 + \frac{k}{m}x_2 - \frac{k}{m}x_1 - \frac{d_1}{m}\dot{x}_1 + \frac{d_1}{m}\dot{x}_2 = 0$$

$$\ddot{x}_2 + 2\beta\dot{x}_2 + \omega_0^2x_2 = 2\beta\dot{x}_1 + \omega_0^2x_1$$

$$\dot{x}_2 = Re[i\omega\underline{X}_2 e^{i\omega t}]$$

$$\ddot{x}_2 = Re[-\omega^2\underline{X}_2 e^{i\omega t}]$$

$$\dot{x}_1 = Re[i\omega\underline{X}_1 e^{i\omega t}]$$

$$Re[-\omega^2\underline{X}_2 e^{i\omega t} + 2\beta i\omega\underline{X}_2 e^{i\omega t} + \omega_0^2\underline{X}_2 e^{i\omega t}] = Re[2\beta i\omega\underline{X}_1 e^{i\omega t} + \omega_0^2\underline{X}_1 e^{i\omega t}]$$

$$Re[e^{i\omega t}(\underline{X}_2[-\omega^2 + 2\beta i\omega + \omega_0^2])] = Re[e^{i\omega t}\underline{X}_1(2\beta i\omega + \omega_0^2)]$$

$$\underline{X}_1 = \frac{-\omega^2 + 2\beta i\omega + \omega_0^2}{2\beta i\omega + \omega_0^2}\underline{X}_2$$

On en déduit les équations des positions de $x_2(t)$ et de $x_1(t)$. Pour vérifier cette relation, trouvons d'abord les différents paramètres relatifs à notre système et ensuite comparons nos résultats expérimentaux avec la théorie.

$$k = \frac{mg}{x_{20} - l_0} \quad (l_0 \text{ étant la longueur libre du ressort})$$

$$\omega_d = \frac{2\pi}{T_d}$$

$$\frac{a\sin(\omega_d t_1 + \phi)e^{-\alpha t_1}}{a\sin(\omega_d t_2 + \phi)e^{-\alpha t_2}} = \Delta x_2 \quad \Leftrightarrow \quad \frac{e^{-\alpha t_1}}{e^{-\alpha t_2}} = \Delta x_2 \quad \Leftrightarrow \quad e^{-\alpha t_1 + \alpha t_2} = \Delta x_2$$

$$\Leftrightarrow \quad \alpha = \frac{\ln(\Delta x_2)}{t_2 - t_1}$$

1.2 Protocole de détermination des paramètres

Nous allons désormais détailler un protocole permettant de trouver les paramètres de l'équation de la position de la masse en fonction du temps que voici :

$$x_2(t) = a \cos(\omega_d t + \phi) e^{-\alpha t}$$

1.2.1 La pulsation ω

La période amortie T_d correspond à la durée en secondes d'une oscillation complète de notre ressort et ce grâce aux graphes du logiciel de tracking. Nous ferons cependant attention à mesurer la durée de plusieurs oscillations pour ensuite faire une moyenne de celles-ci. La pulsation amortie ω_d est obtenue à partir de la pseudo-période T_d via la formule suivante :

$$\omega_d = \frac{2\pi}{T_d}$$

Connaissant la constante de raideur du ressort et la masse qu'on y a attachée, on peut trouver la pulsation propre du ressort (la pulsation sans amortissement).

$$\omega_0 = \sqrt{\frac{k}{m}}$$

1.2.2 L'amplitude a

L'amplitude doit être mesurée expérimentalement, directement sur le modèle ou via le graphique obtenu grâce au logiciel de tracking en pixels. Cependant, si nous travaillons avec les pixels, on ne sait pas comparer les graphes entre eux à cause d'une différence de point de vue. Dès lors, pour connaître l'amplitude, il faut mesurer la hauteur du premier maximum, et en soustraire la position au repos de la masse.

1.2.3 La position de repos l_0

En mesurant la différence de longueur entre le ressort avec et sans masse suspendue, on peut calculer la constante de raideur k grâce à la formule suivante :

$$k = \frac{mg}{x_{20} - l_0} \quad (l_0 \text{ étant la longueur libre du ressort})$$

1.2.4 Le déphasage ϕ

A un déphasage ϕ correspond un décalage temporel de ϕ/ω_d (vers la droite si $\phi < 0$ et vers la gauche si $\phi > 0$). Ainsi, il suffit de multiplier l'abscisse du premier maximum par la fréquence amortie ω_d pour trouver le déphasage.

1.2.5 Le coefficient d'amortissement α

Pour trouver le coefficient d'amortissement à partir du graphique obtenu via le logiciel de tracking, nous avons trouvé deux méthodes. Si la première utilise le rapport entre l'amplitude de l'oscillation amortie et celle de l'oscillation non-amortie, la deuxième méthode utilise le rapport entre deux maxima de l'oscillation amortie. Nous travaillerons dans cette section en considérant que le graphique est centré sur l'axe horizontal, donc que le l_0 correspond à la hauteur 0 de x_2 .

Première méthode

Les propriétés des fonctions périodiques seront utilisées pour cette méthode. Pour calculer l'amortissement du système, il faut effectuer des opérations à partir des trois fonctions suivantes :

1. Le signal périodique du mouvement (sans le caractère amortissant) :

$$f(t) = a \cos(\omega_d t + \phi)$$

2. La fonction amortie, qui vient se greffer à la fonction $f(t)$ pour donner l'équation du système amorti :

$$g(t) = e^{-\alpha t}$$

3. Enfin, pour donner l'équation de la position en fonction du temps de notre système amorti :

$$x_2(t) = f(t)g(t) = a \cos(\omega_d t + \phi) e^{-\alpha t}$$

Comme nous venons de trouver tous les autres paramètres de cette équation, il ne reste plus qu'à déterminer le α , qui peut être trouvé via la relation suivante :

$$g(t) = \frac{x_2(t)}{f(t)}$$

Il suffit de prendre une valeur notée M correspondant à l'abscisse des maximums des deux fonctions respectives $f(t)$ et $x_2(t)$ pour trouver la valeur de $g(M)$ et ainsi déterminer α . Il est nécessaire de préciser que nous ne tenons pas compte ici du léger décalage entre les maxima de $f(t)$ et $x_2(t)$ occasionné par le caractère décroissant de $g(t)$. On a en effet pu observer que tous les maxima de h sont légèrement décalés vers la gauche par rapport aux maxima de f .

Comment déterminer M ?

Pour déterminer M , il suffit d'utiliser les propriétés de la fonction cosinus. Ainsi, on sait que :

$$M = \frac{2\pi}{\omega_d} k - \frac{\phi}{\omega_d} \quad (k \in \mathbb{Z})$$

Une fois un maximant trouvé (le premier par exemple), il suffit de le remplacer dans l'équation ci-dessus pour trouver :

$$e^{-\alpha M} = \frac{x_2(M)}{f(M)} \quad \Leftrightarrow \quad \alpha = -\frac{\ln(\frac{x_2(M)}{f(M)})}{M} \quad \text{où} \quad f(M) = a$$

D'un point de vue pratique, si le graphe n'est pas centré sur l'axe vertical tel que $x_{20} = 0$ (comme considéré depuis le début de cette partie du protocole), il faut tenir compte de la position d'origine x_{20} de la masse sur le graphique, cela donne :

$$\alpha = -\frac{\ln(\frac{x_2(M)-x_{20}}{f(M)-x_{20}})}{M} \quad \text{où} \quad f(M) - x_{20} = a$$

Seconde méthode

Dans cette deuxième méthode, nous nous servons de deux maxima distincts de la fonction $x_2(t)$ (ayant pour abscisses : t_1 et t_2). Notons le rapport entre ces maxima Δx_2 , ainsi on obtient :

$$\begin{aligned} \frac{\sin(\omega_d t_1 + \phi) e^{-\alpha t_1}}{\sin(\omega_d t_2 + \phi) e^{-\alpha t_2}} = \Delta x_2 &\Leftrightarrow \frac{e^{-\alpha t_1}}{e^{-\alpha t_2}} = \Delta x_2 \Leftrightarrow e^{-\alpha t_1 + \alpha t_2} = \Delta x_2 \\ &\Leftrightarrow \alpha = \frac{\ln(\Delta x_2)}{t_2 - t_1} \end{aligned}$$

Cette technique, a priori plus courte, est tout aussi efficace que la première détaillée plus haut. Nous avons désormais deux moyens pour trouver le paramètre α , ce qui nous permet d'avoir davantage de précision (nous verrons plus loin que les résultats obtenus via ces deux méthodes sont semblables).

De plus, nous connaissons l'équation suivante qui sera utilisée plus tard pour valider le modèle et pour trouver la pulsation propre des galvanomètres :

$$\omega_d^2 = \omega_0^2 - \alpha^2$$

$$\alpha = \pm \sqrt{\omega_0^2 - \omega_d^2}$$

1.3 Le logiciel de tracking vidéo

Dans la suite du rapport, un logiciel de tracking vidéo sera souvent mentionné. Il est donc important de définir ce qu'est et ce que fait ce dernier.

Ce logiciel nous a été fourni par l'ULB et permet, grâce à deux fichiers *cv2_tracking1.py* et *cv2_tracking2.py*, de tracker sur une vidéo soit un soit plusieurs points en mouvement. Lors de l'utilisation de ce programme, certaines difficultés ont été rencontrées avant de parvenir à obtenir des graphes suffisamment précis, principalement dues à la sensibilité du détecteur aux couleurs chaudes. C'est pourquoi nous avons du faire attention à maximiser le contraste de couleur entre les points trackés et l'arrière plan.

Une fois le tracking correctement effectué, ce logiciel renvoie un graphique donnant l'allure du déplacement de l'objet tracké et, le plus important, un fichier .csv avec un tableau des données contenant la position (en x et en y) en fonction du temps. Une autre particularité de ce logiciel est qu'il ramène toutes les vidéos à une fréquence d'images de 60 frames. Ainsi, il faut faire attention à la quantité de frames par seconde dans la vidéo originale pour pouvoir reconvertis les frames du tableau de valeurs obtenu en secondes.

Quant aux notions de distances, le logiciel ne permet pas d'exprimer les distances en cm, il les exprime en pixels. Unité malheureusement impossible à convertir en centimètres au travers d'une vidéo.

1.4 Vérification avec Excel

Pour valider les paramètres de notre système, nous avons utilisé Excel. Grâce à un module spécifique nommé « Solver », nous avons pu d'une manière tout à fait précise approcher les valeurs des paramètres de nos différents systèmes. À titre d'exemple, voici la courbe approchée d'un de nos systèmes par Excel :

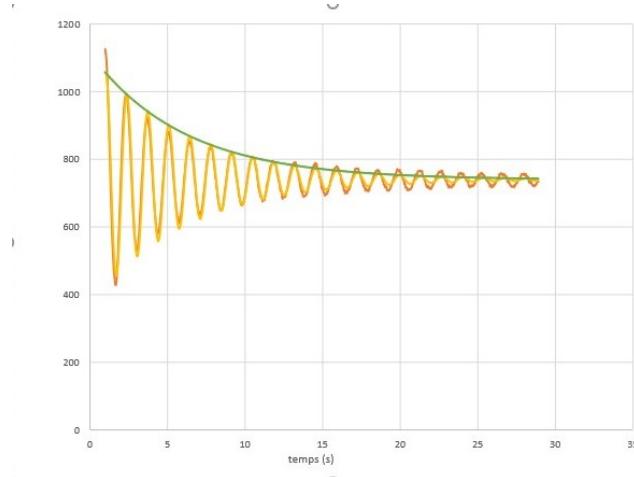


FIGURE 1.1 – Position verticale (y) et horizontale (x) en fonction du temps (en secondes)

Sur ce graphique, nous voyons en **orange** la courbe obtenue expérimentalement issue du logiciel de tracking. Et en **jaune**, la courbe des points fournis par le module "Solver". Ce module a pour fonction de trouver les valeurs des paramètres d'une équation (dont nous lui donnons la forme) de telle sorte à ce que l'écart entre cette fonction et le tableau de valeur qu'on lui fournit soit réduit au maximum. Nous avons procédé par la méthode des moindres carrés pour parvenir à réduire au maximum l'écart entre la courbe théorique obtenue par Excel et la courbe expérimentale.

Nous pouvons constater que l'écart entre la courbe théorique et la courbe obtenue de manière expérimentale est minime. Ainsi, la valeur des paramètres de l'équation de la courbe théorique étant fournis avec une grande précision, nous pouvons vérifier que les paramètres de l'équation en fonction du temps de notre masse obtenus via le protocole présenté ci-dessus sont corrects et précis. Pour faciliter la comparaison, nous avons érigé deux tableaux reprenant les valeurs des paramètres de l'équation 1.2 trouvés via le protocole mathématique et les valeurs de ceux-ci trouvés via Excel.

Chapitre 2

Systèmes masse-ressort

2.1 Les différents systèmes masse-ressort

Nous avons construit 4 systèmes masse-ressort différents. Deux d'entre eux possèdent un amortissement externe et les deux autres un amortissement interne. L'expérience du tiré- lâché a été effectuée pour les 4 systèmes et leurs paramètres ont ensuite pu être calculés.

2.1.1 Système 1 : amortissement externe 1

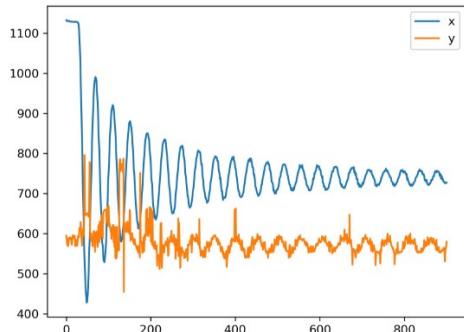
Ce premier système est à amortissement externe. Il est constitué d'une masse suspendue à un ressort et d'une tige qui est plongée dans de l'eau et reliée à cette masse. Cette configuration a été choisie pour diminuer au maximum la poussée d'Archimède en optimisant la surface plongée dans l'eau. Un statif a été construit pour supporter le système.



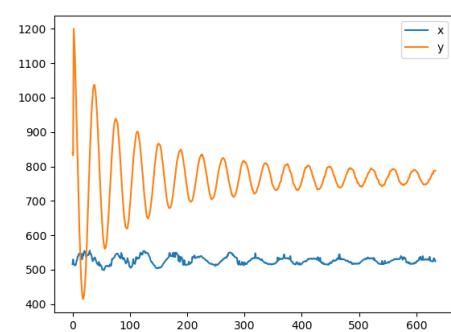
FIGURE 2.1 – Système 1

Sur le bouchon de la bouteille se trouve un petit carré rouge facilement repérable par

le logiciel de tracking vidéo. Nous avons effectué plusieurs fois l'expérience et avons tenté d'obtenir un graphe du mouvement de la masse le plus précis possible. Après quelques dizaines de tentatives, voyant que la légère et permanente oscillation latérale était inévitable, nous nous sommes résolus au meilleur graphique obtenu (la figure 2.2).



Première tentative (2.1)



tentative finale (2.2)

FIGURE 2.2 – Position verticale (y) et horizontale (x) (en pixels) en fonction du temps (en frame)

Notons tout de même qu'à cause de la constante de rigidité trop faible de notre ressort, nous ne sommes pas parvenus à obtenir une fréquence comprise entre 1 et 3 Hz. Nous aurions pu, pour augmenter la fréquence, diminuer la masse de l'objet suspendu au ressort. Mais cela aurait fortement augmenté l'amortissement du système, nous empêchant d'obtenir un nombre suffisant d'oscillations.

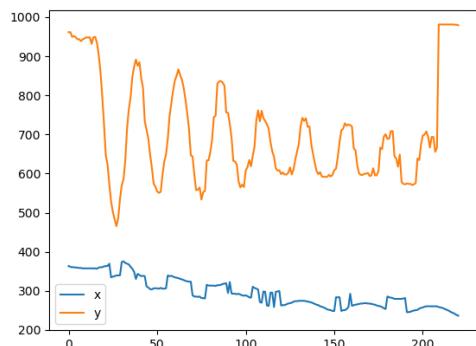
2.1.2 Système 2 : amortissement interne

Pour ce deuxième système, nous avons remplacé le ressort métallique par un élastique en caoutchouc. Cela a permis d'obtenir un amortissement interne.

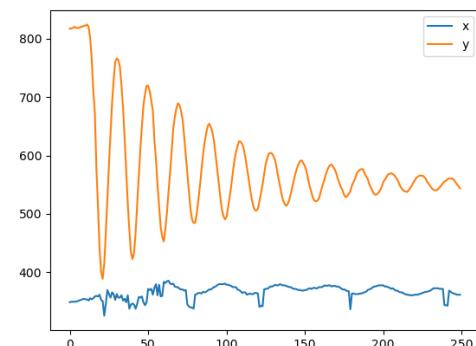


FIGURE 2.3 – Système 2

Nous avons repris le même support que le système 1. Nous avons choisi une bouteille d'eau de 50 cl comme masse, ainsi nous pouvons modifier son poids à volonté. Nous n'avons pas obtenu de bons graphes lors des premières tentatives (voir figure (2.3)). Le problème venait du tracking vidéo. Pour faciliter celui-ci, tout le système a été placé devant un fond noir. La masse a également été remplacée par une bouteille de 33 cl pour stabiliser le système. Voici les graphes obtenus :



Première tentative (2.3)



tentative finale (2.4)

FIGURE 2.4 – Position verticale (y) et horizontale (x) (en pixels) en fonction du temps (en frame)

2.1.3 Système 3 : amortissement interne modifié

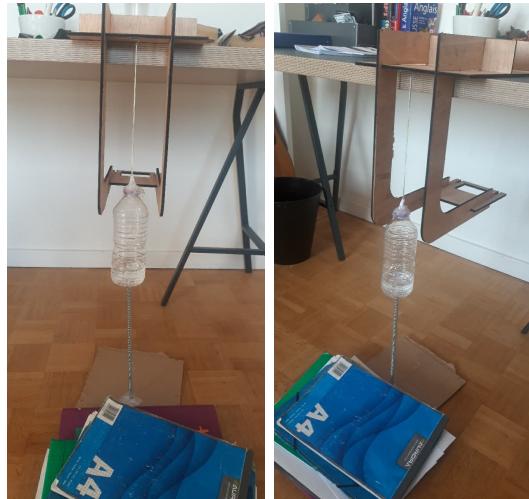


FIGURE 2.5 – Système 3

Ce troisième système est constitué d'un amortissement principalement interne dû à un élastique classique en caoutchouc que nous avons fixé à un support. Il se différencie du deuxième système puisqu'un ressort relie également la masse au sol. Cette modification nous a permis de gagner en stabilité et d'étudier la différence de comportement avec le système 2 lors du tiré-lâché.

