Ecole Supérieure de Mécanique de Marseille

Troisième Année

Spécialité

Acoustique et Vibrations Industrielles

Détermination Expérimentale des Paramètres Mécaniques de l'Anche Double du Hautbois

Matthias Coulon

Travail encadré par Christophe Vergez
Laboratoire de Mécanique et d'Acoustique, Marseille,

et par André Almeida,
Institut de Recherche et Coordination Acoustique/Musique

Effectué à l'Ircam, Centre Georges Pompidou
1, place Igor Stravinsky, 75004 Paris,
du 08 mars au 22 aout 2004.

Table des matières

Avant-propos

Nombreux semblent être les scientifiques qui ont cherché à rapprocher la musique et la physique, cherchant à comprendre comment le savoir-faire des facteurs d'instruments, résultat de plusieurs siècles d'observations empiriques a pu engendrer des systèmes acoustiques aussi parfaits que mystérieux.

N'étant pas musicien, je n'étais pas particulièrement sensible à cette problématique... je dois même reconnaître que je n'avais jamais entendu parler de l'IRCAM avant ma recherche de stage. Je tiens donc à remercier l'ensemble des personnes que j'ai eu l'occasion de côtoyer durant les cinq mois passés à l'Institut, pour leur patience et leur ouverture d'esprit. Si je ne suis pas plus musicien qu'avant, leur passion communicative me fait comprendre pourquoi tant de scientifiques travaillent sur ce sujet.

Je tients tout particulièrement à remercier M. Christophe Vergez, tuteur du stage pour son accompagnement, son écoute, sa rigueur mais aussi sa bonne humeur. Mes ambitions professionnelles n'étant pas tournées vers la Recherche, il a eu le respect d'orienter le stage selon mes envies, afin qu'il constitue un réel stage ingénieur.

Je remercie aussi M. André Almeida, pour sa patience et sa disponibilité.

Enfin, je tiens à saluer l'ensemble des personnes que j'ai pu côtoyer au sein de l'équipe Acoustique Instrumentale de l'IRCAM.

Introduction

L'Institut de Recherche et Coordination Acoustique/Musique rassemble scientifiques et musiciens dans le but d'explorer les possibilités créatrices que la Science et en particulier l'Informatique peut offrir aux compositeurs contemporains à travers la synthèse sonore.
L'ambition du département Acoustique Musicale de l'Institut est d'étudier le fonctionnement des instruments de musique de manière expérimentale et théorique afin d'aboutir à une modélisation, ultime étape avant la synthèse sonore.

Le stage s'effectuera en parallèle du travail de thèse de M. André Almeida, doctorant à l'université de Paris 6. Cette thèse porte sur "La Modélisation Physique des Instruments à Anche Double".

Le stage sera encadré par MM. Cristophe Vergez du Laboratoire de Mécanique et d'Acoustique de Marseille et André Almeida. De manière générale, M. Vergez assurera le rôle classique de tuteur, il sera mon référent et décidera des orientations du stage, ce qui ne l'empéchera pas de constituer une aide précieuse face à divers problèmes rencontrés. M. Almeida aura quand à lui, un rôle d'accompagnement, que ce soit lors des expérimentations ou des phases de programmation.

Il est difficile de résumer en une vingtaine de pages l'ensemble du travail effectué lors des cinq mois de stage. Ainsi, après une définition théorique du cadre de l'étude dont le but est d'introduire au mieux la présentation du sujet de stage, le rapport ne présentera que les différents sujets de travail abordés ainsi que les méthodes de travail adptées, pour enfin conclure sur les différents apports professionnels et personnels qui résultent de ce stage. L'ensemble du travail effectué, les problèmes à résoudre, les solutions proposées et les techniques mises en oeuvre seront détaillés dans des annexes qui viendront clore le rapport. Je vous invite bien évidemment à les consulter afin d'améliorer la compréhension du rapport.

Chapitre 1

Sujet du stage

Avant d'énoncer le sujet du stage effectué au sein du Département Acoustique Instrumentale de l'IRCAM, il convient de dessiner le contexte scientifique dans lequel il se situe, et ce pour toute personne qui comme moi, n'est pas forcément familière des instruments à anche.

1.1 Cadre de l'étude

Le présent paragraphe ne constitue pas un bilan de l'état actuel de la recherche scientifique concernant la compréhension des instruments à anche. Son but est en fait de poser quelques bases théoriques nécessaires à la compréhension du rapport et de familiariser les non-musiciens à l'acoustique des instruments à vent.

1.1.1 Présentation des instruments

Les instruments à anche font partie de la famille des bois dont les plus familiers sont le basson, le hautbois, la clarinette, le cor anglais et le saxophone. Nous nous focaliserons en particulier sur deux d'entre eux. Le haut bois, bien évidemment puisqu'il constitue notre sujet d'étude et la clarinette qui, même si sa structure diffère du hautbois, a l'avantage de constituer un modèle bien connu, tant au niveau des phénomènes physiques dont elle est le siège que des modèles mathématiques efficaces qui la modélisent. La figure 1.1 présente une vue générale de ces deux instruments.

1.1.2 Principe de fonctionnement

On peut décomposer un instrument à anche en trois éléments essentiels : l'anche (l'excitateur), le tube (le résonateur) et les trous latéraux. L'air in-

FIG. 1.1 – Images d'un hautbois (image de gauche) et d'une clarinette (image de droite).

soufflé dans l'instrument à travers l'anche produit des vibrations de la colonne d'air contenue dans le tube qui va être quant à elle, en grande partie responsable du son émis.
La fréquence de vibration de l'air est déterminée en premier lieu par les dimensions du tube. Ces dimensions sont modifiées par l'ouverture et la fermeture des trous, ce qui va permettre à un instrument de jouer plusieurs notes.