Comme masse, nous avons à nouveau utilisé une bouteille d'eau de 33 cl et le statif utilisé nous a été mis à disposition par l'université

Différentes modifications ont été effectuées sur le système au cours des tentatives. Il a entre autre été placé devant un fond noir. Voici l'un des premiers graphiques obtenu en comparaison avec un des derniers :

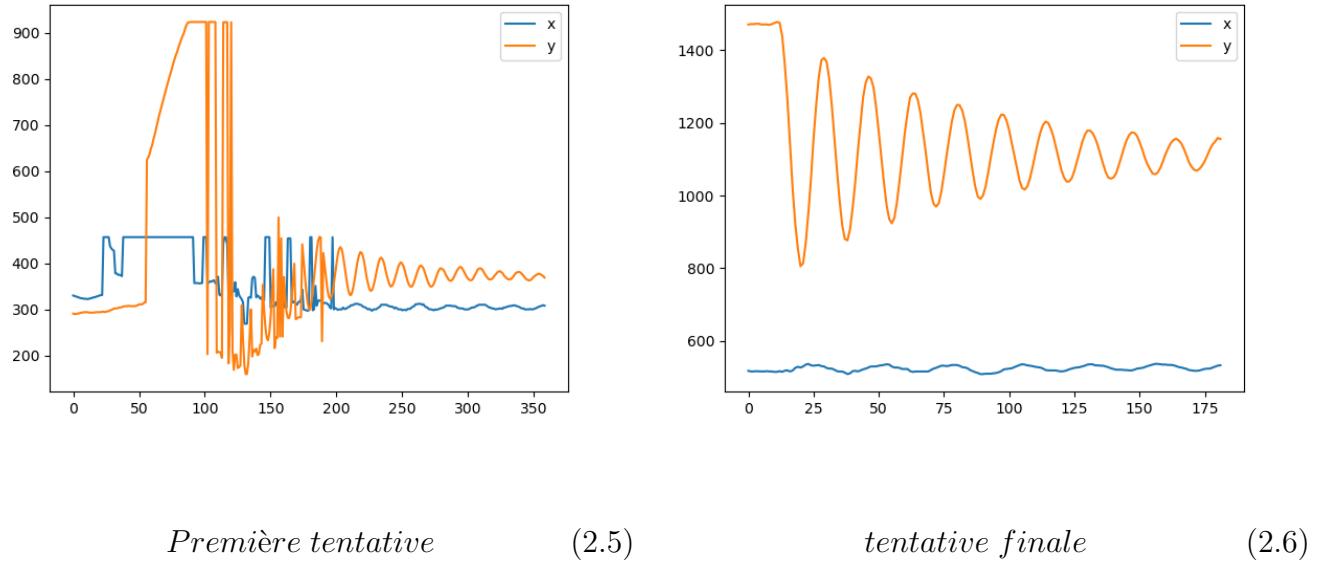


FIGURE 2.6 – Position verticale (y) et horizontale (x) (en pixels) en fonction du temps (en frame)

Ce système se différencie du précédent par la présence du ressort reliant directement la bouteille au sol. Nous avons effectué le tiré-lâché avec et sans le ressort et comparé les graphes obtenus. Avec le ressort, les oscillations sont plus rapides et moins amples. Sur une même durée, le système avec ressort effectue un aller-retour en plus que celui se caractérisant seulement par l'élastique. Ces différences sont visibles sur les graphes résultant des trackings. Nous en avons finalement conclu que le ressort inférieur permet de réduire la force résultante agissant sur la masse. La constante de rappel de l'élastique étant trop élevée, nous ne pouvons observer assez d'oscillations sans le ressort.

2.1.4 Système 4 : amortissement externe 2

Ce dernier système masse-ressort possède un amortissement externe comme le premier. Nous avons cette fois-ci directement plongé la masse dans une bassine d'eau.

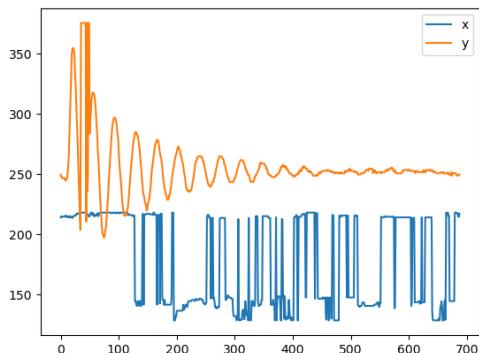
Le système est constitué d'une planche en bois récupérée à laquelle nous avons vissé un crochet. Nous avons fixé la planche au bout d'une table avec deux pinces et avons ainsi obtenu un système très stable.



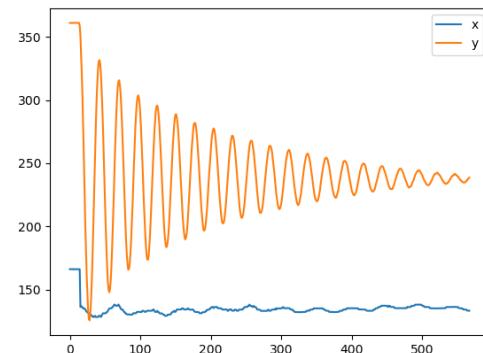
FIGURE 2.7 – Photo du système 4

Il n'a pas été facile de trouver une masse qui nous a permis d'avoir un nombre d'oscillations satisfaisant et une fréquence entre 1 et 3 Hz comme demandé. Nous avons fini par trouver une masse de petite surface et suffisamment lourde qui convenait très bien à nos besoins et l'avons suspendu au ressort.

Nous avons alors réalisé l'expérience plusieurs fois. Le point rouge que le logiciel de tracking repère, initialement submergé a été fixé hors de l'eau grâce à une tige pour augmenter sa visibilité. Les améliorations effectuées sur le système sont visibles sur les graphiques obtenus :



Première tentative (2.7)



tentative finale (2.8)

FIGURE 2.8 – Position verticale (y) et horizontale (x) (en pixels) en fonction du temps (en frame)

2.2 Paramètres des systèmes

Voici les valeurs des paramètres trouvées pour les quatre systèmes grâce au protocole expliqué dans la section 1.2 :

Paramètres	système 1	système 2	système 3	système 4
masse	$m = (0.165 \pm 0, 001) \text{ kg}$	$m = (0, 282 \pm 0, 005) \text{ kg}$	$m = (0, 100 \pm 0, 001) \text{ kg}$	$m = (0.06 \pm 0.001) \text{ kg}$
amplitude	$A = 358 \text{ pixels}$	$A = 335 \text{ pixels}$	$A = 325 \text{ pixels}$	$A = 117 \text{ pixels}$
constante de raideur	$k = (4, 25 \pm 0, 05) \text{ N/m}$	$k = (30.74 \pm 1) \text{ N/m}$	$k = (8.92 \pm 0.05) \text{ N/m}$	$k = (3, 92 \pm 0, 09) \text{ N/m}$
période amortie	$T_d = 1.25s$	$T_d = 0.60s$	$T_d = 0.53s$	$T_d = 0.9s$
pulsation propre	$\omega_0 = 5, 07 \text{ rad/s}$	$\omega_0 = 10.45 \text{ rad/s}$	$\omega_0 = 9.44 \text{ rad/s}$	$\omega_0 = 8.08 \text{ rad/s}$
pulsation amortie	$\omega_d = 5,03 \text{ rad/s}$	$\omega_d = 10.47 \text{ rad/s}$	$\omega_d = 11.85 \text{ rad/S}$	$\omega_d = 6.98 \text{ rad/s}$
coefficient d'amortissement	$\alpha = 0.236 \text{ s}^{-1}$	$\alpha = 0.31 \text{ s}^{-1}$	$\alpha = 0.54 \text{ s}^{-1}$	$\alpha = 0.19 \text{ s}^{-1}$
déphasage	$\phi = 0 \text{ rad}$	$\phi = 0 \text{ rad}$	$\phi = 10.3 \text{ rad}$	$\phi = 3.22 \text{ rad}$

Pour vérifier ces paramètres, nous avons utilisé un module d'excel comme décrit plus haut.

Voici les valeurs obtenues :

Paramètres	système 1	système 2	système 3	système 4
amplitude	$A = 358, 66 \text{ pixels}$	$A = 264 \text{ pixels}$	$A = 364 \text{ pixels}$	$A = 121.64 \text{ pixels}$
pulsation amortie	$\omega_d = 5.01 \text{ rad/s}$	$\omega_d = 9.71 \text{ rad/s}$	$\omega_d = 11.7 \text{ rad/s}$	$\omega_d = 6.99 \text{ rad/s}$
coefficient d'amortissement	$\alpha = 0.23 \text{ s}^{-1}$	$\alpha = 0.39 \text{ s}^{-1}$	$\alpha = 0.52 \text{ s}^{-1}$	$\alpha = 0.18 \text{ s}^{-1}$
déphasage	$\phi = 0 \text{ rad}$	$\phi = 2.61 \text{ rad}$	$\phi = 10.8 \text{ rad}$	$\phi = 2.73 \text{ rad}$

Chapitre 3

Vérification du modèle

Le modèle mathématique maintenant établi, il reste encore à le vérifier.

Vérification du coefficient d'amortissement α

Le paramètre α peut être vérifié grâce à la formule $\omega_d^2 = \omega_0^2 - \alpha^2$. En remplaçant les pulsations respectives par les valeurs trouvées expérimentalement, nous n'obtenons pas toujours le même α que celui trouvé via le protocole. Ces erreurs sont probablement dues aux incertitudes sur les constantes de rigidité k des différents ressorts qui ont été calculées expérimentalement.

Pour calculer la pulsation sans amortissement, refaire un tracking vidéo sans amortissement aurait été la meilleure solution, de telle sorte à pouvoir mesurer la période d'une manière nettement plus précise. Seulement, cette méthode n'aurait fonctionné qu'avec les systèmes à amortissement essentiellement externe. On ne peut en effet pas supprimer l'amortissement s'il provient uniquement du ressort lui-même (comme c'est le cas pour les systèmes 2 et 3).

Élévation monotone

Pour chaque système masse-ressort, une élévation monotone de la masse d'environ 10 centimètres a été réalisée pour vérifier le modèle mathématique établi.

Pour cela, il a fallu déterminer expérimentalement le mouvement qu'il faut conférer au haut du ressort pour que la masse s'élève de 10 cm sans osciller. Après l'avoir déterminé, l'expérience a été effectuée plusieurs fois pour chacun des quatre systèmes.

Un tracking vidéo a à nouveau permis d'obtenir un fichier .csv qui reprend toutes les positions de l'extrémité libre du ressort et de la masse en fonction du temps.

Il reste maintenant à vérifier le modèle. Voici les deux équations de l'oscillateur linéaire amorti obtenues dans le modèle. L'une pour un amortissement externe :

$$\ddot{x}_2 + 2\alpha \dot{x}_2 + \omega_0^2(x_2 - x_1) = 0$$

et l'autre pour un amortissement interne :

$$\ddot{x}_2 + 2\beta(\dot{x}_2 - \dot{x}_1) + \omega_0^2(x_2 - x_1) = 0$$

Un programme python a permis de calculer les dérivées temporelles premières et secondes de la position de la main et de l'extrémité libre du ressort grâce au fichier .csv renvoyé par le tracking. Lorsque ces valeurs sont injectées dans les équations respectives des différents systèmes, le résultat devrait être nul pour tout temps et peut être représenté sur un graphique.

Ci-dessous se trouvent, respectivement, les graphes des 4 systèmes :

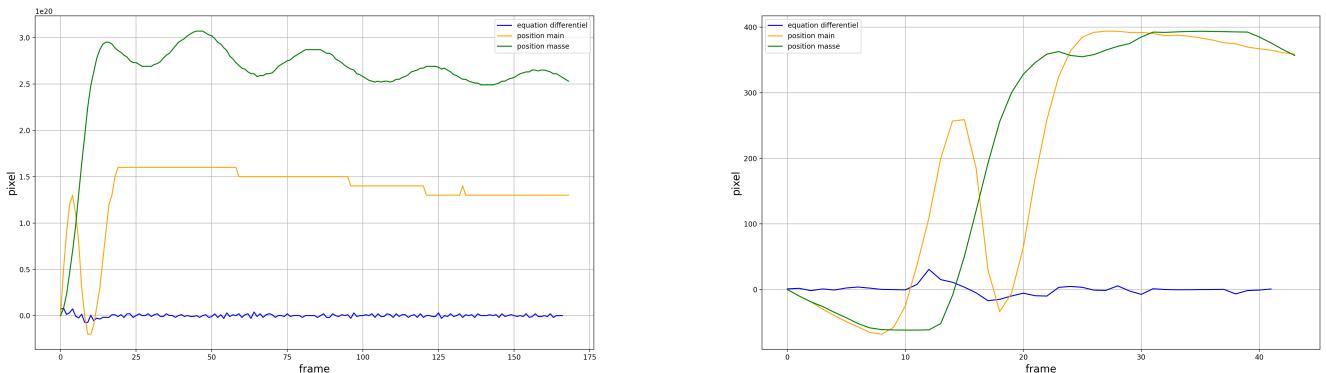


FIGURE 3.1 – Graphes de l'élévation monotone des systèmes 1 et 2 et du résultat de l'ED en tout temps (en bleu)

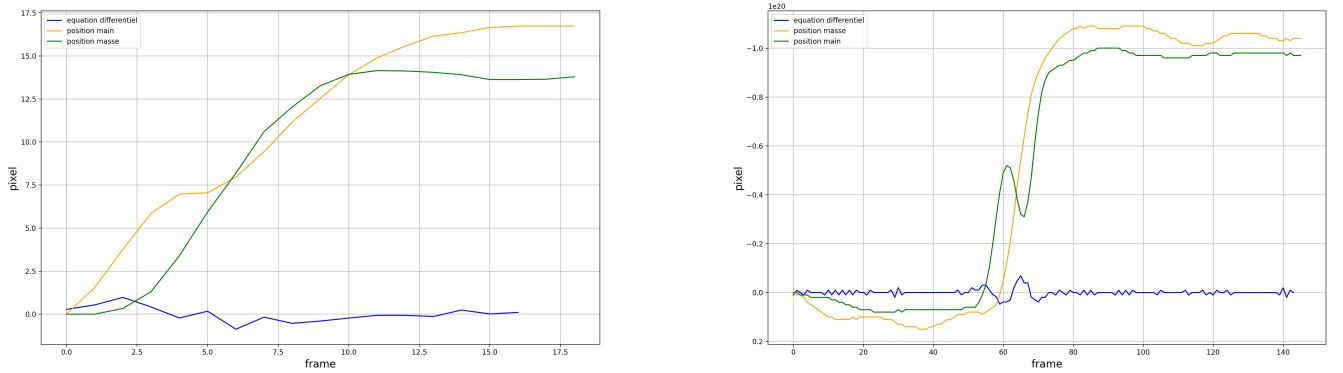


FIGURE 3.2 – Graphes de l’élévation monotone des systèmes 3 et 4 et du résultat de l’ED en en tout temps (en bleu)

Pour le système 3, la tâche a été plus compliquée. Malgré un grand nombre d’essais, le tracking par le logiciel fourni ne se faisait pas correctement. Nous avons finalement trouvé un logiciel appelé "Tracker" [18] qui nous a permis de faire un tracking manuel moins précis mais tout de même efficace.

On peut voir sur ces graphes que les équations établies sont vérifiées. Les graphes présentent une incertitude mais celle-ci peut être négligée par rapport à l’amplitude du mouvement. Différentes raisons expliquent que la courbe bleue ne soit pas exactement nulle à chaque instant :

- Les imprécisions dues à l’élévation elle-même. D’une part, on a considéré dans notre modèle un cas idéal où la position en x est constante. Si l’on modifie cette hypothèse, il se peut que quelques incohérences apparaissent. Cependant, ce mouvement en x étant assez léger, il ne peut donc pas fondamentalement influencer le résultat obtenu. D’autre part, le mouvement conféré au haut du ressort présente une imprécision inévitable car il est effectué à la main. Cette imprécision a pu être minimisée en réalisant un grand nombre d’essais.
- Les imprécisions des différents coefficients dans l’équation différentielle (les erreurs de mesure liées à k, à la masse m, au coefficient d’amortissement...). On a donc essayé de modifier les paramètres de quelques unités pour voir si une imprécision pouvait fondamentalement changer notre résultat. Il s’est avéré que non, la réponse reste quasiment la même. Nous en concluons donc que ces imprécisions sont des sources d’erreur qui peuvent

être négligées.

- Les imprécisions dues au tracking vidéo car celui-ci n'est pas absolu. Il se peut qu'il y aient quelques erreurs concernant la position exacte des points considérés. Notamment des micro-mouvements du point de tracking au sein même de la zone rouge à tracker.

Chapitre 4

Description du dispositif laser et démarche utilisée

Afin de manipuler le dispositif show laser pour ensuite générer les formes souhaitées, il a fallu comprendre son fonctionnement.

Tout d'abord, il est important d'analyser sa dynamique. Une analogie peut être faite avec les systèmes masse-ressort étudiés au premier quadrimestre. Les miroirs présentent une inertie due à leur masse. Ils sont reliés au boîtier des galvanomètres par une tige pouvant être assimilée au ressort de nos systèmes. Elle possède en effet une capacité de torsion. Cette analogie avec les systèmes nous permet d'appliquer les équations obtenues pour les systèmes masse-ressort aux galvanomètres. Pour projeter des figures voulues avec le dispositif, des impulsions électriques variables sont appliquées aux bobines des galvanomètres. Un champ magnétique est donc généré et des forces magnétiques entraînent la rotation des galvanomètres.

Nous avons eu la possibilité d'envoyer des fichiers de commande via un serveur Owncloud pour commander le dispositif à distance. Le format des fichiers d'entrée est du type .csv et comprend trois colonnes à remplir. Les deux premières correspondent aux signaux passés respectivement aux galvanomètres 0 (vertical) et 1 (horizontal) et la dernière indique au moyen d'un entier si le laser doit être éteint (0) ou allumé (1)¹. Chaque ligne du fichier correspond donc à un état spécifique du système dont la fréquence de lecture est de 44,1 kHz. Par analogie

1. Malheureusement, cette fonctionnalité n'a pas pu être activée.

avec nos systèmes masses-ressort, nous appellerons frame un état du signal envoyé. Notons que les valeurs du fichier doivent être comprises entre -1 et 1 et que nous nous sommes limités à 4 décimales en ce qui concerne les deux premières colonnes pour ne pas inutilement alourdir les fichiers.

Afin de réaliser nos essais, nous avons choisi d'utiliser l'interface de programmation interactive open-source Jupyter. La bibliothèque matplotlib a été utilisée afin de générer la plupart de nos graphiques via python, lesquels sont donc affichés directement dans notre notebook. Pour générer nos fichiers et effectuer certains calculs, nous utilisons également le module numpy [19]. Cette méthode a permis de déterminer les paramètres de nos galvanomètres [G].

Les signaux contenus dans les fichiers sont transmis aux deux galvanomètres grâce à une carte son. Ces derniers reproduisent les commandes et contrôlent par conséquent le tracé du laser. La trajectoire de ce dernier est trackée par une caméra, ce qui nous permet d'obtenir des fichiers de retour pour analyser les résultats obtenus. L'ensemble des données récoltées nous sont ensuite envoyées dans un répertoire de sortie. Ces dernières sont compressées dans un dossier .zip et contiennent entre autre le signal de sortie du laser, une image de la figure obtenue et une vidéo du résultat.

Chapitre 5

Recherche des paramètres des galvanomètres

5.1 Paramètres des galvanomètres

Avant de pouvoir produire des figures au laser, il était nécessaire de déterminer les valeurs des différents paramètres des deux galvanomètres. Trois méthodes détaillées en annexe¹ ont été utilisées pour les trouver. Pour vérifier que ces valeurs étaient bien les bonnes, deux autres méthodes ont été employées.

Tout d'abord, le graphique de l'amplitude d'oscillation du laser en fonction de la fréquence du signal d'impulsion a été établi. Ces graphiques serviront aussi par la suite pour trouver le type d'amortissement des galvanomètres. Pour réaliser ces courbes, nous avons envoyé des signaux sinusoïdaux à amplitude fixe sur un large spectre de fréquences aux galvanomètres et avons ensuite mesuré l'amplitude du mouvement du laser pour chaque fréquence de signal d'entrée. Ci-dessous les graphiques obtenus :

1. Voir annexe A

Galvanomètre 0 (vertical)



FIGURE 5.1 – Graphique de l'amplitude en pixels en fonction de la pulsation des oscillations

Galvanomètre 1 (horizontal)

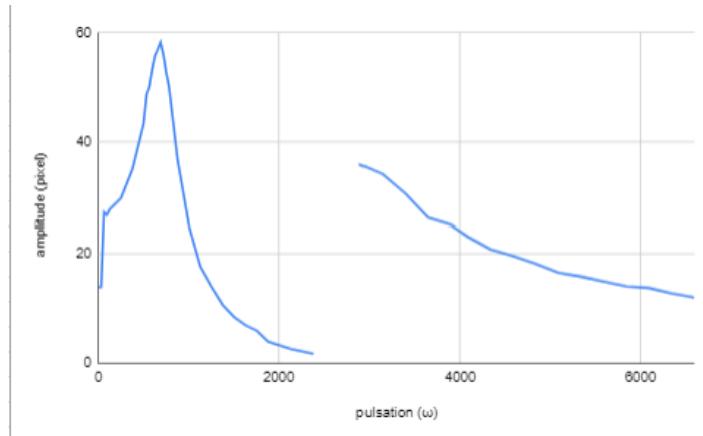


FIGURE 5.2 – Graphique de l'amplitude en pixels en fonction de la pulsation des oscillations.

La discontinuité visible sur ce deuxième graphe est due à un changement d'amplitude des sinus d'impulsion. Pour obtenir des valeurs exploitables pour des pulsations élevées (de 3500 à 6500 rad/s), l'amplitude des sinus d'entrée a été adaptée, elle est passée de 0.05 à 0.5. Ainsi, la marge d'erreur s'est vue fortement réduite. Ce saut de continuité dans la fonction ne dérange pas puisque seuls les comportements asymptotiques en l'infini et proches de ω_0 sont utiles ici et non celui à la transition entre les deux.

5.1.1 Discussion des résultats

Comme on peut le constater dans les tableaux de résultats ci-dessous, des différences notables apparaissent entre les valeurs des paramètres calculés selon les différentes méthodes. Cependant, comme nous pouvons le voir sur les graphiques ci-dessous, la méthode 3 est plus adaptée pour trouver le α . En effet, comme les graphiques 5.1.1 et 5.1.1 en témoignent, les paramètres d'amortissement déterminés par la troisième méthode correspondent presque parfaitement au mouvement décrit par le laser. C'est pourquoi nous utiliserons ces valeurs pour la suite. Voici ci-dessous les deux graphes obtenus avec en bleu les courbes provenant du tracking laser, et en orange les courbes tracées grâce à l'approximation d'excel, utilisant les α de la méthode 3 :

Galvanomètre 1 (horizontal)

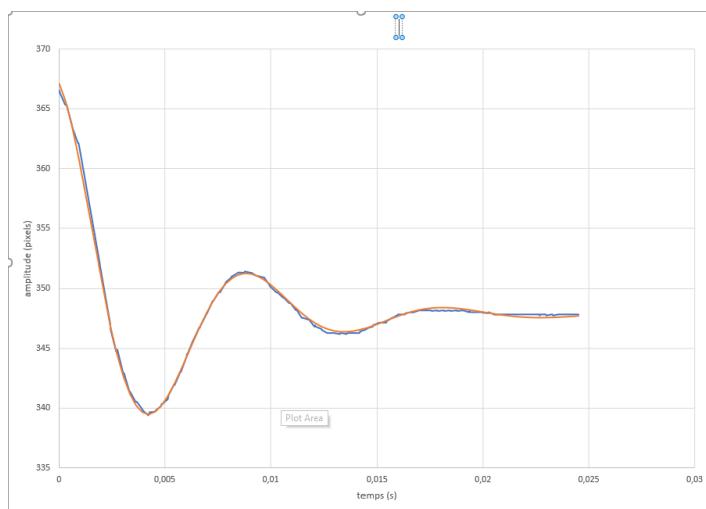


FIGURE 5.3 – Graphique de la position horizontale du laser (en pixels) en fonction du temps (en secondes)

Galvanomètre 0 (vertical)

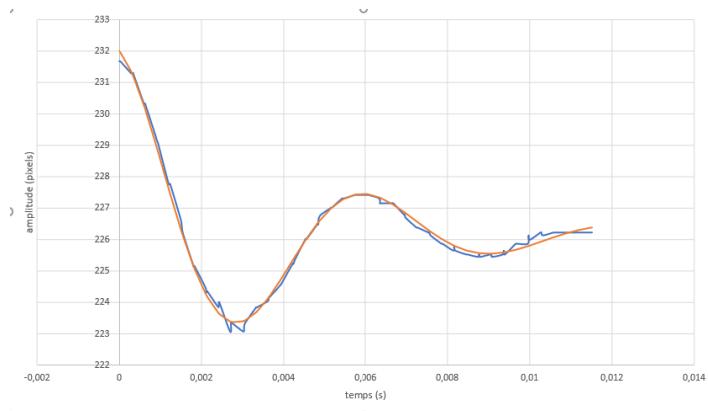


FIGURE 5.4 – Graphique de la position verticale du laser (en pixels) en fonction du temps (en secondes)

Paramètres

Ci-dessous sont reprises les valeurs des paramètres obtenus avec les différentes méthodes pour le galvanomètre vertical (0) (en partant d'une pulsation amortie ω_d valant 1068 rad/s) :

	fréquence propre (ω_0)	coefficients d'amortissement (α)
méthode 1	1068 rad/s	197,67 rad/s
méthode 1 bis	1068 rad/s	151,185 rad/s
méthode 2	1088,78 rad/s	211,72 rad/s
méthode 3	1020,01 rad/s	248,66 rad/s

Et celles du galvanomètre horizontal (1) (avec $\omega_d = 691,15 \text{ rad/s}$) :

	fréquence propre (ω_0)	coefficients d'amortissement (α)
méthode 1	691,15 rad/s	216,5 rad/s
méthode 1 bis	691,15 rad/s	185,22 rad/s
méthode 2	711,74 rad/s	169,98 rad/s
méthode 3	679,18 rad/s	190,18 rad/s

5.1.2 Vérification de l'allure des courbes de résonance

Pour vérifier que nous avons choisi les paramètres les plus cohérents, les courbes de résonance expérimentales et théoriques ont été comparées. Cette comparaison visible sur les graphiques de la courbe de résonance du galvanomètre horizontal qui figure ci-dessous permet de constater que si les courbes du galvanomètre horizontal sont quasi semblables, celles du galvanomètre vertical présentent de légères différences. Sur le graphique ci-dessous figurent en vert la courbe de résonance expérimentale, en orange la courbe théorique qui correspond à un amortissement interne et en bleu celle de l'amortissement externe. Les paramètres utilisés qui permettent d'obtenir un tel résultat sont ceux obtenus via la troisième méthode. Remarquons tout de même que la courbe expérimentale semble correspondre à celle d'un amortissement interne proche de la pulsation de résonance, et à un amortissement externe pour des plus grandes valeurs de ω . Nous trouverons par la suite que le galvanomètre horizontal est à amortissement interne.

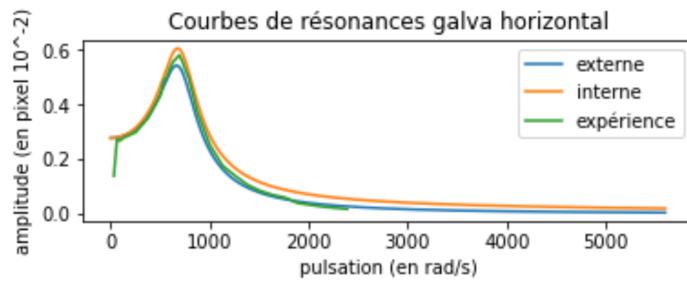


FIGURE 5.5 – Courbes de résonance du galvanomètre horizontal

Pour le galvanomètre vertical, avec les paramètres déterminés via la méthode 3, voici le résultat :

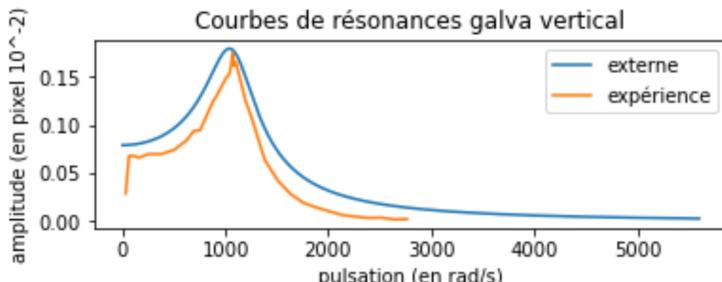


FIGURE 5.6 – Courbes de résonance du galvanomètre vertical

Lorsque nous utilisons le coefficient d'amortissement obtenu via la méthode numéro 2, les

deux courbes sont plus proches :

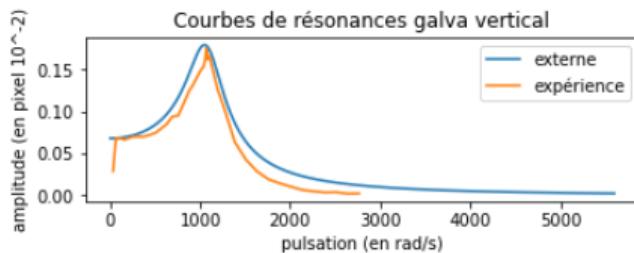


FIGURE 5.7 – Courbes de résonance du galvanomètre vertical

Comme cette légère différence de valeur de α ne semble pas avoir d'impact considérable sur la qualité des formes réalisées à l'aide du laser, nous conserverons pour la suite la valeur de 248,66 Hz pour α . L'imprécision de la détermination de α par la méthode 3 est très certainement due à un manque d'ampleur des oscillations post-impulsions.

5.2 Détermination de l'amortissement des galvanomètres

5.2.1 Comportement des systèmes à amortissement interne et externe

Nous allons maintenant déterminer expérimentalement le type d'amortissement des deux galvanomètres. Pour cela, nous allons étudier les différences de comportement entre un système à amortissement interne et un système à amortissement externe.

Dans le modèle mathématique, nous avons établi un lien entre la position de l'extrémité libre du ressort x_1 et la masse attachée au ressort x_2 grâce aux phaseurs. Dans l'analogie avec les galvanomètres, l'extrémité libre représente la base du galvanomètre et la masse représente le miroir. Nous pouvons utiliser ces équations de phaseurs pour les galvanomètres.

Étudions les comportements asymptotiques des systèmes à amortissement interne et externe, c'est à dire pour des valeurs de ω qui tendent vers l'infini.

1) amortissement externe :

$$\begin{aligned} |\underline{X}_E| &= \frac{|\underline{X}_s| \omega_0^2}{\omega_0^2 + 2\alpha i\omega - \omega^2} \\ \lim_{\omega \rightarrow \infty} |\underline{X}_E| &= -\frac{\omega_0^2}{\omega^2} |\underline{X}_s| \end{aligned} \quad (5.1)$$

2) amortissement interne :

$$\begin{aligned} |\underline{X}_I| &= \frac{\omega_0^2 - 2\beta i\omega}{\omega_0^2 + 2\beta i\omega - \omega^2} |\underline{X}_s| \\ \lim_{\omega \rightarrow \infty} |\underline{X}_I| &= -\frac{2\beta i}{\omega} |\underline{X}_s| \end{aligned} \quad (5.2)$$

Nous pouvons conclure que ces deux systèmes sont différentiables grâce à leurs comportements asymptotiques en l'infini. Quand un galvanomètre à amortissement interne se comportera à très hautes pulsations en $1/\omega$, un galvanomètre à amortissement externe se comportera en $1/\omega^2$.

5.2.2 Application aux galvanomètres du projet

Pour faire la différence entre les deux types d'amortissement, nous avons envoyé des signaux harmoniques à très hautes fréquences aux deux galvanomètres et avons mesuré l'amplitude des signaux reçus. Pour mieux pouvoir différencier les deux amortissements, il est utile de linéariser les deux expressions (5.1) et (5.2) en portant $\log(|\underline{X}_2|)$ en fonction de $\log(\omega)$ sur un graphe.

Si le galvanomètre est à amortissement externe, la pente de la droite sur le graphe logarithmique sera de -2 , alors que si le galvanomètre est à amortissement interne, elle sera de -1 .

1) Galvanomètre vertical

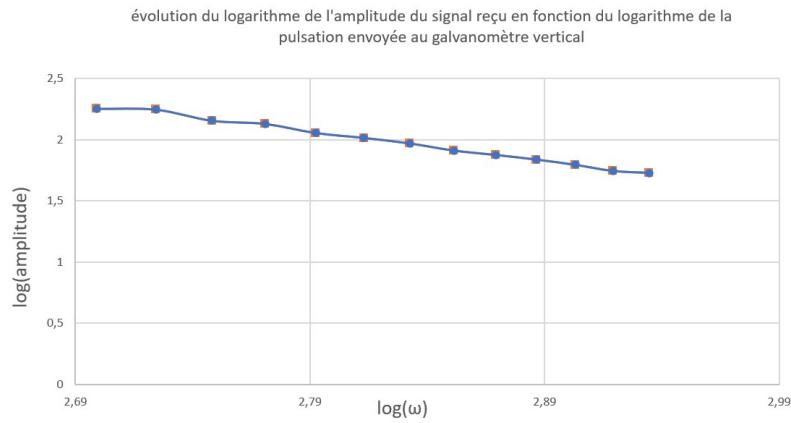


FIGURE 5.8 – graphe de $\log(|X_2|)$ en fonction de $\log(\omega)$

Pour la pente p de la droite qui approxime le mieux la courbe, on obtient : $p = -2,17$. Pour trouver une telle pente, nous avons utilisé les amplitudes correspondant à des fréquences allant de 500 à 1000 Hz.

Le galvanomètre vertical possède donc un amortissement externe. Ce résultat est confirmé par la figure 5.7 où l'on voit que les comportements de la courbe théorique d'un amortissement externe et la courbe expérimentale sont assez proches.

2) Galvanomètre horizontal

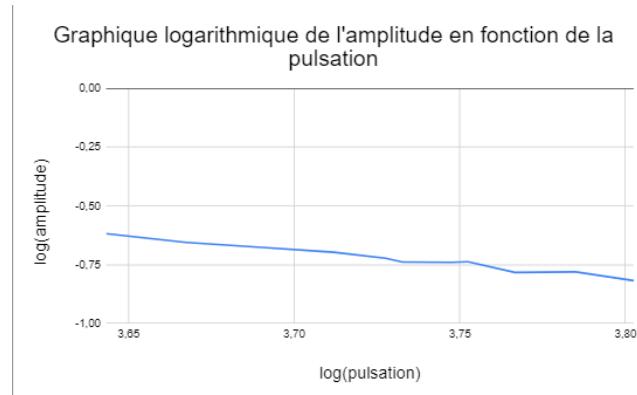


FIGURE 5.9 – graphe de $\log(|X_2|)$ en fonction de $\log(\omega)$

Nous avons pris pour le galvanomètre horizontal les mesures des amplitudes correspondant à des fréquences allant de 710 à 1000Hz et avons ainsi obtenu une pente de $-1,25$. Cette valeur ne permet pas de trancher sur le type d'amortissement du galvanomètre, mais elle est plus proche d'un amortissement interne. Nous réaliserons chaque figure en considérant les deux types d'amortissement. Cette démarche permettra de déterminer que le galvanomètre horizontal est à amortissement interne (même si la figure 5.1.2 pourrait laisser penser le contraire).