Que ce soit pour la clarinette ou pour le hautbois, l'anche joue un rôle essentiel dans la production des vibrations. Il convient donc de souligner son importance.
L'anche est composée d'une lame unique de roseau, montée sur une embouchure (cas de la clarinette, voire figure 1.2), ou de deux lames (cas du hautbois, voire figure 1.3).

La vibration de l'anche ouvre et ferme alternativement une fine fente (située soit entre l'anche et l'embouchure, soit entre les deux lames) par laquelle l'air est insufflé dans le tube.

L'anche joue le rôle de soupape qui alimente en énergie vibratoire l'air contenu dans le tube. Pour cela, elle transforme le flux d'air permanent expiré par l'instrumentiste en séries de bouffées se succédant à une fréquence imposée en grande partie par le tube.

De plus, les caractéristiques mécaniques de l'anche influent également, mais dans une moindre mesure la fréquence d'auto-oscillation.

FIG. 1.2 – Anche simple de clarinette (image de gauche) et embouchure sur laquelle elle est fixée (image de droite).

FIG. 1.3 – Vues générales d'une anche double de hautbois.

1.1.3 Le travail de thèse d'André Almeida

Il est possible de dégager une description commune à tous les instruments à anche de leur fonctionnement général : c'est à dire s'appuyant sur le couplage entre un excitateur, l'anche, et un résonateur, le tube.
Cependant, force est de constater qu'il est délicat d'établir un modèle commun à tous. En effet, s'il existe un modèle physique tout à fait performant et accepté de tous concernant la clarinette, cela est autrement plus délicat concernant les instruments à anche double tels que le hautbois. C'est en partie à cette problématique que le travail de thèse d'André Almeida cherche à apporter des réponses.

Ainsi, André Almeida a réalisé un modèle physique de hautbois dont l'annexe A du rapport donne une description simplifiée. Le stage proposé par l'IRCAM constitue une étude complémentaire de ce travail.

1.2 Sujet du stage

Le sujet du stage se superpose au travail de thèse de M. André Almeida et a pour but de l'aider dans ses recherches sur un point particulier de son étude : la mise en place d'un modèle physique performant de hautbois.

Il s'agit de **caractériser expérimentalement les paramètres mécaniques de l'anche double en vue d'affiner le modèle physique.**

En effet, son modèle dépend de plusieurs paramètres physiques tels que la raideur de l'anche ou bien sa position au repos. Il convient de déterminer expérimentalement ces paramètres si l'on veut confronter par la suite les résultats de simulations basées sur le modèle avec des observations expérimentales du comportement de l'anche. De plus, la détermination de ces paramètres est primordiale pour la synthèse sonore.

L'aspect expérimental du stage étant primordial, il est difficile de poser des objectifs précis, les problèmes apparaissant au fur et à mesure que l'étude avance. On peut cependant remarquer que la caractérisation de nombreux paramètres repose sur une étude conjointe des variations de pression du système et des déformations mécaniques de l'anche. Ainsi, nous verrons au chapitre 2 que les deux premiers objectifs du stage concerneront l'interprétation de données de capteurs de pression et le traitement d'images sous Matlab, étapes nécessaires avant de pouvoir caractériser les paramètres de l'anche.

Chapitre 2

Déroulement du stage

2.1 Introduction

Comme nous l'avons vu au chapitre précédent, le stage proposé par l'IRCAM repose sur la caractérisation des paramètres mécaniques de l'anche double en vue d'affiner le modèle physique.

Cependant, il convient de préciser qu'il est délicat de définir des objectifs sur une durée de cinq mois, tant une étude expérimentale peut engendrer des difficultés imprévisibles.

En accord avec le tuteur du stage, M. Christophe Vergez, nous avons convenu avec André Almeida de fonctionner selon la méthode suivante durant les cinq mois que dureront le stage :

1. Familiarisation avec un problème particulier de l'étude.
2. Elaboration d'une solution à partir de recherches personnelles, mais aussi à l'aide d'indications et pistes de recherche distillées par l'encadrement du stage.
3. Mise en place de la solution.
4. Test de la validité de la solution.

Ainsi, après une période de recherche bibliographique afin de me familiariser avec le fonctionnement physique des instruments à anche, le stage peut se diviser en plusieurs objectifs à moyens termes. Nous présenterons chacun d'entre eux en respectant la chronologie des cinq mois de stage, en précisant le contexte dans lequel s'inscrit le problème traité et en détaillant la méthode de travail adoptée.

On remarquera que le rapport de stage est accompagné d'annexes qui viennent compléter la présentation du travail effectué. En effet, le rapport

privilégiera la présentation des résultats obtenus, ainsi que la méthodologie choisie.

Comme nous l'anoncions lors de la présentation du sujet de stage, la caractérisation des paramètres mécaniques de l'anche ne peut se faire qu'après la mise en place d'outils d'analyse des données. Ainsi, les deux premiers travaux réalisés concernent la calibration des capteurs de pression et le traitement des images .

2.2 Calibration des capteurs de pression

Un capteur de pression génère un signal de sortie qui va indiquer la pression effectivement mesurée par le capteur. Comme tout appareil de mesure, une calibration est nécessaire si l'on veut passer du signal émis à une donnée de pression.

Quelques jours avant mon arrivée, l'équipe acoustique instrumentale avait reçu trois nouveaux capteurs de la marque ENTRAN susceptibles de mesurer des pressions statiques et dynamiques, la mesure se faisant à partir d'une cellule piezzo. Ils présentent une membrane souple d'un diamètre d'environ 2 mm ; ses faibles dimensions permettant de les insérer dans des instruments à vents.

FIG. 2.1 – Capteurs de pression ENTRAN, modèle EPE, du type de ceux utlisés par l'équipe Acoustique Instrumentale de l'IRCAM.

2.2.1 L'objectif

Il m'a été demandé d'assurer la calibration de ces nouveaux capteurs (c'est à dire de déterminer une fonction qui donne la relation reliant le signal

émis et la pression mesurée) et d'étudier la variation de cette calibration lorsquel'on fait varier le temps durant lequel le capteur est sous tension en amont de la mesure. En effet, André Almeida avait supposé qu'il existait un laps de temps durant lequel le capteur, pas suffisament "chaud", voyait sa sensibilité aux conditions hygrométriques varier.