Chapitre 6

Résultats finaux des figures à tracer

Après avoir calculé les différents paramètres des galvanomètres et déterminer leur amortissement, montrons les résultats des différentes figures non complexes à projeter. Ces figures possèdent des équations paramétrique en coordonnées cartésiennes en fonction du temps t , qui sont détaillées dans l'annexe B. Nous les avons utilisées dans le modèle comme expliqué au chapitre 7. Certaines d'entre elles sont triviales et d'autres nécessitent un développement mathématique. Les codes des différentes formes réalisées se trouvent également en annexe. Voici les résultats obtenus pour diverses figures :

6.1 Cercle

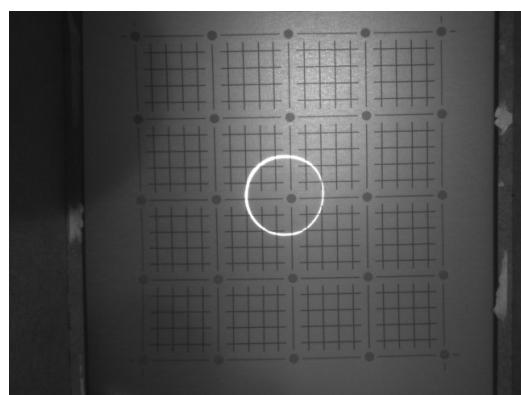


FIGURE 6.1 – Cercle

6.2 Ellipse

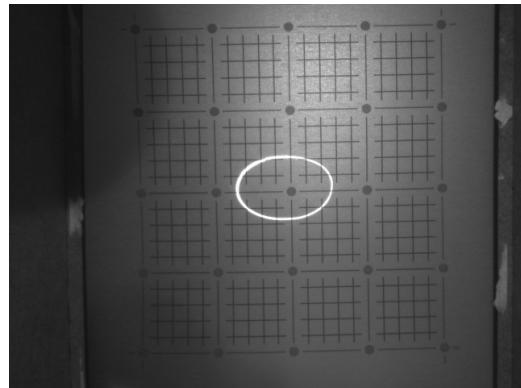


FIGURE 6.2 – Ellipse

6.3 Hypotrochoïde

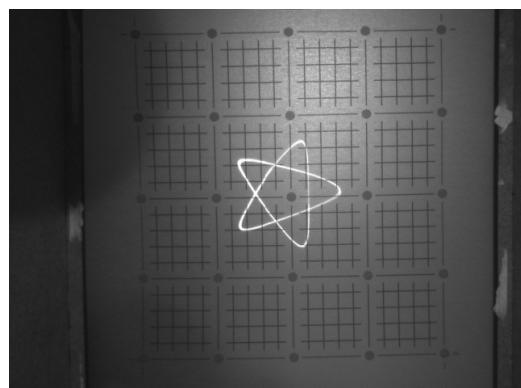


FIGURE 6.3 – Hypotrochoïde étoile

6.4 Hypotrochoïde nuage

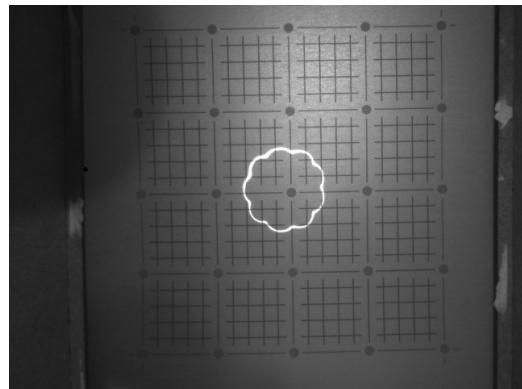


FIGURE 6.4 – Hypotrochoïde nuage

6.5 Figure dynamique

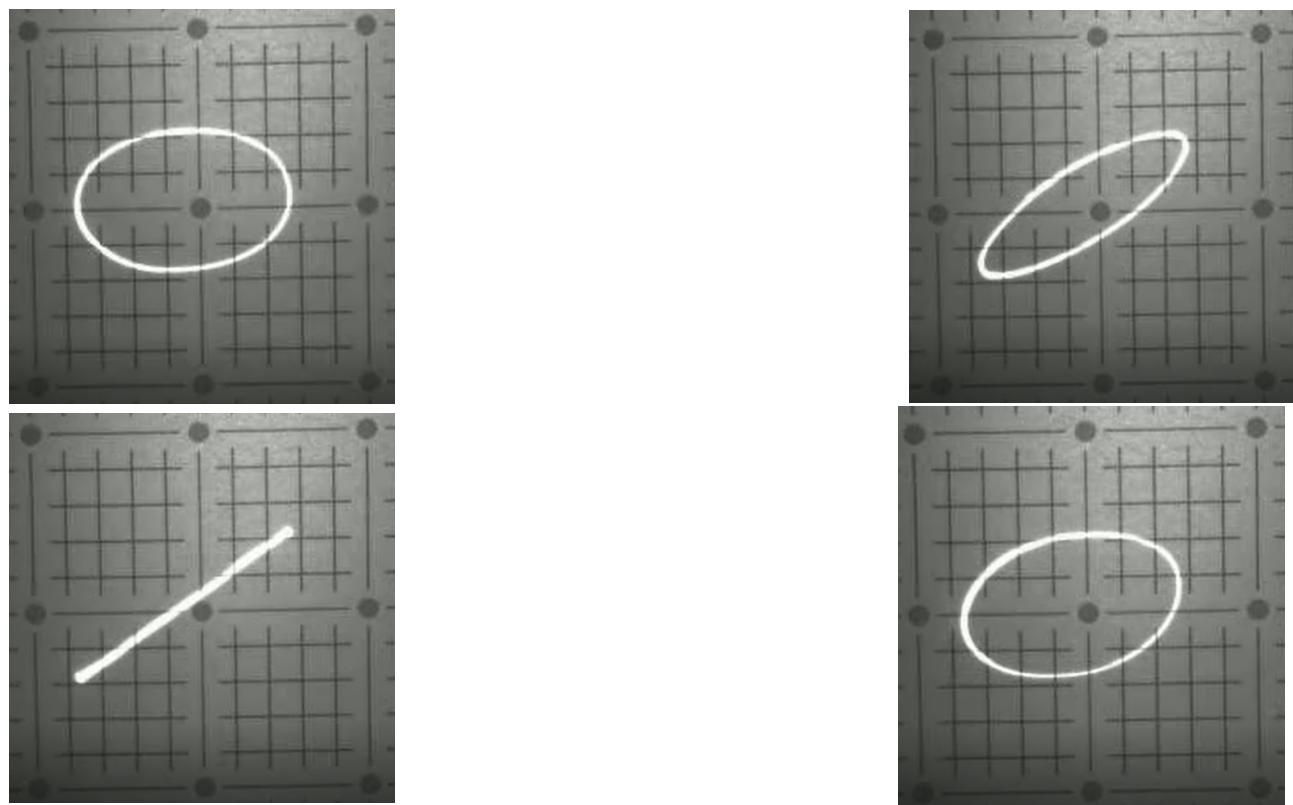


FIGURE 6.5 – Etat de l'ellipse à différents instants de l'animation.

6.6 Lemniscate de Bernoulli

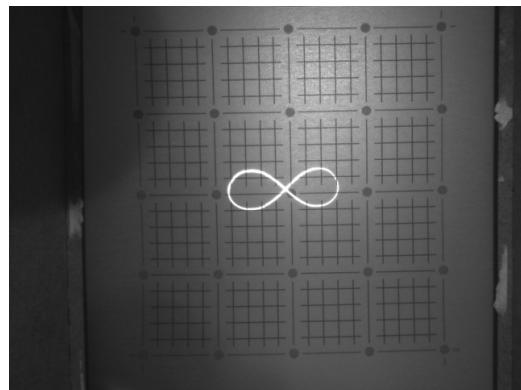


FIGURE 6.6 – Lemniscate

6.7 Carré

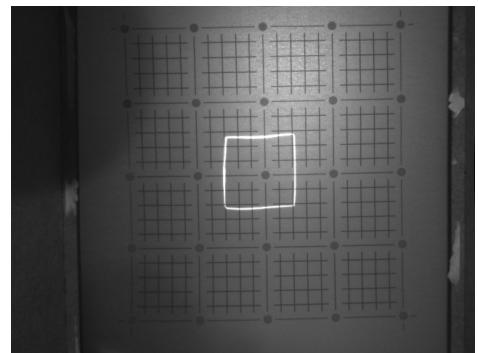
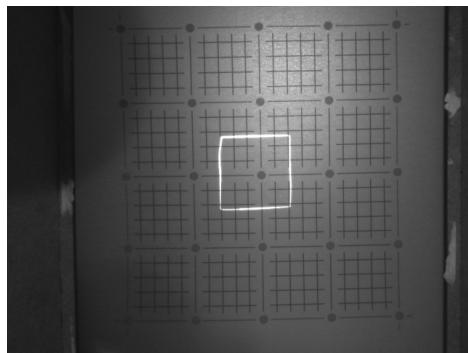


FIGURE 6.7 – De gauche à droite : carré avec un taux de rafraîchissement de 30 et 60 Hz

Chapitre 7

Programmation

7.1 Fonctions de transfert

Le galvanomètre étant un système oscillant, les signaux de sortie ne sont pas directement proportionnels aux signaux d'entrée envoyés. L'étude des systèmes masses-ressort nous a toutefois permis de décrire les relations entre ces deux mouvements. C'est sur base de ces connaissances que nous pourrons générer les formes souhaitées. Le logiciel final est décrit au travers de ce dernier chapitre que nous commençons au travers des fonctions de Transfert.

Les deux modèles impliquent l'utilisation des dérivées premières et secondes des signaux de sortie que nous supposons connus, nous avons donc défini une fonction permettant de dériver nos listes de points. Les signaux étant discrets par définition, nous calculons pour chaque couple de points les pentes qu'ils définissent entre eux afin de déterminer le signal dérivé. A chaque opération de dérivation, la longueur de la liste étant diminuée d'un point, il nous faudra anticiper cette perte dans la création des signaux de sortie.

Pour déterminer les signaux d'entrée dans le cas d'un amortissement externe, nous définissons une fonction qui calculera chacun des points du signal sur base de l'équation que nous avons obtenue et en fonction des données connues pour la sortie désirée et les paramètres intrinsèques du système.

Dans le cas d'un amortissement interne, nous avons dès lors affaire à une équation différentielle linéaire du premier ordre. Sur base de nos recherches, nous avons décidé d'utiliser la méthode d'Euler [4] pour sa résolution numérique, ce qui nous permet de calculer le signal d'entrée point par point de la même manière que pour le cas externe. Bien que des méthodes plus précises existent, les résultats obtenus avec cette dernière sont globalement satisfaisants.

Les paramètres déterminés au chapitre 5 sont exprimés selon les unités du SI. Toutefois, comme nous l'avons déjà mentionné ci-dessus, chaque ligne du signal envoyé correspond à une frame. Pour éviter toute confusion dans la suite et pour plus de facilité dans nos calculs numériques, nous avons décidé de travailler dans ces unités-là. La fréquence du laser étant de 44,1 kHz, nous devons diviser les pulsations propres et coefficients d'amortissement α de nos galvanomètres par cette valeur pour rester cohérent en terme dimensionnel par la suite. Ce choix a été fait suite à de nombreuses erreurs quant à la réalisation du logiciel final.

7.2 Fonctions de calcul pour les équations paramétriques

Une fois les fonctions permettant le transfert entre le signal de sortie et d'entrée établies, il nous a fallu écrire le programme permettant de générer les signaux souhaités. Nous avons commencé par établir le cas simple des signaux défini par les équations paramétriques établies au chapitre précédent.

Pour chaque figure, nous avons calculé la liste de points en traduisant directement les équations correspondantes grâce aux fonctions trigonométriques que nous fournit le module numpy. Pour le carré, les équations utilisées dépendant de la valeur de la variable frame, nous avons procédé à un calcul par morceau et point par point au moyen d'une boucle de calcul. Afin de pouvoir contrôler le nombre de périodes sur un même signal, nous avons inséré notre boucle dans une autre dont l'itération est basée sur le nombre de périodes à calculer. Notons que la dernière période du signal n'est pas forcément complète. Le calibrage dépendra de la durée du signal et de sa fréquence. L'ensemble de ces fonctions se trouve dans la partie Geometry du logiciel.

7.3 Généralisation à toute forme complexe

En parallèle de l'élaboration des figures paramétriques, nous avons envisagé la possibilité de générer des signaux pour des figures plus complexes. Au début de nos recherches, nous n'avions pas considéré le fait que nous traitions des signaux discrets et nous avons donc cherché une méthode pour obtenir une approximation de classe C^2 de n'importe quel signal souhaité. Nous avons ainsi découvert les transformées de Fourier qui nous permettaient de réaliser cette opération. Une de ses applications nous permet en effet de déterminer les équations paramétriques de n'importe quelle figure sous forme d'une combinaison linéaire de fonctions sinus et cosinus et d'un terme indépendant. Même si l'obtention des signaux sous cette forme n'a finalement pas été utilisée par la suite, nos recherches approfondies sur le sujet nous ont permis de comprendre comment s'y prendre pour échantillonner les points de n'importe quelle figure.

Grâce à nos connaissances préalables en terme d'image vectorielle [Inkscape *Dessiner en toute liberté*, 12 et Scalable Vector Graphics *Chapter 9 : Paths*, 14], nous avons décidé d'utiliser le logiciel *Inkscape* pour générer sur base d'images matricielles nos objets vectoriels définis à l'aide de noeuds. Ces noeuds se retrouvent encodés dans un fichier SVG que nous devons traiter au moyen d'un programme afin de récolter leurs coordonnées dans un repère cartésien. L'ensemble des fonctions utilisées pour y parvenir sont regroupées dans la partie *Read_XML* du programme final.

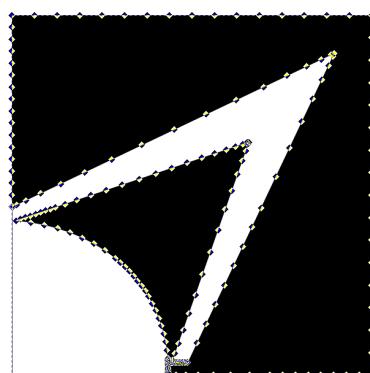


FIGURE 7.1 – Le logo de l'EPB et ses noeuds grâce au logiciel *Inkscape*

Pour construire ce code, il nous faut tout d'abord une fonction *find_path()* qui trouve et

récupère le chemin de nœuds inscrit dans le fichier XML et assigné à l'attribut d. Par le biais de nos recherches et observations [12, 14], nous avons pu déterminer comment les informations de chaque nœud y sont indiquées. En effet, chacun d'entre eux contient deux coordonnées et une commande implicite¹ symbolisée par une lettre définissant la manière dont le chemin y est parcouru. Nous pouvons dresser un tableau des commandes que nous utilisons dans les fonctions *clear()* et *coord()* travaillant conjointement dans le but de renvoyer les coordonnées cartésiennes de la figure.

Commandes		Description
Moveto	M	Se déplace vers les coordonnées absolues (x,y)
	m	Effectue une translation d'un vecteur (x,y) par rapport au point actuel
Closepath	Z ou z	Ferme un chemin de noeuds et retourne au point de départ du dernier chemin
Lineto	L	Se déplace en ligne droit vers les coordonnées absolues (x, y)
	l	Effectue une translation d'un vecteur (x, y) par rapport au point actuel
Horizontal lineto	H	Se déplace horizontalement vers la coordonnée absolue x
	h	Effectue une translation horizontale d'un vecteur (x, 0) par rapport au point actuel
Vertical lineto	V	Se déplace verticalement vers la coordonnée absolue y
	v	Effectue une translation verticale d'un vecteur (0, y) par rapport au point actuel

Enfin, il nous a fallu traiter les données récoltées dans la dernière fonction *center()* qui comme son nom l'indique, centre la figure en l'origine.

7.4 Observations des résultats et corrections

Lors de la réalisation des différents programmes, nous avons été confrontés à de nombreux problèmes que nous allons décrire par la suite ainsi que leur résolution.

1. Tant qu'une nouvelle commande n'est pas mentionnée, chaque noeud bénéficie de la propriété actuelle.

Tout d'abord, nous avons remarqué que les coefficients de proportionnalité entre les signaux de sortie et d'entrée ne sont pas les mêmes pour les deux galvanomètres. Nous voyons sur l'image ci-dessous que le cercle envoyé est étiré à la verticale, ce qui nous fait penser que le gain statique est différent selon l'axe.

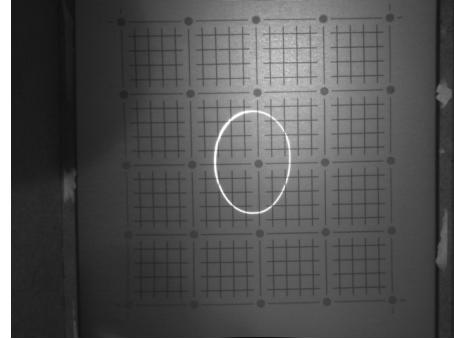


FIGURE 7.2 –

Pour calculer les coefficients de déformabilité a et b des deux galvanomètres, nous nous référerons à la relation établie ci-dessous entre l'amplitude du signal d'entrée envoyé (X_{in}) et de celle de la sortie obtenue après tracking (X_{out}) :

$$X_{out} = aX_{in}$$

Calculons ces paramètres en prenant la moyenne des résultats obtenus ci-dessous dans le cas du cercle :

Amplitude du signal d'entrée	Amplitude du signal de sortie		Coefficient	
	Galva 0	Galva 1	a	b
0.03	16.31	21.06	550.971	712.713
0.039	21.19	27.82	536.476	705.303
0.059	31.78	42.24	538.738	714.079
0.079	42.48	56.164	537.721	712.283
0.098	52.63	71.67	537.640	721.160
Moyenne		a = 541	b = 713	

En appliquant ces résultats, nous obtenons l'amplitude des signaux à envoyer tel que les proportions entre les deux galvanomètres soient respectées.