Cette étude n'avait pas d'autres buts que de me familiariser avec le matériel expérimental du laboratoire et leur maniement (bouche artificielle, manomètre, carte d'acquisition sous LabView, etc...) et de confirmer à André Almeida qu'il était bien préférable de laisser un capteur de pression plusieurs dizaines de minutes avant de l'utiliser pour des mesures.

2.2.2 L'intérêt

Ce dernier est évident tant l'utilisation de capteurs de pression est fréquente en acoustique musicale.

2.2.3 La méthodologie

A l'aide d'un manomètre qui servirait de référence, on mesure plusieurs couples (pression mesurée, tension en sortie du capteur), et ce à différents instants après que le capteur ait été mis sous tension.

2.2.4 Les résultats

Il est apparu que la réponse en tension du capteur ne varie pas de manière significative selon que l'on est proche ou éloigné de l'instant où le capteur fut mis sous tension.

Par contre, on a constaté lors de la calibration simultanée de deux capteurs que leur réponse en tension n'était pas tout à fait affine. En effet, il semblerait que pour de fortes pressions (de l'ordre de 40 kPa), la réponse s'éloigne du modèle linéaire.

Une fois la remarque soumise à André Almeida et Christophe Vergez, je fus chargé d'établir le profil réel de la réponse de chacun des capteurs et d'élaborer un programme sous Matlab d'étalonnage qui permettrait d'accéder aux valeurs de pression effectivement mesurées. L'annexe B du rapport présente ce travail dont on peut voire un exemple de résultat figure 2.2.

2.2.5 Les difficultés rencontrées

La rédaction du programme n'a pas présenté de réelles difficultés tant la majeure partie des fonctions utilisées avaient été abordées lors des différentes

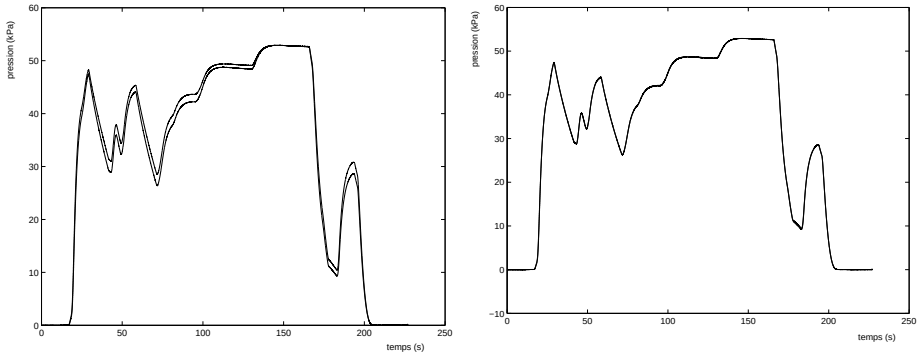

FIG. 2.2 – Comparaison de la calibration de deux capteurs différents pour une même mesure de pression : pour une caractéristique linéaire (à gauche), les valeurs mesurées en pression diffèrent d'un capteur à l'autre alors que pour une caractéristique interpôlée à l'ordre 2 (à droite), ces valeurs concordent.

séances de Travaux Pratiques suivies à l'ESM2. Toute la difficulté résidait dans l'élaboration de la méthode de calcul.

Cependant, il convient de préciser que Christophe Vergez m'a apporté une aide précieuse concernant l'utilisation du logiciel Matlab, notamment à propos des différents moyens d'interpolation.

Remarque : le code sous Matlab de calibration a servi par la suite de support à l'utilisation des interfaces graphiques que Matlab met à disposition de l'utilisateur.
Ainsi, à l'aide d'un manuel Matlab, il a été possible de se familiariser avec cette utilisation particulière du logiciel. La figure 2.3 présente l'interface réalisée. L'annexe B présente quant à elle l'ensemble de sa réalisation et détaille son utilisation.

2.3 Détermination des dimensions de l'anche

Le paragraphe ici présent traite de ce qui a constitué la majeure partie du travail effectué durant le stage. Il s'appuie sur la connaissance des possibilités que Matlab offre en matière de traitement d'images, permettant ainsi de déterminer la forme de l'ouverture de l'anche à chaque instant à partir d'images filmées par une caméra. L'annexe C présente ces possibilités ainsi que le résultat de ce travail. Nous nous contenterons ici de détailler l'intérêt de ce projet ainsi que la méthode de travail adoptée.

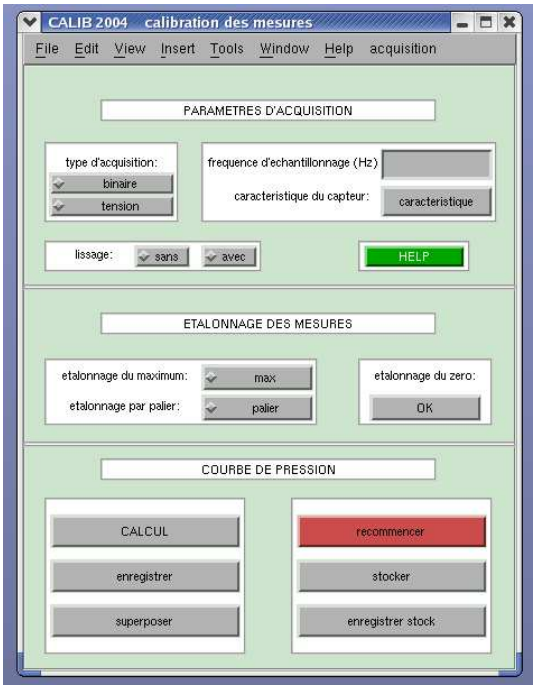

FIG. 2.3 – Interface graphique principale du programme de calibration

2.3.1 L'intérêt

La connaissance de l'évolution au court du temps de la forme de l'anche par traitement d'images présente de multiples intérêts, que ce soit pour l'anche double (aire d'ouverture et largeur) ou l'anche simple (distance entre la lame et l'embouchure). Voici une liste non exhaustive de quelques applications :

1. Pour l'anche double, la connaissance de l'aire d'ouverture permet de déterminer le comportement visco-élastique de l'anche ainsi que sa constante de raideur, comme nous le verrons par la suite. De plus, son aire d'ouverture au repos est un paramètre important de tout modèle physique qu'il est difficile d'estimer par des mesures de part la géométrie particulière de l'ouverture.
2. Pour l'anche simple, du type clarinette, la distance entre la lame et l'embouchure traduit le pincement des lèvres de l'instrumentiste sur le roseau.

2.3.2 L'objectif

L'annexe C présente ce qui a été rendu à l'IRCAM en plus du programme de traitement d'image. Cependant, il convient de détailler de quelle manière ce travail fut réalisé. En effet, l'objectif premier n'était pas initialement de réaliser une interface graphique de traitement d'image mais uniquement une méthode de choix de seuil de binarisation. L'idée était d'utiliser la connaissance des interfaces graphiques sous Matlab acquise lors du projet de calibration des capteurs de pression. Il fallait réaliser un code qui présenterait une barre de défilement faisant varier le seuil de binarisation d'une image et présentant le résultat à l'écran.

Au fur et à mesure que le code avançait, le projet devenait de plus en plus ambitieux pour devenir une interface permettant de visualiser le résultat de différents combinaisons de fonctions de traitement d'images telles que la binarisation, l'inversion de binarité, mais aussi les ouvertures morphologiques ou bien encore les harmonisations de contraste. Dès le départ, il était convenu que le programme devrait s'appliquer à des images seules comme à des séquences d'images.

En dernière étape, nous avons décidé en accord avec l'encadrement du stage d'introduire le code de calcul de détermination des dimensions de l'ouverture, que ce soit sur une image isolée ou sur une séquence d'image. Il convient de préciser que Christophe Vergez et André Almeida avait déjà élaboré une méthode de calcul que j'ai pu améliorer grâce à leur aide, notamment concernant la distinction entre une anche ouverte et une anche quasi-fermée. Bien évidemment, de nombreux problèmes à résoudre sont apparus au fur et à mesure que le code prenait forme tels que la nécessité d'introduire un coefficient de correction sur les dimensions du pixel ou bien concernant l'estimation de la longueur et de la largeur de l'ouverure.

Le traitement en temps réel constitue la dernière étape du projet. Pas vraiment envisagée initialement, elle est apparue comme une possibilité qu'il serait intéressant d'exploiter de par son intérêt lors de certaines manipulations.
La figure 2.4 présente l'interface graphique de traitement d'image, résultat de ce travail. Ses possibilités et son utilisation sont détaillés en annexe C du rapport.

2.3.3 La méthode de travail

J'ai bénéficié du soutient très précieux d'André Almeida et de Christophe Vergez durant toute la durée du projet, notamment lorsqu'il fut question de se familiariser avec le traitement d'images sous Matlab. De plus, il est possible

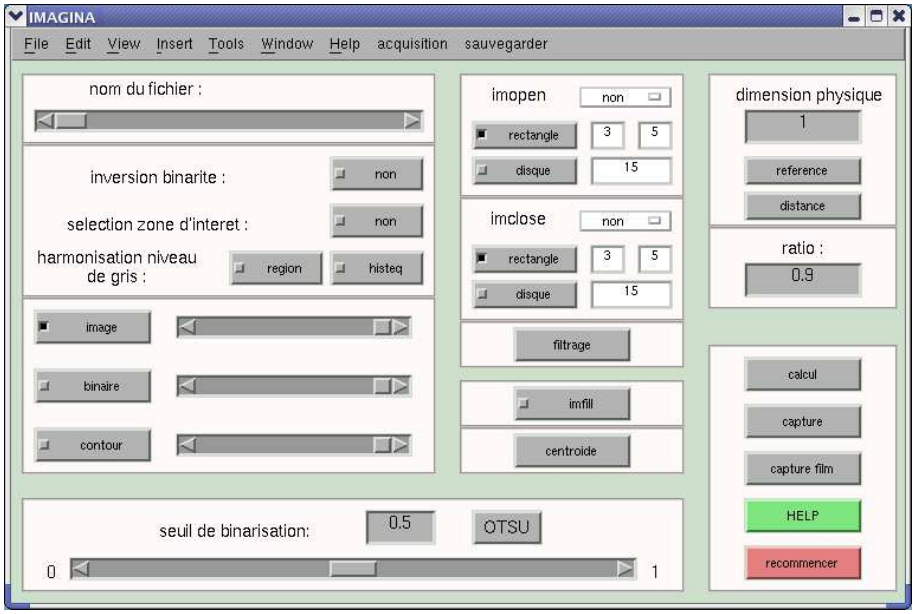

FIG. 2.4 – Interface du programme de traitement d'image.

de trouver de nombreuses documentations en ligne traitant du sujet.

Ainsi, nous avons fonctionné par retouches, rajoutant systématiquement de nouvelles applications à l'interface, nous réunissant régulièrement afin de définir dans quelle voie il serait intéressant de continuer, le projet devenant de plus en plus ambitieux.

2.4 Problème de visco-élasticité

Ce projet suit directement la réalisation de l'interface de traitement d'images puisqu'il repose sur une étude de l'évolution de l'aire d'ouverture d'une anche double. Quelques concepts de traitement d'images seront mentionnés dans l'annexe D du rapport présentant le projet. Ainsi nous vous invitons à consulter l'annexe C au préalable.

Ce projet a pour but d'améliorer la connaissance du comportement mécanique de l'anche double dans l'optique d'affiner le modèle physique d'André Almeida. On cherche à déterminer les constantes de temps permettant de caractériser le caractère visco-élastique d'une anche double.