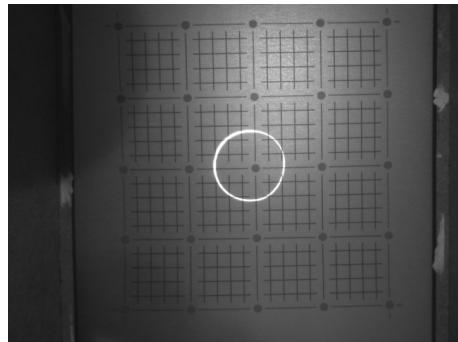
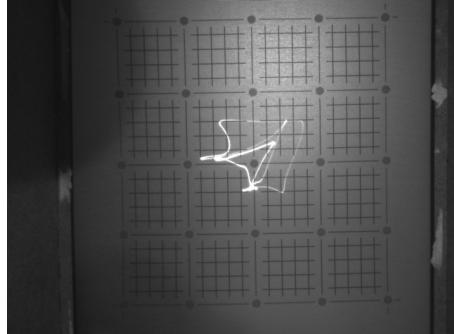


FIGURE 7.3 –

Ensuite, pour la réalisation des formes complexes, nous nous sommes basé.e.s sur quelques exemples dont le carré², le cœur et le logo de l'*EPB*. Nous avons tout d'abord remarqué que notre programme fonctionnait uniquement pour les formes dont le trait est connexe et refermé sur lui-même. En effet, pour passer d'un morceau à un autre, le chemin déterminé par les nœuds n'est pas toujours le plus optimal. Pour remédier à ce problème, nous avons modifié les images à la main pour qu'elles soient définies en une seule partie. Nous pouvons remarquer une petite différence en résolvant ce problème.



En analysant la figure ci-dessus, nous remarquons que certaines zones sont plus saturées que d'autres. La raison est due à une concentration de points en ces parties particulières. En effet, lorsqu'on génère des nœuds sur notre logiciel de dessin vectoriel, on remarque rapidement qu'ils ne sont pas répartis à égale distance. Le laser sera donc ralenti en ces amas et accéléré ailleurs, ce qui pouvait causer l'origine de nos soucis. Nous avons donc décidé de rajouter une

2. Nous pourrions en effet nous contenter d'utiliser cette méthode pour générer les autres formes paramétriques.

partie *Optimisation* à notre programme final. Ce code regroupe les fonctions nécessaires pour répartir équitablement les points d'une figure sans en modifier l'apparence. L'algorithme est pensé pour fonctionner en deux étapes.

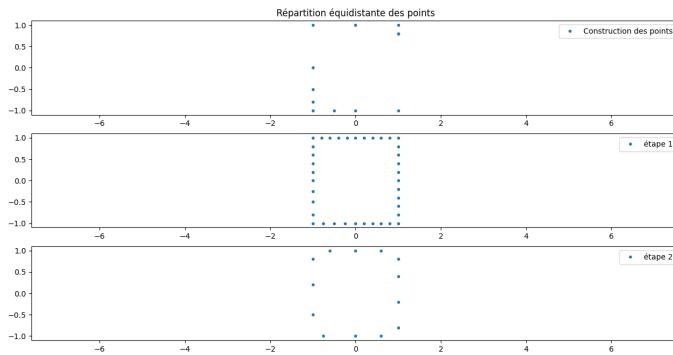


FIGURE 7.4 –

1. Rajout des points répartis à égale distance
2. Réduction du nombre de points

Malgré ces modifications, nous pouvions toujours observer que le tracé de nos figures complexes était plus ou moins fortement vallonné en certains endroits comme nous le montre les figures ci-dessous.

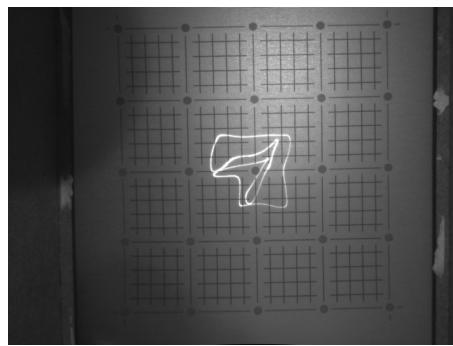


FIGURE 7.5 –

Nous avons constaté que ce problème était lié au fait que le début du graphe de nos signaux n'était presque jamais dans le prolongement de leur fin, ce qui signifie qu'entre deux périodes,

le signal change brusquement de direction. Or, il faut pouvoir en donner l'information aux galvanomètres. Si nous assemblons conjointement chaque période bout à bout par une expérience de la pensée et que nous considérons le graphe continu de la fonction, nous remarquons rapidement que tous les points pour lesquels cette dernière n'est pas dérivable doivent être contenus à l'intérieur du signal envoyé. Ainsi, en rajoutant une fonction de déphasage du signal dans la partie *Optimisation* du logiciel, et en posant un décalage temporel³ arbitraire à ce dernier, nous obtenons finalement un résultat convaincant comme le montre l'image ci-dessous.

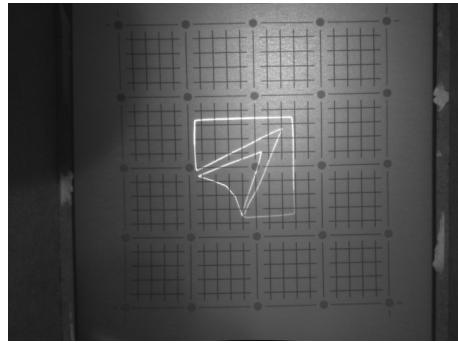
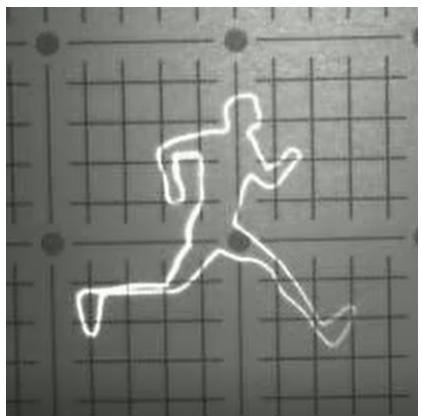
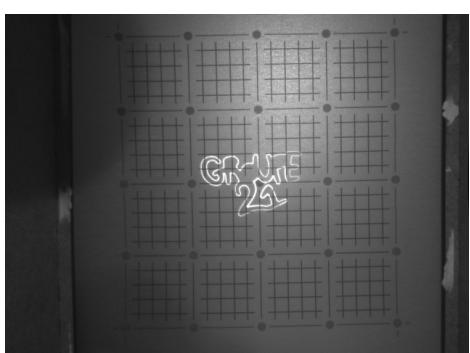
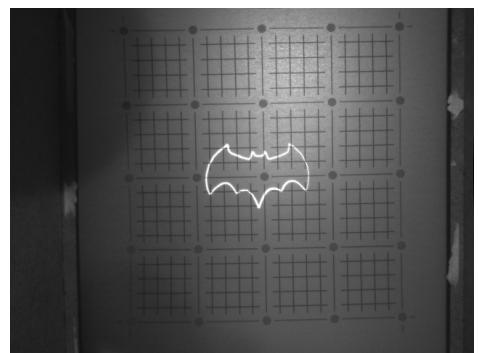
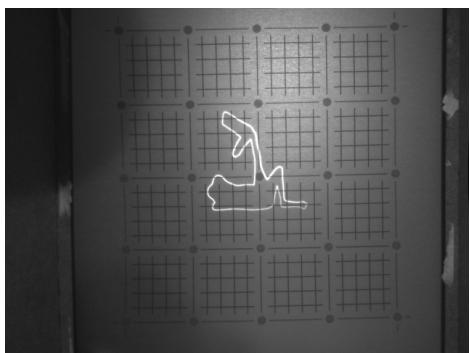
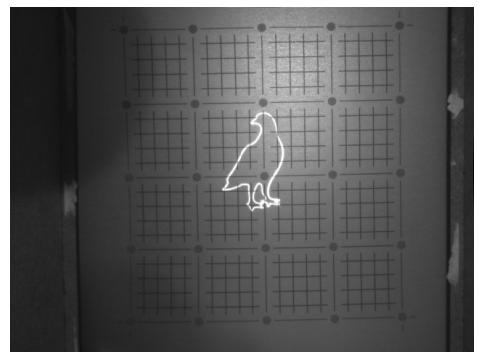
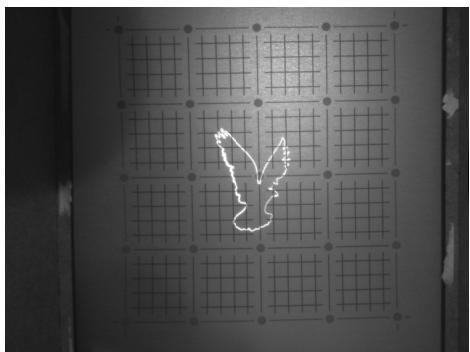
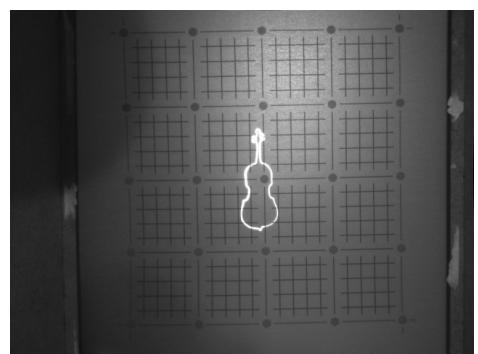
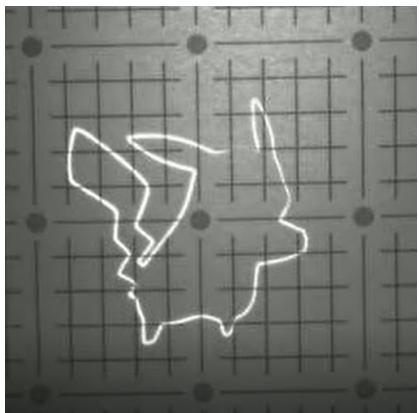


FIGURE 7.6 –

7.5 Logiciel final

Les quatre parties du programme final ayant été écrites, nous pouvons les rassembler au moyen d'un dernier fichier *Main* qui générera les fichiers à envoyer selon les demandes de l'utilisateur. Une interface a également été écrite pour aider à la bonne utilisation du programme. Nous présentons ci-dessous une partie de l'étendue des images complexes que nous pouvons afficher. Il ne reste plus qu'à faire preuve d'imagination.

3. Le terme temporel est un abus de langage puisque nous travaillons ici en frame. Ce décalage équivaut à 1/30 du nombre de frames total du signal.



Analyse de l'équipe - SWOT

SWOT du premier quadrimestre :

Forces	<ul style="list-style-type: none">— Certains membres du groupe se débrouillent très bien avec l'outil Latex— Membres du groupe disponibles et ponctuels— Bonne entente et entraide au sein du groupe malgré le fait de devoir faire les réunions en ligne— Groupe discipliné, débrouillard et avec une bonne réactivité
Faiblesses	<ul style="list-style-type: none">— Différents niveaux de compréhension— Mauvaise gestion du temps et du planning— La répartition des tâches n'est pas toujours équilibrée— Le travail est parfois fait en dernière minute— Administration négligée

Menaces	<ul style="list-style-type: none"> — Situation sanitaire actuelle, isolement général — Certains points théoriques indispensables pour le projet n'ont pas encore été vus en cours — Manque de motivation — Niveau théorique du projet
Opportunités	<ul style="list-style-type: none"> — Aide de personnes extérieures au projet (connaissances faisant des études de physique, entourage ayant fait des études d'ingénieur) — La crise sanitaire nous permet de gagner du temps que l'on peut investir dans le projet — Le cours d'informatique aide à la prise en main de Latex

SWOT du deuxième quadrimestre :

Forces	<ul style="list-style-type: none"> — Certains membres du groupe se débrouillent très bien avec l'outil Latex, d'autres ont des connaissances en dessin vectoriel — Tous les membres du groupe sont disponibles et ponctuels — Bonne entente et entraide au sein du groupe malgré le fait de devoir faire les réunions en ligne — Groupe discipliné, débrouillard et avec une bonne réactivité — Personne n'a abandonné polytechnique, le groupe est resté au complet — Bonne gestion du temps et du planning
--------	--

Faiblesses	<ul style="list-style-type: none"> — Différents niveaux de compréhension — La répartition des tâches n'est pas toujours équilibrée mais c'est améliorée depuis le Q1
Menaces	<ul style="list-style-type: none"> — Situation sanitaire actuelle, isolement général — Certains points théoriques indispensables pour le projet n'ont pas encore été vus en cours — Manque de motivation qui ne c'est pas améliorée avec le prolongement de la crise sanitaire — Niveau théorique du projet — Une partie du travail du premier quadrimestre n'a pas été terminée dans les temps
Opportunités	<ul style="list-style-type: none"> — Aide de personnes extérieures au projet (connaissances faisant des études de physique, entourage ayant fait des études d'ingénieur) — La crise sanitaire nous permet de gagner du temps que l'on peut investir dans le projet (transports, ...) — Entraide entre les différents groupes de projet

Il y a eu une amélioration entre le premier et le deuxième quadrimestre au niveau de la gestion de l'équipe. L'administration a moins été mise de côté et d'autres connaissances de certains membres du groupe nous ont aidés pour le projet. La répartition des tâches était, quant à elle, plus équilibrée lors du deuxième quadrimestre mais ne l'était toujours pas entièrement.

Analyse des matériaux

En tant qu'ingénieur(e), il est important de s'intéresser à l'impact des matériaux qu'on utilise sur l'environnement. Sur base du tableau qui nous a été fourni par l'université, nous avons donc fait une analyse détaillée des matériaux que nous avons utilisés lors de la construction du système 4. Voici le tableau que nous avons obtenu :

Pièces	Matériaux			Quantités	Unité			
Ressort	Ressort en acier galvanisé (1 unité)	Ressort acier		2,00E-02	kg			
Masse	Bouteille en plastique (25 Cl)	Polyéthylène téréphthalate		2,80E-02	kg			
Crochet en métal	Crochet en métal	Acier		1,60E-02	kg			
Contenant	Bocal en verre	Verre		1,93E+00	kg			
Catégorie d'impact midpoint				Catégorie endoint				
Potentiel réchauffement climatique		Formation de particules fines	Epuisement des ressources minérales	Epuisement des ressources fossiles	Consommation d'eau			
kg CO2 eq		kg PM2.5 eq	kg Cu eq	kg oil eq	m3			
					DALY			
					species.yr			
					USD2013			
Pièces	1,04E-01	3,68E-04	1,00E-02	2,36E-02	5,70E-04	4,74E-07	5,06E-10	7,32E-03
Ressort	1,04E-01	3,68E-04	1,00E-02	2,36E-02	5,70E-04	4,74E-07	5,06E-10	7,32E-03
Masse	3,36E-02	3,58E-05	9,52E-05	8,26E-03	2,94E-04	8,40E-08	1,50E-10	2,34E-03
Crochet en métal	8,29E-02	2,94E-04	8,00E-03	1,89E-02	4,56E-04	3,79E-07	4,05E-10	5,86E-03
Contenant	1,19E+00	1,47E-03	2,41E-03	3,52E-01	7,82E-03	2,35E-06	6,00E-09	1,15E-01
Total	1,41E+00	2,17E-03	2,05E-02	4,03E-01	9,14E-03	3,29E-06	7,06E-09	1,30E-01

FIGURE 7.7 – Analyse des matériaux

Cette analyse a permis d'avoir une idée de l'impact des constituants utilisés sur les écosystèmes et de la gestion des ressources utilisées pour leur production. Dans le système 4, de l'eau a aussi été utilisée pour l'amortissement externe, l'eau étant de l'eau du robinet. Elle n'a donc pas d'impact ou un impact très faible que l'on peut négliger.

Dans l'analyse du cycle de vie ci-dessus, nous voyons que la catégorie : "Potentiel réchauffement climatique" est la phase qui a le plus d'impact pour notre système. Dans pratiquement toutes les phases de l'ACV (Analyse du Cycle de Vie), c'est le bocal en verre qui a l'impact le plus élevé. Ces impacts élevés sont dus à la masse importante du bocal.

Pour améliorer notre système, et donc réduire l'impact de celui-ci, une solution serait de remplacer le bocal en verre utilisé par un récipient en plastique bien plus léger. Cependant, l'analyse des matériaux a ses limites. Les données utilisées pour réaliser l'ACV sont des données généralisées. Il faut donc garder à l'esprit que l'ACV comporte des incertitudes.

Budget

Matériel	Prix
Ressorts	16.84 euros
Masses	0 euros
Planches en bois	15.29 euros (pour systèmes masse-ressort)
Total	32.13 euros

$$\dot{x}_1(t) = \frac{\ddot{x}_2}{2\beta} + \dot{x}_2 + \frac{\omega_0^2}{2\beta}(x_2 - x_1)$$

$$x_1(t) = \frac{\ddot{x}_2}{\omega_0^2} + \frac{2\alpha\dot{x}_2}{\omega_0^2} + x_2$$

$$\alpha = ln$$

où Δx_2 est le rapport entre deux maxima

$$\alpha = \frac{ln(\frac{x_2(t_1)}{x_2(t_2)})}{t_2 - t_1}$$

Conclusion

Pour conclure, nous allons revenir sur les différentes étapes du projet par lesquelles nous sommes passées.