2.4.1 L'intérêt

L'intérêt de la connaissance du comportement visco-élastique de l'anche double est essentiellement expérimental. Pour le comprendre, il est nécessaire de se rappeller que l'étude expérimentale d'André Almeida est en régime quasi-statique, ce qui implique que le temps n'est pas pris en compte. Or, le comportement visco-élastique de l'anche implique que pour des conditions expérimentale constantes, les dimensions de l'anche vont varier dans le temps. Ainsi, une expérience supposée identique d'un point de vue quasi-statique ne sera pas reproductible selon qu'on la réalise en 1, 5 ou 20 minutes.

De plus, une modélisation du comportement visco-élastique de l'anche double permettrait de perfectionner le modèle physique du hautbois d'André Almeida, cet aspect n'étant pas pris en compte initialement (cf annexe A).

2.4.2 Les résultats

Les résultats de l'étude sont présentés en annexe D du rapport. On remarquera que de nombreuses questions restent en suspens et que le rapport ne donne pas réellement une description valable du comportement visco-élastique de l'anche. La raison principale à cela est bien évidemment un manque de temps, mon stage se terminant au court de l'étude.

2.4.3 Les difficultés rencontrées

Les difficultés rencontrées sont avant tout expérimentales. En effet, il est relativement délicat de régler un éclairage de façon à permettre une binarisation de l'image qui permette la détermination des dimensions de l'anche. A ce titre, l'interface de traitement d'image réalisée précédemment fut d'une aide précieuse, puisqu'elle permet lors de l'installation du protocole de visualiser le résultat de la binarisation de l'image, et d'ajuster l'éclairage du montage en conséquence.

Conclusion

La conclusion ne porte pas sur le contenu scientifique du stage ou l'interprétation théorique des résultats expérimentaux puisque ce stage n'était pas un stage de Recherche, mais bien un stage Ingénieur. Cependant, j'espère que le travail réalisé durant ces cinq mois sera utile à MM. André Almeida et Christophe Vergez et qu'il les aidera à mener à bien leurs recherches par la suite.

Ainsi, il convient de rappeller que mon rôle se situait en amont de toute interprétation. En effet, il m'a été demandé d'élaborer des outils sous Matlab qui répondraient à des problèmes précis, de mise en place expérimentale ou de post-traitement des résultats, et ce afin d'aider au mieux les recherches d'André Almeida.
Bien évidemment, j'ai participé à des expérimentations. Mais ce fut essentiellement dans le but de tester les outils élaborés afin de révéler divers problèmes qu'il était difficile d'anticiper.

Du point de vue de ma formation, ce stage m'a apporté une meilleure connaissance des techniques expérimentales, du maniement du matériel de mesure et bien évidemment du logiciel Matlab. C'est surtout à propos de ce dernier point que je suis le plus enthousiaste. En effet, en plus de me perfectionner dans l'utilisation des fonctions de calcul matriciel, complétant ainsi la formation reçue à l'ESM2, j'ai pu découvrir les possibilités offertes par le logiciel en matière de traitement d'images et d'interfaces graphiques.

D'un point de vue plus personnel, ce stage fut l'occasion d'acquérir une réelle autonomie dans le travail. En effet, s'il n'est pas toujours évident de se greffer au travail de quelqu'un sans pour autant avoir un rôle d'assistant, je me dois de remercier Christophe Vergez qui m'a laissé toute liberté sur le programme de traitement d'image, me permettant de m'approprier le projet et d'en gérer l'évolution.

Je dois reconnaître n'avoir jamais réellement envisagé d'évoluer professionnellement dans le monde de la Recherche Fondamentale. Cependant, fort

de mon expérience acquise au sein de l'IRCAM, ma recherche d'emploi se dirigera certainement dans les premiers temps vers des départements R&D. En effet, j'ai particulierment apprécié la rigueur inhérente à cette activité, la recherche permanente de solutions aux différents problèmes qui apparaissent chaque jour.

Annexe A

Modèle physique des instruments à anche

Nous nous appuierons dans le présent paragraphe sur un article rédigé par MM. André Almeida, Christophe Vergez, René Caussé et Xavier Rodet : *Toward a Simple Physical Model of Double-Reed Musical Instruments : Influence of Aero-Dynamical Losses in the Embouchure on the Coupling between the Reed and the Bore of the Resonator.*
Publié en 2003 dans la revue *Acustica*.

Cette article a pour but de présenter un modèle physique valable pour les instruments à anche double, et ce en régime dynamique.

A.1 Décomposition schématique d'un instrument à anche

Les instruments à anche peuvent être décomposés schématiquement de la même manière, que l'anche soit simple ou double. Nous verrons par la suite que la géométrie de l'embouchure constitue un paramètre essentiel du modèle, ainsi, on prend soin de la faire figurer sur le schéma géneral d'un instrument à anche (cf figure A.1).

On décompose ainsi l'instrument en quatre sous-systèmes :

1. **La bouche** : La pression p_m imposée par l'instrumentiste y est considérée comme statique.
2. **L'anche** : On l'associera à un oscillateur à un degré de liberté de position z, de vitesse $\dot{z}$ et d'accélération $\ddot{z}$. L'anche est le siège de la formation d'un jet d'air. On pose p_j et v_j respectivement la pression et la vitesse de ce dernier.

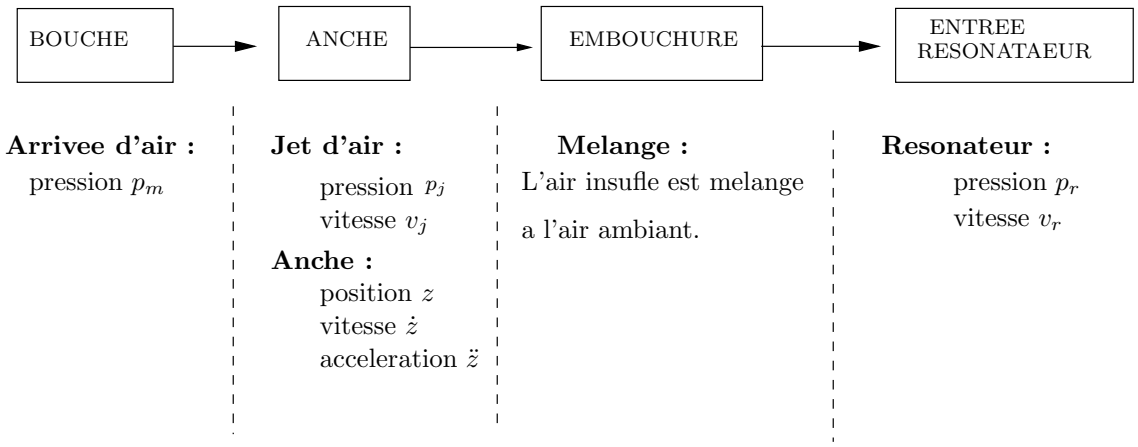

FIG. A.1 – Schéma des principales parties d'un instrument à anche et variables associées.

3. **L'embouchure** : Le jet d'air s'y mélange avec l'air ambiant de l'instrument. On pose p_r et v_r la pression et la vitesse du fluide à la sortie de l'embouchure, et donc à l'entrée du résonateur.
4. **Le résonateur** : cylindrique (clarinette) ou conique (hautbois), il va donner une relation entre p_r et v_r.

Le modèle d'André Almeida repose sur une description du fonctionnement de l'instrument en régime quasi-statique. Les équations ainsi déterminées seront ensuite étendues au régime dynamique.

Le modèle physique de clarinette constitut un bon point de départ avant d'aborder le modèle de hautbois. Ainsi, afin de déterminer un modèle quasi-statique des instruments à anche double, nous étudierons un modèle quasi-statique de clarinette. Nous nous pencherons ensuite sur les différences majeures qu'un modèle de hautbois doit impliquer.

A.2 Modèle physique quasi-statique de clarinette

En première approximation, la clarinette est constituée d'un système excitateur (anche + bec) et d'un résonateur (un tube de perce cylindrique muni d'un pavillon et de trous latéraux, dont les effets seront négligés ici).

A.2.1 Les équations du modèle

Elles reposent sur différentes hypothèses qui amèneront chacune certaines approximations.

Hypothèse 1 : Le jet d'air en sortie de l'anche voit son énergie cinétique totalement dissipée par turbulence. Cette hypothèse repose sur le fait que la surface du résonateur est bien plus grande que l'aire d'ouverture de l'anche simple, ainsi que l'angle d'ouverture de l'embouchure relativement grand qui va favoriser un décollement du jet des bords de l'embouchure (cf figure A.2).

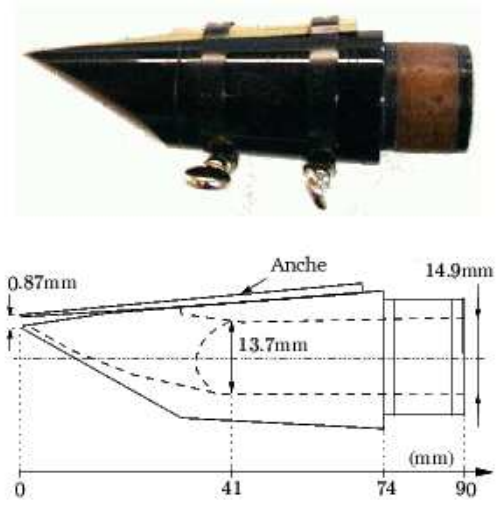

FIG. A.2 – Embouchure d'une clarinette.

Ainsi, on négligera la vitesse de l'écoulement v_r à la sortie de l'embouchure devant la vitesse à son entrée v_j. L'écoulement à travers l'embouchure vérifie donc la relation de Bernoulli simplifiée :

$$\begin{cases} p_m = p_j + \frac{1}{2}\rho v_j^2 \\ p_j = p_r \end{cases} \tag{A.1}$$

où ρ correspond à la densité de l'air.

Hypothèse 2 : L'anche simple de la clarinette est modélisée par un système { masse + ressort + amortissement} à un degré de liberté. L'hypothèse selon laquelle le régime est quasi-statique implique que les effets d'amortissement et d'inertie de l'anche ne seront pas considérés. Ainsi, l'ouverture de l'anche est commandée par la relation :

$$k(z - z_0) = (p_j - p_m) \tag{A.2}$$

où z_0 est la position au repos de l'anche et k sa raideur surfacique.

Hypothèse 3 : L'hypothèse d'incompressibilité de l'écoulement dans l'embouchure implique que l'on peut y considérer le débit (noté q) comme constant (en espace). On peut ainsi exprimer le débit en fonction des dimensions de l'ouverture de l'anche par la relation :

$$q = \alpha S_i v_j \tag{A.3}$$

On prendra soin de préciser les paramètres suivants :

- La surface d'ouverture de l'anche simple S_i qui est égale au produit de la hauteur de l'ouverture z et de la largeur de l'anche l.
- Le coefficient de *vena contracta* noté α. Il traduit l'écart existant entre l'aire d'ouverture de l'anche et la section effective du jet. Fortement dépendant du nombre de Reynolds, α s'obtient de manière semi-empirique. Il peut varier entre 0,61 entre 1.

A.2.2 La caractéristique de l'anche de clarinette

On désigne par **caractéristique de l'anche** la relation qui relie le débit q de l'écoulement et la différence de pression entre la pression dans la bouche imposée p_m et la pression à l'entrée du résonateur p_r ou p_j (cf équation (A.1)).