Tout d'abord, une modélisation mathématique nous a permis d'étudier le comportement de différents systèmes masse-ressort dotés d'un amortissement externe ou interne et pour lesquels des tracking vidéo ont été faits. Ensuite, le modèle mathématique a été généralisé pour pouvoir décrire le mouvement simultané de la masse et de l'extrémité libre du ressort. Afin d'en vérifier la cohérence, des manipulations du système (élévations monotones) ont été faites et ont à nouveau été traitées avec l'outil de tracking.

A la fin du premier quadrimestre, la majorité des objectifs ont été atteints à l'exception de la vérification du modèle mathématique. Cette partie a été réalisée conjointement aux nouveaux objectifs du deuxième quadrimestre.

Les paramètres des galvanomètres ont par la suite été déterminés afin de pouvoir être utilisés dans les signaux à envoyer. Ceci a été fait de 3 façons différentes en utilisant notamment les résultats obtenus avant l'évaluation de mi-parcours comme l'équation des phaseurs. Ensuite, grâce à ces paramètres, les figures désirées pouvaient être tracées en partant de leurs équations paramétriques qui servaient de signaux de sortie à obtenir. Afin de trouver le signal d'entrée correspondant à chaque forme, les programmes de résolution des équations du mouvement et de calculs numériques ont été mis au point et ont permis de tracer le lemniscate de Bernoulli, le cercle et l'ellipse. L'hypotrochoïde et la courbe dynamique ont été calibrées correctement tandis que le carré a finalement été obtenu. De plus, plusieurs formes non obligatoires complexes ont été tracées avec succès comme le logo de l'EPB et d'autres symboles connus. A l'avenir, un show laser pourrait être envisagé sur base de ces résultats

Annexes

Annexe A

Méthodes de détermination des paramètres des galvanomètres

A.1 Première méthode - largeur de la courbe de résonance ($\delta\omega^*$)

Détaillée à la page 36 du chapitre 5 du cours de physique II, cette méthode consiste à mesurer la largeur de la courbe de résonance ($\delta\omega^*$) à la moitié de l'amplitude maximale notée $|X|_{max}$ et à utiliser les formules suivantes pour en déduire alpha.

Sachant que :

$$|X(\delta\omega^*)| = \frac{a}{2\omega_0\sqrt{\delta\omega^{*2} + \alpha^2}}$$

et que, à la moitié de l'amplitude maximale :

$$|X(\delta\omega^*)| \equiv \frac{|X|_{max}}{2} \quad \text{avec} \quad |X|_{max} = \frac{a}{2\omega_0\alpha}$$

on en déduit que :

$$\delta\omega^* = \sqrt{3}\alpha$$

Ici, la largeur de la courbe à la moitié de l'amplitude a été prise. En mesurant la largeur

aux 3/4 de l'amplitude¹, l'égalité suivante est obtenue :

$$9\delta\omega^*{}^2 = 7\alpha^2$$

Cette méthode ne fonctionne que si $\alpha \ll \omega_0$, or ne connaissant pas encore la nature des amortissements, elle n'est pas suffisante pour déterminer le coefficient d'amortissement. De plus, ces équations ne tiennent pas compte de la différence entre la pulsation propre du ressort (ω_0) et la pulsation correspondant au pic dans la courbe de résonance, elle la considère comme négligeable. Il est donc important de noter que le ω_0 présent ci-dessus ne correspond pas à la pulsation propre à proprement parler mais à la pulsation amortie. Or, il existe une différence non négligeable entre celles-ci. Cette différence est traduite par l'égalité :

$$\omega_d^2 = \omega_0^2 - \alpha^2$$

Ci-dessous est introduite une nouvelle méthode pour trouver la valeur de α .

A.2 Seconde méthode - l'ordonnée à l'origine ($|\underline{X}|_s$)

Cette deuxième méthode se base une fois de plus sur la courbe de résonance obtenue précédemment, plus précisément sur l'ordonnée à l'origine et le maximum de celle-ci. Les relations suivantes sont utilisées :

$$|\underline{X}_s| = \frac{a}{\omega_0^2} \quad \text{et} \quad |\underline{X}|_{max} = \frac{a}{2\omega_0\alpha}$$

Donc,

$$\alpha = \frac{|\underline{X}_s|\omega_0}{2|\underline{X}|_{max}}$$

En exprimant ω_0 en fonction de α et ω_d :

$$\alpha = \sqrt{\frac{|\underline{X}_s|^2\omega_d^2}{4|\underline{X}|_{max} - |\underline{X}_s|^2}}$$

La valeur de l'ordonnée à l'origine de la courbe de résonance ($|\underline{X}_s|$) correspond au dépla-

-
1. Cette méthode sera nommée la méthode 1 bis.
 2. Cette note de bas de page doit aussi être traitée comme un exposant

cement du laser lorsqu'on envoie une impulsion constante au galvanomètre. Or, lorsqu'un tel signal a été envoyé, le faisceau laser ne semblait pas bouger. Le $|X_s|$ a donc dû être déterminé graphiquement. Constatant que la pente de la courbe de résonance s'aplatit lorsqu'on s'approche d'une pulsation de 100 rad/s (la forte pente soudaine visible sur le graphique 5.2 à l'approche de 0 n'est pas à considérer ici), la valeur de l'amplitude pour une pulsation de 60 rad/s a été considérée comme l'ordonnée à l'origine. Cette manœuvre pouvant être source d'imprécisions, nous verrons une autre manière de déterminer $|X_s|$.

Le point faible de cette technique est qu'elle se base sur une approximation de $|X_s|$. Dès lors, une troisième méthode sera utilisée pour confirmer les valeurs obtenues.

A.3 Troisième méthode - protocole des paramètres

Cette troisième méthode se réfère au protocole des paramètres détaillé au point 1.2.5 applique aux graphiques de la position en fonction du temps des laser quand une brève impulsion est envoyée (voir graphiques ci-dessous). Ce protocole est surtout pratique pour déterminer les coefficients α . Pour trouver les autres membres de l'équation (ω_0 , a , $|X_s|$), les méthodes détaillées plus haut seront utilisées. Pour vérifier les résultats, le module solver d'excel a été employé, obtenant ainsi les paramètres les plus précis possibles.

Annexe B

Équations des figures

B.1 Cercle

$$\begin{cases} x(t) = r \cos(\omega t) \\ y(t) = r \sin(\omega t) \end{cases} \quad r \in \mathbb{R}, \quad \omega, t \in \mathbb{R}^+ \quad (\text{B.1})$$

B.2 Ellipse

$$\begin{cases} x(t) = a \cos(\omega t) \\ y(t) = b \sin(\omega t) \end{cases} \quad a, b \in \mathbb{R}, \quad \omega, t \in \mathbb{R}^+ \quad (\text{B.2})$$

B.3 Hypotrochoïde

Une hypotrochoïde est une courbe décrite par un point fixé au rayon r d'un cercle tangent intérieurement à un deuxième cercle de rayon $R > r$ et qui se déplace le long de ce dernier. Écrivons les équations paramétriques avec une pulsation $\omega = 2\pi$ à laquelle est envoyé le signal.

$$\begin{cases} x(t) = l \cos\left(\frac{R-r}{r}\omega t\right) + (R - r) \cos(\omega t) \\ y(t) = -l \sin\left(\frac{R-r}{r}\omega t\right) + (R - r) \sin(\omega t) \end{cases} \quad t, \omega \in \mathbb{R}^+ \quad \text{et} \quad r, R, d \in \mathbb{R} \quad (\text{B.3})$$

où l est la longueur du segment qui relie le point qui trace l'hypotrochoïde au cercle de rayon

r.

Cette figure est donc tracée par le mouvement périodique d'un point. De fait, une certaine hypotrochoïde a, en fonction de R et r, une certaine période T après laquelle le point reprend sa position initiale avant de tracer la courbe une deuxième fois. Pour trouver cette période, prenons par exemple la coordonnée x(t) de l'hypotrochoïde.

Celle-ci est une somme de deux cosinus avec des arguments différents, or la période de $\cos(k)$ vaut $\frac{2\pi}{k}$. De plus, le quotient $\frac{R-r}{r}$ vaut, après la division, un certain nombre rationnel qui peut ensuite se réécrire comme un autre quotient entre deux naturels $\frac{d}{n}$ ($d, n \in \mathbb{N}$). Il en découle donc que la période du premier cosinus vaut $\frac{2\pi n}{d\omega}$. Pour déterminer la période de la coordonné $x(t)$, il suffit dès lors de trouver le plus petit commun multiple de $\frac{2\pi n}{d\omega}$ et $\frac{2\pi}{\omega}$ (qui est la période du deuxième cosinus), c'est à dire le temps pris pour compléter les périodes des deux cosinus un nombre entier de fois. Or,

$$\begin{aligned}\frac{2\pi n}{\omega d} \cdot d &= \frac{2\pi n}{\omega} \\ \frac{2\pi}{\omega} \cdot n &= \frac{2\pi n}{\omega}\end{aligned}$$

$\frac{2\pi n}{\omega}$ est donc la période de $x(t)$. Comme l'expression de $y(t)$ est analogue à celle de $x(t)$, le même raisonnement peut y être appliqué et la période du signal complet de l'hypotrochoïde est $\frac{2\pi n}{\omega}$.

La fréquence à laquelle est envoyé le signal se cache dans l'expression de la période ci-dessus et plus précisément dans le ω . Pour rappel, T est le temps que prend une hypotrochoïde complète pour se répéter. Cependant, la durée du signal envoyé par l'utilisateur du système peut être quelconque et ne vaut pas forcément T mais plutôt la valeur correspondant au maximum t_f de la liste des abscisses temporelles t , par exemple 4 secondes. Il faut donc trouver la fréquence pour obtenir un tour complet en un temps t_f . Pour cela, il suffit d'égaliser l'expression de T à t_f .

$$\begin{aligned}\frac{2\pi n}{\omega} &= \frac{2\pi n}{2\pi f} = t_f \\ f &= \frac{n}{t_f}\end{aligned}$$

Autrement, le signal s'achèverait en un certain instant avant que la forme soit complétée et recommencerait à la tracer depuis cet instant quelconque. La fréquence à donner à l'hypotrochoïde

est donc égale à $\frac{n}{t_f}$. Enfin, il est utile de mentionner que pour certains choix des paramètres R , r et d , il est possible d'obtenir un cercle et une ellipse avec l'équation de l'hypotrochoïde. Si $R = 2r$ et $d \neq 0$ on obtient une ellipse et $\forall R, r$ si $d = 0$ on obtient un cercle de rayon $R - r$.

Voici le graphique obtenu à l'aide des équations ci-dessus :

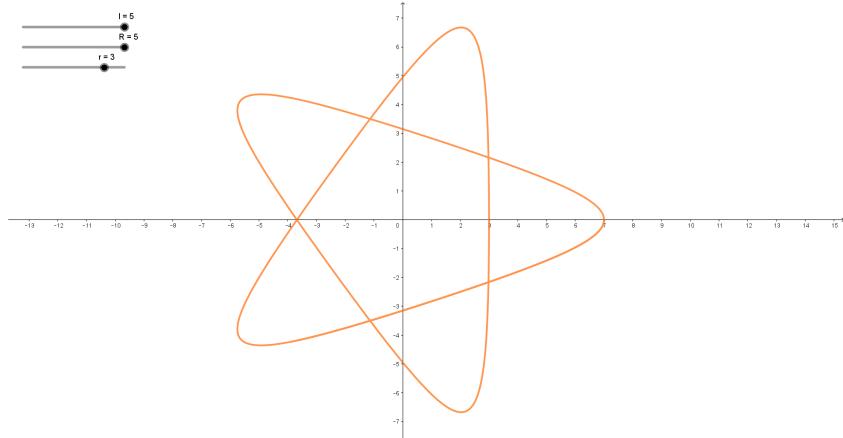


FIGURE B.1 – L'hypotrochoïde étoilée

Quand on modifie les paramètres :

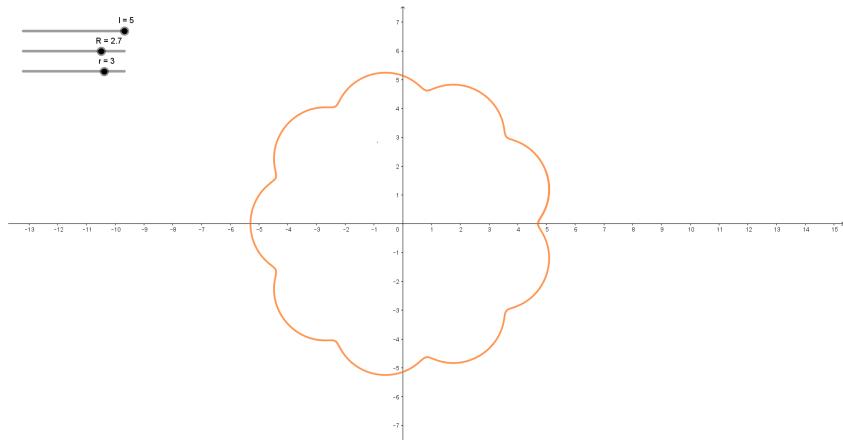


FIGURE B.2 – L'hypotrochoïde en nuage

B.4 Figure dynamique

La figure dynamique choisie a été celle d'une ellipse qui pivote en perspective. Afin de donner l'illusion d'une forme qui se déplace, il convient d'introduire, dans l'équation de l'ellipse

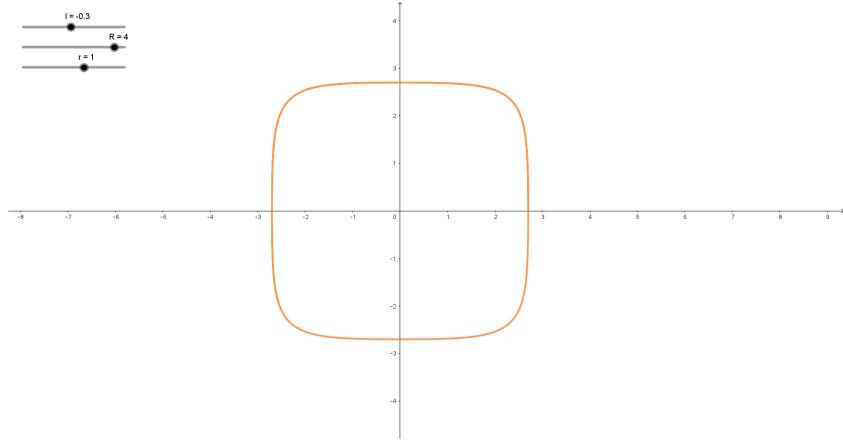


FIGURE B.3 – L'hypotrochoïde carrée

un coefficient de déformation k

$$\begin{cases} x(t) = a \cdot \cos(k\omega t) \\ y(t) = b \cdot \sin(\omega t) \end{cases} \quad a, b, \omega, k \in \mathbb{R}, \quad t \in \mathbb{R}^+$$

L'une des deux coordonnées ayant son argument multiplié par k , celle-ci a toujours une "avance" dans la mesure où elle se trouve toujours en $\cos(k\omega t)$ là où l'autre se trouve $\cos(\omega t)$. Ici se pose le même problème que pour l'hypotrochoïde puisqu'il faut calibrer la fréquence pour que le signal se répète en sa valeur périodique. Dans le cas d'une mauvaise fréquence, la figure présente une sorte de "discontinuité" puisque l'ellipse se déforme et recommence au mauvais moment.

En toute cohérence, le même raisonnement que pour l'hypotrochoïde peut être appliqué ici car la période de $a\cos(k\omega t)$ est $\frac{2\pi}{k\omega}$ et celle de $b\sin(\omega t)$ est $\frac{2\pi}{\omega}$. Il faut donc à nouveau trouver quand ces deux périodes sont complétées un nombre entier de fois.

$$k = \frac{d}{n} \quad T = \frac{2\pi n}{\omega}$$

$$\frac{2\pi n}{2\pi f} = t_f \quad f = \frac{n}{t_f}$$

Avec ($d, n \in \mathbb{N}$)

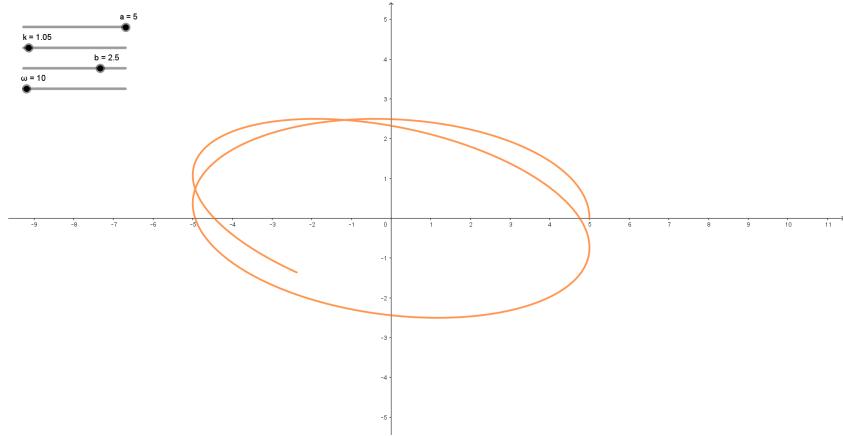


FIGURE B.4 – Une partie de l'animation

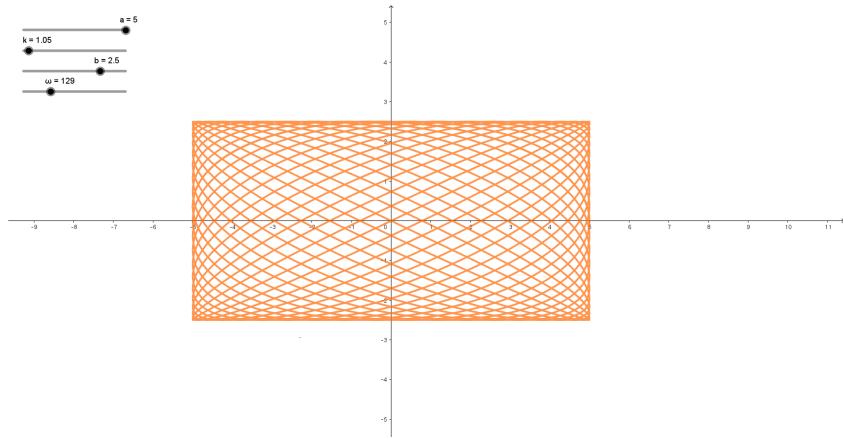


FIGURE B.5 – L'animation complète

B.5 Lemniscate de Bernoulli

$$\begin{cases} x(t) = c \cdot \frac{\sin(\omega t)}{1+\cos^2(\omega t)} \\ y(t) = c \cdot \frac{\sin(\omega t)\cos(\omega t)}{1+\cos^2(\omega t)} \end{cases} \quad t, c, \omega \in \mathbb{R}^+ \quad (\text{B.4})$$

B.6 Carré

Pour déterminer les équations paramétriques du signal carré (par extension rectangulaire), nous avons effectué le développement suivant. Considérons que notre forme carrée est parcourue par un vecteur tournant à vitesse angulaire ω constante autour de l'origine d'un repère cartésien. Nous remarquons directement que les équations seront déterminées par morceau en fonction du quadrant occupé par l'angle α , comme illustré sur le schéma ci-dessous.

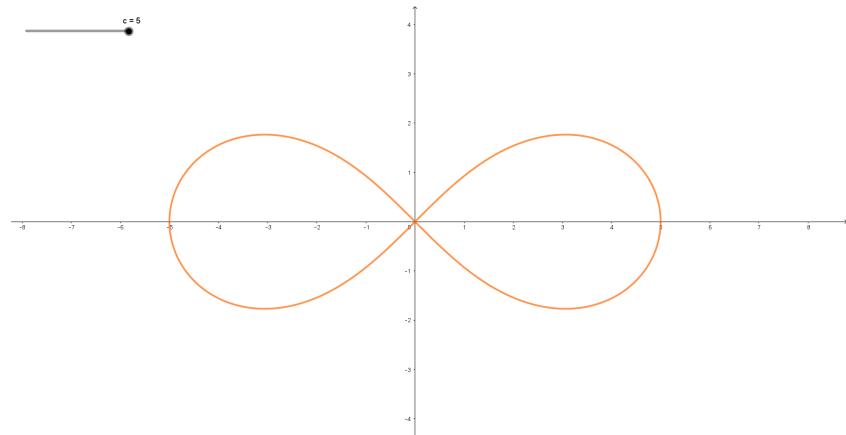
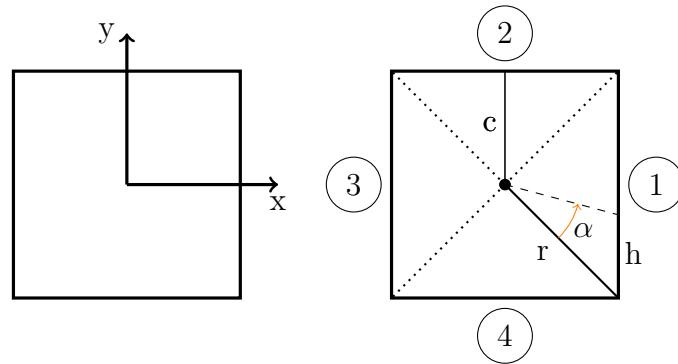


FIGURE B.6 – La lemniscate de Bernoulli



Avec :

- $\omega t = \alpha(t)$ l'angle balayé par ce vecteur depuis le segment r .
- h_i la longueur du segment défini par le coin débutant le quadrant courant i et l'extrémité du vecteur tournant.
- c la longueur du demi côté du carré.
- r le rayon du cercle circonscrit au carré avec $r = \sqrt{2}c$.

- Quadrant 1

$$\begin{cases} x_1(t) = c \\ y_1(t) = -c + h_1(t) \end{cases} \quad (B.5)$$

En appliquant la loi des sinus sur les côtés r et h_1 , nous déterminons $h_1(t)$:

$$\frac{h_1}{\sin(\alpha)} = \frac{r}{\sin(3\pi/4 - \alpha)} \Rightarrow h_1 = c \cdot \frac{2\sin(\alpha)}{\cos(\alpha) + \sin(\alpha)}$$

Nous déterminons ainsi les équations paramétriques du signal carré sur le premier quadrant :

$$\iff \begin{cases} x_1 = c \\ y_1 = c \cdot \left(\frac{2\sin(\alpha)}{\cos(\alpha) + \sin(\alpha)} - 1 \right) \end{cases} \quad (\text{B.6})$$

Et nous procédons de même pour les autres quadrants.

- Quadrant 2

$$\begin{cases} x_2(t) = c - h_2(t) \\ y_2(t) = c \end{cases} \quad (\text{B.7})$$

$$\iff \begin{cases} x_2 = c \cdot \left(1 - \frac{2\cos(\alpha)}{\cos(\alpha) - \sin(\alpha)} \right) \\ y_2 = c \end{cases} \quad (\text{B.8})$$

- Quadrant 3

$$\begin{cases} x_3(t) = -c \\ y_3(t) = c - h_3(t) \end{cases} \quad (\text{B.9})$$

$$\iff \begin{cases} x_3 = -c \\ y_3 = c \cdot \left(1 - \frac{2\sin(\alpha)}{\cos(\alpha) + \sin(\alpha)} \right) \end{cases} \quad (\text{B.10})$$

- Quadrant 4

$$\begin{cases} x_4(t) = -c + h_4(t) \\ y_4(t) = c \end{cases} \quad (\text{B.11})$$

$$\iff \begin{cases} x_4 = c \cdot \left(\frac{2\cos(\alpha)}{\cos(\alpha)-\sin(\alpha)} - 1 \right) \\ y_4 = -c \end{cases} \quad (\text{B.12})$$

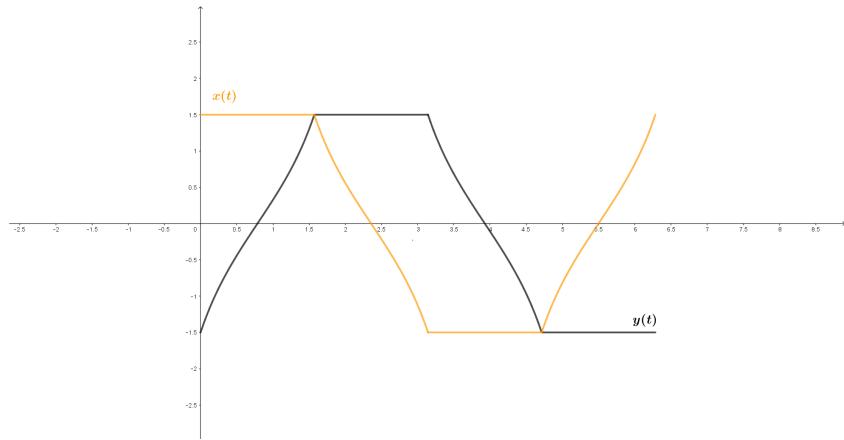


FIGURE B.7 – Fonctions du signal carré sur une période

Annexe C

Transfert

"""

Auteurs : Hanssens Dimitri & Matin Joel

Date : février - mars 2021

Contient les fonctions utilisées dans le calcul des points du signal

→ d'entrée

sur base du signal de sortie désiré

"""

```
def derivative(x, t):
```

"""

Renvoie la dérivée de la fonction $x(t)$

Entrée : liste des valeurs de x et de t

"""

```
res = []
```

```
dt = (max(t) - min(t)) / (len(t) - 1) # On calcule l'accroissement entre  
→ deux points de  $t$ 
```

```
for i in range(len(x) - 1):
```

```
slope = (x[i + 1] - x[i]) / dt # On calcule la pente entre deux  
→ points
```

```

    res.append(slope)

    return res

def external(x2,dx2,ddx2,a,wo):
    """
    On calcule le résultat de l'équation  $x_1 = x_2''/(w_0^2) + 2*a/(w_0^2)*x_2' +$ 
    →  $x_2$ 

    Avec  $x_2$  la fonction de sortie et  $x_1$  la fonction d'entrée
    """

    res = []
    for i in range(len(ddx2)):                      # On rajoute un à
        → un (sur base de la plus petite liste, ddmass)
        x1_i = ddx2[i]/(wo**2)+2*a/(wo**2)*dx2[i]+x2[i]      # Traduction de
        → l'équation
        res.append(x1_i)
    return res

def internal(x2, dx2, ddx2, a, wo, t):
    """
    On calcule le résultat de l'équation  $x_1' = x_2''/(2a) + x_2' +$ 
    →  $(w_0^2)/(2a)*(x_2-x_1)$ 

    Par la méthode d'Euler

    Avec  $x_2$  la fonction de sortie et  $x_1$  la fonction d'entrée
    """

    ys = [x2[0]]  # ys initial (= x2 initial)
    for i in range(1, len(ddx2)):  # Plus petite liste = ddx2
        dt = t[1]  # Pas
        y = ys[-1] + dt * (wo ** 2 / (2 * a) * (x2[i - 1] - ys[i - 1]) + dx2[i
        → - 1] + ddx2[i - 1] / (2 * a)))
        ys.append(y)
    return ys

```

Annexe D

Geometry

"""""

Auteur.e.s : Hanssens Dimitri, Matin Joel & Lindemann Theresa

Date : février - mars 2021

Contient les fonctions utilisées dans le calcul des signaux de sorties

→ définis selon leurs équations paramétriques
déterminées au préalable

""""

```
import numpy as np
from fractions import Fraction

def square(cx, cy, freq, f):
    """
    Sur base des longueurs des demi-côtés du rectangle, créé le signal de
    → sortie rectangulaire
    engendré par un vecteur tournant à vitesse angulaire constante ( $\omega =$ 
    →  $2\pi/T$  avec  $T$  la période du signal)
    autours de l'origine
    """

```

```

Y = []
X = []

n = int(freq*(max(f)-2)) # Nombre de périodes au sein du signal de sortie
t = (max(f)-2) // n      # durée en frame d'une période
# On calcule la fréquence (en rad/frame) à insérer dans nos fcts trig.
→ (sur base de T = max(t))
w = (1 / t) * 2 * np.pi
for k in range(n):
    for fi in range(t):
        # On travaille quadrant par quadrant sur un cycle complet en
        → fonction de la valeur associée à t
        if fi <= t / 4:
            xi = cx
            yi = cy * (2 * np.sin(fi * w) / (np.cos(fi * w) + np.sin(fi *
            → w)) - 1)
            X.append(xi)
            Y.append(yi)
        elif t / 4 <= fi <= t / 2:
            xi = cx * (1 - 2 * np.cos(fi * w) / (np.cos(fi * w) -
            → np.sin(fi * w)))
            yi = cy
            X.append(xi)
            Y.append(yi)
        elif t / 2 <= fi <= 3 * t / 4:
            xi = - cx
            yi = cy * (1 - 2 * np.sin(fi * w) / (np.cos(fi * w) +
            → np.sin(fi * w)))
            X.append(xi)
            Y.append(yi)
        elif 3 * t / 4 <= fi:

```

```

        xi = cx * (2 * np.cos(fi * w) / (np.cos(fi * w) - np.sin(fi *
        ↵   w)) - 1)
        yi = - cy
        X.append(xi)
        Y.append(yi)

    return X, Y

def circle(r,freq,f):
    """
    Sur base du rayon du cercle, crée le signal de sortie circulaire
    engendré par un vecteur tournant
    à vitesse angulaire constante autour de l'origine
    """
    w = freq*(2*np.pi)
    X = r*np.cos(w*f)
    Y = r*np.sin(w*f)
    return X, Y

def ellipse(c,d,freq,f):
    """
    Sur base des longueurs c et d des demi-axes, crée le signal de sortie
    elliptique engendré par un vecteur tournant
    à vitesse angulaire constante autour de l'origine. Dans le cas
    particulier où c = d, la figure de sortie sera
    un cercle
    """
    w = freq*(2*np.pi)
    X = c * np.cos(w * f)
    Y = d * np.sin(w * f)
    return X, Y

```

```

def anim(c,d,k,freq,f):
    """
    Sur base des longueurs c et d des demi-axes, crée le signal de sortie
    → elliptique dont la forme varie
    au cours du temps t engendré par un vecteur tournant à vitesse angulaire
    → constante autours de l'origine
    """

    w = freq * (2 * np.pi)
    X = c * np.cos(k * w * f)
    Y = d * np.sin(w * f)
    return X, Y


def hypotrochoide(d,R,r,freq,f):
    """
    Sur base des paramètres d, R, r de l'hypotrochoïde, crée le signal de
    → sortie de la figure
    engendrée par un vecteur tournant à vitesse angulaire constante autours
    → de l'origine
    """

    w = freq*(2*np.pi)
    X = (R-r)*np.cos(w*f)+d*np.cos(w*f*(R-r)/r)
    Y = (R-r)*np.sin(w*f)-d*np.sin(w*f*(R-r)/r)
    return X, Y


def lemniscate(c,freq,f):
    """
    Sur base des paramètres c et du lemniscate, crée le signal de sortie de
    → la figure
    engendrée par un vecteur tournant à vitesse angulaire constante autours
    → de l'origine
    """

```

```

w = freq*(2*np.pi)

X = c*np.sin(w*f)/(1+np.cos(w*f)**2)
Y = c*np.sin(w*f)*np.cos(w*f)/(1+np.cos(w*f)**2)

return X,Y

def calcul_freq_hypotro(R, r,taux, duree_signal_s):
    """
    Calcule la fréquence en seconde du signal de l'hypotrochoïde à envoyer
    sur base des paramètres R et r de la figure
    """
    q = str(Fraction((R-r)/r).limit_denominator(1000)) # Renvoie le str du
    → rationnel  $(R-r)/r$  sous forme de fraction (approximée)
    n = q.split("/") [1] # On prend le dénominateur de la fraction
    freq_s = int(n)/duree_signal_s # duree_signal_s =  $T = 2\pi n/w = n/f$  (cf.
    → explication rapport)
    return freq_s*taux

def calcul_freq_anim(k,taux, duree_signal_s):
    """
    Calcule la fréquence en seconde du signal de l'ellipse animée à envoyer
    sur base du paramètre psi de la figure
    """
    if type(k) == float:
        q = str(Fraction(k).limit_denominator(1000)) # Renvoie le str du
        → rationnel k sous forme de fraction (approximée)
        n = q.split("/") [1] # On prend le dénominateur de la fraction
    elif type(k) == int:
        n = 1
    freq_s = int(n)/duree_signal_s # duree_signal_s =  $T = 2\pi n/w = n/f$  (cf.
    → explication rapport)
    return freq_s*taux

```

Annexe E

Read_XML

"""

Auteur : Hanssens Dimitri

Date : février - mars 2021

*Sur base d'un fichier .svg d'une image vectorisée, renvoie les coordonnées
→ de chacun des noeuds de l'objet vectoriel.*

Conditions d'utilisation : Les noeuds doivent tous être reliés linéairement.

Il faut donc applatir les courbes au préalable.

*Pour ce faire, il faut augmenter le nombre de noeuds (précision) et rendre
→ rectiligne chaque segment
sur l'ensemble de l'image via le logiciel de dessin vectoriel.*

"""

```
def find_path(file):
    """
    Cherche et enregistre le chemin de noeuds dans le fichier SVG entré en
    → paramètre
    Renvoie le chemin brut
    """
```

```

for line in open(file, 'r', encoding="utf-8"):

    # La ligne correspondante au chemin commence par l'assignation de la
    ↳ variable d au string recherché

    if line.strip().find('d') == 0: # Si d est le premier caractère de
    ↳ la ligne

        res = line.strip().strip()[3:-1] # On assigne le chemin net en
        ↳ nettoyant les bords des guillemets

return res


def clear(str):

    """
    Nettoye le chemin de noeuds du fichier .svg
    Renvoie une liste contenant les coordonnées et la commande associées
    → pour chaque noeud
    """

    lst = str.strip().split()
    res = []
    balise = ["Z", "z"]
    commande = ["H", "h", "V", "v", "Z", "z", "M", "m", "L", "l"] # Regroupe
    ↳ les commandes de noeuds dans une liste
    dico2 = {1: str} # Indique la balise activée au moyen d'un dictionnaire
    for elem in lst:
        if elem in balise:
            # Rajoute la balise permettant de renouveler le chemin depuis le
            ↳ point de départ
            res.append("Start")
        if elem in commande:
            dico2[1] = elem # Active la nouvelle commande en remplaçant la
            ↳ dernière active
        if not elem.isalpha():

```

```

    res.append(elem + "," + dico2[1]) # Rajoute l'élément
    ↪ (x,y,balise) sous forme de string dans la liste

return res

def coord(path):
    """
    Récolte les coordonnées cartésiennes en x et y du chemin sur base des
    ↪ informations liées à chaque noeud
    Renvoie les coordonnées traitables sous forme de liste
    """

# Définition d'un point de référence (vecteur nul)
X = [0]
Y = [0]
start = 1

for elem in path:
    coord = elem.split(",")
    # Si la coordonnée est une balise "closepath" : Entame un nouveau
    ↪ chemin depuis le dernier point de départ
    if elem == "Start":
        X.append(X[start])
        Y.append(Y[start])
        start = len(X) # Réinitialise l'indice du point de départ
    # Si la balise est une lettre minuscule, les coordonnées du noeud
    ↪ indiquent le déplacement à effectuer
    # pour atteindre les coordonnées du point correspondant
    if coord[-1].islower():
        if coord[-1] not in ["h", "v"] : # Cas général
            x = X[-1] + float(coord[0])
            y = Y[-1] - float(coord[1])
            X.append(x)