On obtient une première expression de la caractéristique en associant les équations (A.1) et (A.3) :

$$q = \alpha z l \sqrt{\frac{2}{\rho}(p_m - p_j)} \tag{A.4}$$

Les paramètres α, l, ρ et p_m étant des constantes du système, on s'affranchit de la position z de l'anche simple à l'aide de l'équation (A.2). On obtient ainsi l'expression :

$$q = \alpha l (z_0 - \frac{1}{k}(p_m - p_j)) \sqrt{\frac{2}{\rho}(p_m - p_j)} \tag{A.5}$$

La figure A.3 présente le tracé d'une caractéristique pour une anche de clarinette pour les paramètres : $\alpha = 1$, $z_0 = 1.10^{-3}$, $l = 1.6.10^2$m, $k = 1.6.10^7$Pa m^{-1} et $p_m = 8.10^3$Pa.

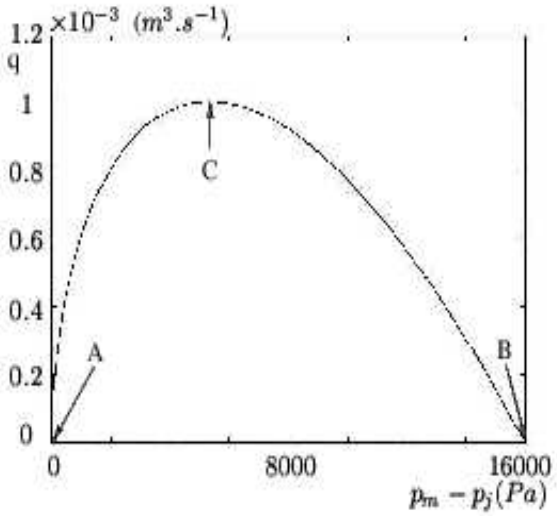

FIG. A.3 – Exemple de caractéristique d'une anche de clarinette.

A.2.3 Interprétations

La caractéristique du couplage donne essentiellement des indications sur le comportement de l'anche de clarinette. L'anche se ferme pour les cas $p_j = p_m$ (point A de la figure A.3) et $p_m - p_j = z_0 k$ (point B de la figure A.3). On notera que lorsque $p_m - p_j \geq z_0 k$, l'anche reste fermée.

De plus, on remarquera que l'anche présente son maximum d'ouverture pour le cas $p_m - p_j = \frac{1}{3} z_0 k$ (point C de la figure A.3).

La caractéristique de l'anche ne permet pas de modéliser l'instrument dans son ensemble. En effet, la caractéristique est la réunion d'une équation traduisant le comportement mécanique de l'anche (équation A.2), et d'une équation traduisant le couplage entre l'anche et le résonateur (équation A.1). Il s'agit donc d'associer à l'équation A.5, une relation entre le débit et la pression à l'entrée du résonateur caractéristique de ce dernier. On utilise l'impédance acoustique du résonateur pour une description fréquentielle. On parlera aussi de réponse impulsionnelle g du tube en temporel :

$$p_j = p_r = g * q \tag{A.6}$$

Les équations (A.5) et (A.6) forment ainsi un système décrivant le fonctionnement de l'instrument en régime quasi-statique.

A.3 Modèle physique quasi-statique de hautbois

L'idée est de s'inspirer du modèle caractérisant le couplage anche + résonateur de la clarinette. Afin de pouvoir faire cette analogie, nous supposerons que :

1. Nous nous plaçons en régime quasi-statique.
2. Le comportement mécanique des deux lames de roseau de l'anche double est symétrique. On pourra assimiler chacune d'entre elles à une lame simple d'anche de clarinette.

A.3.1 Les équations du modèle

L'hypotèse 2 caractérisant le modèle de clarinette semble toujours appropriée au modèle : ainsi, chacune des lames sera considérée comme un oscillateur à un degré de liberté de raideur k. Sa position z vérifiera la relation :

$$k(z - z_0) = (p_j - p_m) \tag{A.7}$$

Nous verrons que la différence principale du modèle de hautbois avec celui de clarinette touche à l'hypothèse 1.

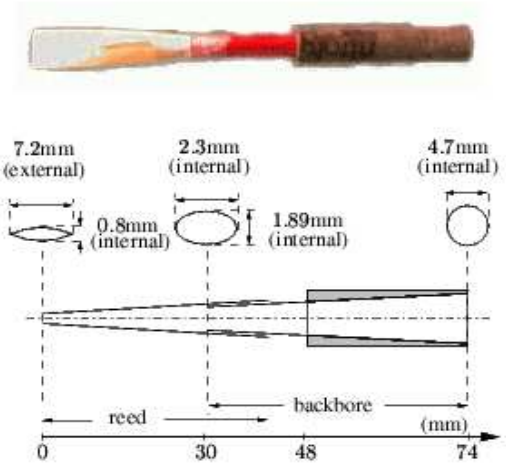

FIG. A.4 – Embouchure de hautbois.

En effet, la géométrie spécifique de l'anche double (cf figure A.4, on constate que la section de l'embouchure croît de manière lente et continue)

rend peu crédible l'hypothèse selon laquelle la totalité de l'énergie cinétique du jet à l'entrée de l'instrument serait dissipée par turbulence à la traversée de l'embouchure.

En tenant compte de la viscosité de l'air, deux comportements sont envisagés par l'article :

Première possibilité, l'ecoulement se rattache aux bords de l'embouchure (cf figure A.5) : on suppose l'existence d'un décollement de l'écoulement des parois des lames dû à la vitesse d'entrée du jet et à la viscosité de l'air, cela créant deux vortex à l'entrée de l'anche double. Un réattachement aux parois a lieu un peu plus en aval de l'anche.

Seconde possibilité, la dissipation par turbulence de l'energie cinétique est partielle (cf figure A.6) : essentiellement dûe à la géométrie de l'embouchure, l'écoulement reprendrait sa direction après un zone de turbulences.