```

```

Y.append(y)

elif coord[-1] == "h": # Tracé horizontal : On ne dispose que du
    # déplacement en x
    x = X[-1] + float(coord[0])
    y = Y[-1]
    X.append(x)
    Y.append(y)

elif coord[-1] == "v": # Tracé vertical : On ne dispose que du
    # déplacement en y
    x = X[-1]
    y = Y[-1] - float(coord[0])
    X.append(x)
    Y.append(y)

# Si la balise est une lettre majuscule, les coordonnées du noeud
# sont les coordonnées du point souhaité

if coord[-1].isupper():

    if coord[-1] not in ["H", "V"] : # Cas général
        X.append(float(coord[0]))
        Y.append(-float(coord[1]))

    elif coord[-1] == "H": # Tracé horizontal : On ne dispose que de
        # la coord. en x
        X.append(float(coord[0]))
        Y.append(Y[-1])

    elif coord[-1] == "V": # Tracé vertical : On ne dispose que de
        # la coord. en y
        X.append(X[-1])
        Y.append(-float(coord[0]))

del X[0], Y[0] # Supprime le point de référence
return X, Y

```

```

def center(x,y):
    """
    Centre l'objet (défini par ses coordonnées) en l'origine
    """

    X = []
    Y = []
    a = -(max(x)-min(x))/2 - min(x)                      # Vecteur déplacement
    ↪   horizontal
    b = (max(y) - min(y))/2 - max(y)                      # Vecteur déplacement
    ↪   vertical
    for xi in x:
        X.append(xi + a)
    for yi in y:
        Y.append(yi + b)
    return X,Y

```

Annexe F

Optimisation

"""

Auteur : Hanssens Dimitri & Lindemann Theresa

Date : mars 2021

Optimise les listes de points générées par lecture d'un fichier .svg en les
→ répartissant à égale distance
tout en conservant au mieux les propriétés de la figure d'origine

"""

```
import numpy as np
```

```
def distances(x, y):  
    """  
    Calcule les trois données suivantes :  
        - La distance totale de la courbe segmentée en sommant les distances  
        → entre deux points  
        - La plus petite distance entre deux points parmi tous les points de  
        → calcul*  
        - La plus grande distance entre deux points parmi tous les points de  
        → calcul
```

```

* Cette distance est minorée par une certaine contrainte pour éviter
↪ d'atteindre les limites de valeur imposées par
python dans les calculs ultérieurs.

Renvoie les trois valeurs mesurées dans le même ordre que mentionné
↪ ci-dessus

"""

# Initialisation des calculs

d1 = np.sqrt((x[1] - x[0]) ** 2 + (y[1] - y[0]) ** 2)
res_tot = d1
res_min = d1
res_max = d1

for i in range(2, len(x)): # On réitère le calcul d1 pour les autres
    ↪ points et on discute de la valeur obtenue
        # Distance entre deux points
    di = np.sqrt((x[i] - x[i - 1]) ** 2 + (y[i] - y[i - 1]) ** 2)
        # On définit arbitrairement une borne inférieure à la valeur de
    ↪ d_min sur base du dernier d_max enregistré :
        if di < res_min and di > res_max / 10e6:
            res_min = di
        if di > res_max:
            res_max = di
    res_tot += di

return res_tot, res_min, res_max

def rajout(x, y):
    """

    On rajoute le nombre de points nécessaire par interpolation linéaire
    ↪ pour une répartition équidistante
    entre les points définissant la figure initiale, sur base de la distance
    ↪ minimale

```

séparant deux points consécutifs donnés.

Renvoie les nouvelles coordonnées des points de la figure séparés d'une
→ distance égale

"""

```
xs = []
ys = []
d_min = distances(x, y)[1] # Calcul de la distance minimale entre deux
→ points
for i in range(1, len(x)):
    di = np.sqrt((x[i] - x[i - 1]) ** 2 + (y[i] - y[i - 1]) ** 2)
    rapport = int(di / d_min) # On calcule le nombre de points à
→ rajouter entre nos deux points de calcul
    if rapport >= 1:
        x_inter, y_inter = interpolation(x[i], y[i], x[i - 1], y[i - 1],
→ rapport) # On génère les points souhaités
        xs.extend(x_inter)
        ys.extend(y_inter)
return xs, ys
```

def interpolation(x2, y2, x1, y1, nb):

"""

Rajoute le nombre de points indiqué en paramètre entre deux points sur
→ base de leur coordonnée
et par interpolation linéaire

"""

```
xs = []
ys = []
# Calcul des composantes du vecteur directeur u de la droite passant par
→ (x1,y1) et (x2,y2)
ux = (x2 - x1) / nb
```

```

uy = (y2 - y1) / nb
# Rajout pour tout t entier entre 0 et nb non inclus
for t in range(nb):
    xs.append(x1 + t * ux)
    ys.append(y1 + t * uy)
return xs, ys

def réduction(x, y, N):
    """
    Sur base du nombre N de points demandé, sélectionne N' points parmi les
    points donnés en paramètre tel que
    ces derniers sont répartis à distance égale et N' est le plus proche
    possible de N (par excès)
    Renvoie les coordonnées sous forme de liste des N' points équidistants
    """
    xs = []
    ys = []
    # Si le nombre de point souhaité est moindre que le nombre de points
    # appelé par la fonction
    if N < len(x):
        pas = len(x)//N
        # On itère N' fois
        for i in range(0, len(x), pas):
            xs.append(x[i])
            ys.append(y[i])
    # Sinon, on garde les points appelés
    else:
        xs = x
        ys = y
return xs, ys

```

```

def optimisation(x, y, N):
    """
    Renvoie N' points équidistants conservant le plus fidèlement possible
    les propriétés de la figure initiale définie sur base des points appelés
    → par la fonction
    """

    # Etape 1 : rajout des points pour une répartition équidistante
    x1, y1 = rajout(x, y)

    # Etape 2 = réduction du nombre de points en fonction de celui souhaité
    x2, y2 = réduction(x1, y1, N)

    return x2, y2


def decalage(x,y):
    """
    Renvoie un nouveau signal en déphasage avec le signal envoyé en entrée.
    Le décalage temporel est donné arbitrairement à 1/10 de la longueur
    totale en frame du signal.
    """

    phi = len(x)//30
    X = []
    Y = []

    # On rajoute au début les derniers points du signal appelé
    for i in range(phi,len(x)):
        X.append(x[i])
        Y.append(y[i])

    # On rajoute à la fin les premiers points du signal appelé
    for i in range(phi):
        X.append(x[i])
        Y.append(y[i])

```

```
return X,Y
```

Annexe G

Interface

----- Voici une liste du type de figure proposé par le logiciel :

↔ -----

- Carré
- Cercle
- Ellipse
- Ellipse animée
- Hypotrochoïde *
- Lemniscate de Bernoulli
- Forme complexe **

* Dont voici les paramètres pour trois figures différentes :

- Etoile à 5 branches :

l = 5

R = 5

r = 3

- Empreinte de gâteau / nuage :

$l = 5$

$R = 2.7$

$r = 3$

- Carré aux côtés arrondis :

$l = -0.3$

$R = 4$

$r = 1$

** Il faudra alors vous assurer que votre fichier SVG contenant votre figure

↪ se trouve bien dans le même répertoire

que celui du logiciel. Ce dernier doit contenir un nombre raisonnable de

↪ noeuds (entre 800 et 2000,

selon la complexité de la figure) pour obtenir un résultat correct tout en

↪ évitant des calculs trop longs.

Le programme actuel ne traite que les objets vectoriels dont les noeuds sont

↪ reliés par des segments rectilignes.

Annexe H

Main

"""

Auteur : Hanssens Dimitri

Date : mars 2021

"""

```
import Geometry as gm
import Read_XML as xml
import Transfert as tf
import Optimisation as opti
import numpy as np
import matplotlib.pyplot as plt
```

Définitions de fonctions main :

```
def divide(lst, k):
```

"""

Renvoie un élément sur <k> des éléments de la liste appelée en paramètre

"""

```
res = []
```

```
for i in range(0, len(lst), k):
```

```
    res.append(lst[i])
```

```

return res

def amplitude(x,y,c,d):
    """
    Borne le signal envoyé en paramètre entre les valeurs -c et c
    """

    X = []
    Y = []

    a = max(x)
    b = max(y)

    n = 1

    if a > ampli_max:
        n = a/ampli_max

    if b > ampli_max and b > a:
        n = b/ampli_max

    for i in range(len(x)):
        X.append(x[i]/(n*c))
        Y.append(y[i]/(n*d))

    return X,Y

```

```

def ajustement(lst):
    """
    Ajuste le signal envoyé en paramètre en tranchant les valeurs
    → supérieures à 1 en valeur absolue
    Renvoie un signal borné entre -1 et 1
    """

    res = []
    for elem in lst:
        if elem > 1:

```

```

    res.append(1)

elif elem < -1:
    res.append(-1)

else:
    res.append(elem)

return res

# Constantes globales

freq_laser = 44100

duree_param = 0.1 # Durée définie arbitrairement (La fréquence d'envoi doit
                  → être un multiple de 10)

duree_anim = 10      # Durée définie arbitrairement (max : 10 s)

duree_complexe = 0.07 # Durée définie expérimentalement pour obtenir une image
                      → complète dans la majorité des cas

k = 1.005            # Coefficient défini arbitrairement selon la rapidité de
                      → l'animation

# Paramètres des galvanomètres

wo_h = 118.03 * (2 * np.pi) / freq_laser
wo_v = 175.306 * (2 * np.pi) / freq_laser
a_h = 30.267 * (2 * np.pi) / freq_laser
a_v = 33.69 * (2 * np.pi) / freq_laser

# Coefficients de déformabilité des galvanomètres

a = 541
b = 713
ampli_max = 70

```

```

# Commandes avec l'utilisateur (forme souhaitée, nom du fichier de sortie
↪ souhaité, taille en temps, ...)
for line in open("Interface.txt", 'r', encoding='utf-8'):
    print(line.strip())

figure = input('Veuillez encoder votre choix (sans accent et en minuscule) :
↪ ').strip()
if figure == "carre":
    freq_s = float(input("Fréquence du signal envoyé en s^-1 (multiple de 10
↪ : "))
    freq_f = freq_s / freq_laser # On passe de s^-1 à frame^-1
    duree_f = int(duree_param * freq_laser)
    f = np.arange(duree_f+2)
    cx = float(input("Longueur du demi-côté en x du rectangle (max. 70) : "))
    cy = float(input("Longueur du demi-côté en y du rectangle (max. 70) : "))
    print("Calcul en cours ...")
    xh, xv = gm.square(cx,cy,freq_f,f)
    xh, xv = opti.decalage(xh,xv)

elif figure == "cercle":
    freq_s = float(input("Fréquence du signal envoyé en s^-1 (multiple de 10
↪ : "))
    freq_f = freq_s / freq_laser # On passe de s^-1 à frame^-1
    duree_f = int(duree_param * freq_laser)
    f = np.arange(duree_f + 2)
    r = float(input("Rayon du cercle (max. 70) : "))
    xh,xv = gm.ellipse(r,r,freq_f,f)

elif figure == "ellipse":
    freq_s = float(input("Fréquence du signal envoyé en s^-1 (multiple de 10
↪ : "))

```

```

freq_f = freq_s / freq_laser # On passe de s^-1 à frame^-1
duree_f = int(duree_param * freq_laser)
f = np.arange(duree_f + 2)
c = float(input("Longueur du demi-axe horizontal (max. 70) : "))
d = float(input("Longueur du demi-axe vertical (max. 70) : "))
xh,xv = gm.ellipse(c,d,freq_f,f)

elif figure == "ellipse animee":
    taux = int(input("Taux de rafraîchissement souhaité (1-10) : "))
    freq_s = gm.calcul_freq_anim(k,taux,duree_anim)
    freq_f = freq_s / freq_laser # On passe de s^-1 à frame^-1
    duree_f = int(duree_anim * freq_laser)
    f = np.arange(duree_f + 2)
    c = float(input("Longueur du demi-axe horizontal (max. 70) : "))
    d = float(input("Longueur du demi-axe vertical (max. 70) : "))
    xh,xv = gm.anim(c,d,k,freq_f,f)

elif figure == "hypotrochoide":
    l = 10*float(input("Valeur de l : "))
    R = 10 * float(input("Valeur de R : "))
    r = 10*float(input("Valeur de r : "))
    taux = int(input("Taux de rafraîchissement souhaité (1-10) : "))
    freq_s = gm.calcul_freq_hypotro(R,r,taux,duree_param)
    freq_f = freq_s / freq_laser # On passe de s^-1 à frame^-1
    duree_f = int(duree_param * freq_laser)
    f = np.arange(duree_f + 2)
    xh,xv = gm.hypotrochoide(l,R,r,freq_f,f)

elif figure == "lemniscate de bernoulli":
    freq_s = float(input("Fréquence du signal envoyé en s^-1 (multiple de 10)
    ↵ : "))

```

```

freq_f = freq_s / freq_laser # On passe de s^-1 à frame^-1
duree_f = int(duree_param*freq_laser)
f = np.arange(duree_f + 2)
c = float(input("Amplitude en x (max. 70) : "))
xh,xv = gm.lemniscate(c,freq_f,f)

elif figure == "forme complexe":
    file_out = input("Veuillez entrer le nom du fichier .svg :
    ↳ ").strip() + ".svg"
    print("Calcul en cours ...")
    duree_f = int(duree_complexe * freq_laser)
    f = np.arange(duree_f + 2)
    path = xml.clear(xml.find_path(file_out))
    x,y = xml.coord(path)
    xh,xv = xml.center(x,y)
    xh,xv = opti.optimisation(xh,xv,duree_f)
    xh,xv = opti.decalage(xh,xv)

# Calcul des signaux d'entrée
xh,xv = amplitude(xh,xv,a,b)

dxh = tf.derivative(xh,f)
ddxh = tf.derivative(dxh,f[:-1])

dxv = tf.derivative(xv,f[:-1])
ddxv = tf.derivative(dxv,f[:-1])

xh_in = tf.internal(xh,dxh,ddxh,a_h,wo_h,f)
xv_in = tf.external(xv,dxv,ddxv,a_v,wo_v)

xh_in = ajustement(xh_in)

```

```

xv_in = ajustement(xv_in)

# Génération des figures de sortie

data_in = np.zeros((len(xh_in),3))
data_in[:,1] = xh_in
data_in[:,0] = xv_in
data_in[:,2] = 1

file_in = input("Veuillez entrer le nom du fichier de sortie .csv : "
    " ").strip() + ".csv"
np.savetxt(file_in, data_in, fmt="% .4f", delimiter=",")
print("Le fichier a bien été enregistré.")

# Renvoi visuel des points de sortie

print("Voici un aperçu visuel des points de sortie et d'entrée souhaités :")

plt.subplot(3,1,1)
plt.plot(xh,xv,label="signal de sortie")
plt.axis('equal')
plt.subplot(3,1,2)
plt.plot(xh_in,label="signal d'entrée x")
plt.subplot(3,1,3)
plt.plot(xv_in,label="signal d'entrée y")
plt.show()

```

Bibliographie

1. Blackpenredpen *Sum of two periodic functions, is it still periodic ? Why or why not ?*
<https://www.youtube.com/watch?v=2K7_qyH8h5Y> Consulté le 12.02.2021
2. CanalBlog *Dessinons avec des (épi)ⁿ – cycloïdes*
<http://eljjdx.canalblog.com/archives/2018/02/15/36145181.html> 23.02.2021
3. COYIER Chris SVG, la syntaxe Path
<https://la-cascade.io/svg-la-syntaxe-path/> Consulté le 27.02.2021
4. DELANDTSHEER Anne Analyse I : Chapitre 6 : Équations différentielles
Syllabus MATH – H – 1002, 2020 – 2021
5. DELCHAMBRE Alain Mécanique rationnelle I : recueil de slides
Syllabus MECA – H – 100, 4^{ème} édition : 2020-2021
6. DERAEMAEKER Arnaud Vibrations - SDOF systems
<https://www.youtube.com/watch?v=7EYHWLXcBgcfeature=youtu.be> Consulté le 13.12.2020
7. EL jj Deux (deux ?) minutes pour l'éléphant de Fermi Neumann
<https://www.youtube.com/watch?v=uazPP0ny3XQ> Consulté le 03.03.2021
8. Femto-physique Méthodes de Runge-Kutta
<https://femto-physique.fr/omp/runge-kutta.php> Consulté le 28.02.2021
9. GARNIER Bernard Vibrations des structures industrielles - Outils de modélisation, de prédition et d'interprétation
<https://www-techniques-ingénieur-fr.ezproxy.ulb.ac.be/base-documentaire/environnement-securite-th5/vibrations-en-milieu-industriel-mesures-surveillance-et-controle-42424210/vibrations-des-structures-industrielles-r6191/> Consulté le 26.10.2020

10. *Geogebra* Deriving the Hypocycloid Equations

<<https://www.geogebra.org/m/f3Xbunz3>> Consulté le 01.03.2021

11. *Haelterman Marc* Physique générale II : Chapitre 5 : Oscillations et ondes

Syllabus PHYS-H-100/PHYS-H-101, 4ème édition : 2019-2020

12. *Inkscape* Dessiner en toute liberté

<<https://inkscape.org/fr/release/inkscape-0.48/>> Consulté le 02.03.2021

13. *Jupyter*

<<https://jupyter.org/>> Consulté le 09.03.2021

14. *Scalable Vector Graphics* Chapter 9 : Paths

<<https://www.w3.org/TR/SVG/paths.html>> Consulté le 01.03.2021

15. *Saint-Cirgue Guillaume* Tutoriel python scipy

<<https://machinelearnia.com/tutoriel-python-scipy/>> Consulté le 15/02/21

16. *SimplonLab Robot Tracteur*

<<https://fabmanager.simplon.co/!/projects/robot-traceur-a-partir-d-un-dessin-svg-inskape>> Consulté le 22/02/21

17. *Schmitz Tony - Smith Scott* Mecanical Vibrations : modeling and measurement

<<https://link.springer.com/10.1007/978-1-4614-0460-6>> Consulté le 26.01.2021

18. *Tracker* Video Analysis and Modeling Tool

<<https://physlets.org/tracker>> Consulté le 22/02/21

19. *Numpy*

<<https://numpy.org/>> Consulté le 05.03.2021 Consulté le 20/02/21