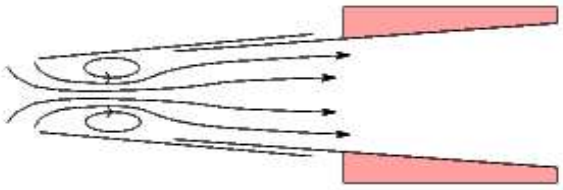

FIG. A.5 – Décollement puis réattachement de l'écoulement.

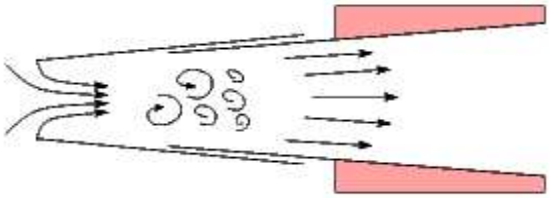

FIG. A.6 – Dissipation partielle par turbulence.

Ainsi, l'hypothèse 1 du modèle de clarinette n'est pas applicable au hautbois. Posons les équations du modèle du hautbois en quasi-statique :

On applique Bernouilli entre l'entrée et la sortie de l'anche double :

$$p_m = p_j + \frac{1}{2}\rho v_j^2 \tag{A.8}$$

On réutilise l'hypothèse 2 du modèle de clarinette selon laquelle le débit au sein de l'embouchure reste uniforme. On exprime ainsi le débit q à la sortie

de l'anche :

$$q = \alpha S_i v_j \tag{A.9}$$

où S_i représente l'aire d'ouverture de l'anche double et α le coefficient de *vena contracta* (cf modèle de clarinette).
On obtient ainsi une expression du débit au sein de l'embouchure en réunissant les équations (A.8) et (A.9) :

$$q = 2zl\gamma\alpha\sqrt{\frac{2}{\rho}(p_m - p_j)} \tag{A.10}$$

où le facteur $2z$ traduit la distance entre les deux lames, l la largeur de chacune et γ la correction dûe au fait que l'ouverture n'est pas rectangulaire.

Il s'agit dorénavant de traduire la dissipation partielle de l'énergie cinétique :

$$p_j = p_r + \frac{1}{2}\rho\Psi\frac{q^2}{S^2} \tag{A.11}$$

Le coefficient Ψ traduit cette disspation partielle. L'obtention de l'équation (A.11) ne sera pas détaillée ici. Elle résulte d'un bilan des effets dissipatifs, de leur expression et d'une approximation. Le terme S représente la section de l'embouchure où l'écoulement devient homogène (après le réattachement sur la figure A.5 ou après la zone turbulente sur la figure A.6). On remarquera que le cas $\Psi = 0$ correspond à une dissipation totale de l'énergie cinétique. On retrouve bien alors la condition $p_j = p_r$ qui caractérise une anche de clarinette.

A.3.2 La caractéristique de l'anche double

Il s'agit d'établir la relation liant le débit q et la différence de pression $p_m - p_r$. On va dans un premier temps établir la relation liant le débit à la différence de pression $p_m - p_j$ à partir des équations (A.7) et (A.10) dans une démarche analogue à l'établissement de la caractéristique de l'anche de clarinette :

$$q = 2l\gamma\alpha(z_0 - \frac{1}{k}(p_m - p_j))\sqrt{\frac{2}{\rho}(p_m - p_j)} \tag{A.12}$$

Il s'agit maintenant d'établir une relation liant les différences de pression $p_m - p_r$ et $p_m - p_j$ afin d'obtenir un système permettent de déterminer la caractéristique de l'anche double.

Nous allons pour cela utiliser la variable v_j (cf équation (A.9)) qui représente la vitesse du jet à l'entrée de l'embouchure. v_j est reliée à la différence de pression $p_m - p_j$ par la relation :

$$v_j = \sqrt{\frac{2}{\rho}(p_m - p_j)} \tag{A.13}$$

Les équations (A.7) et (A.9) permettent de relier le débit à v_j selon l'expression :

$$q = 2l\gamma\alpha(z_0 - \frac{1}{k}(p_m - p_j))\, v_j \tag{A.14}$$

On peut donc poser un système reliant p_m, p_r et p_j à v_j obtenu à partir des équations (A.8) et (A.11)

$$\begin{cases} p_m = p_j + \frac{1}{2}\rho\, v_j^2 \\ p_j = p_r + \frac{1}{2}\rho\Psi\frac{1}{S^2}\Big(2l\gamma\alpha(z_0 - \frac{1}{k}(p_m - p_j))\Big)^2 v_j \end{cases} \tag{A.15}$$

Le système d'équations (A.15) associé à l'équation (A.13) permet d'établir un polynôme de $p_m - p_j$ de degré 3 et ainsi relier les différences de pression $p_m - p_j$ et $p_m - p_r$:

$$(p_m - p_j)^3 - 2z_0k(p_m - p_j)^2 + k^2(z_0^2 + D)(p_m - p_j) - k^2D(p_m - p_r) = 0 \tag{A.16}$$

où $D = \frac{S^2}{4\alpha^2\gamma^2l^2\Psi}$.

Les équations (A.16) et (A.12) permettent de construire une méthode numérique afin d'établir la caractéristique de l'anche double. En effet, pour chaque valeur de $p_m - p_r$, on pourra trouver une ou plusieur valeur de $p_m - p_j$ par la relation (A.16), et donc ainsi accéder au débit par l'équation (A.12).

La figure A.7 présente le tracé de plusieurs caractéristiques obtenues à partir des expressions analytiques déterminées précédemment. Pour le tracé, on garde pour valeur numérique des paramètres : $\alpha = 0,8$, $z_0 = 8.10^{-4}$, $l = 7.10^7$m, $k = 1.6.10^7$Pa m^{-1}, $p_m = 12.10^3$Pa et $\gamma = 1$.

A.3.3 Interprétations

Comme nous l'avons vu au paragraphe A.2.3, le modèle n'est pas complet sans une relation propre au résonateur. Là encore, la réponse impulsionnelle de ce dernier peut assurer ce rôle.

Concernant la caractéristique de l'anche double, nous remarquerons que :

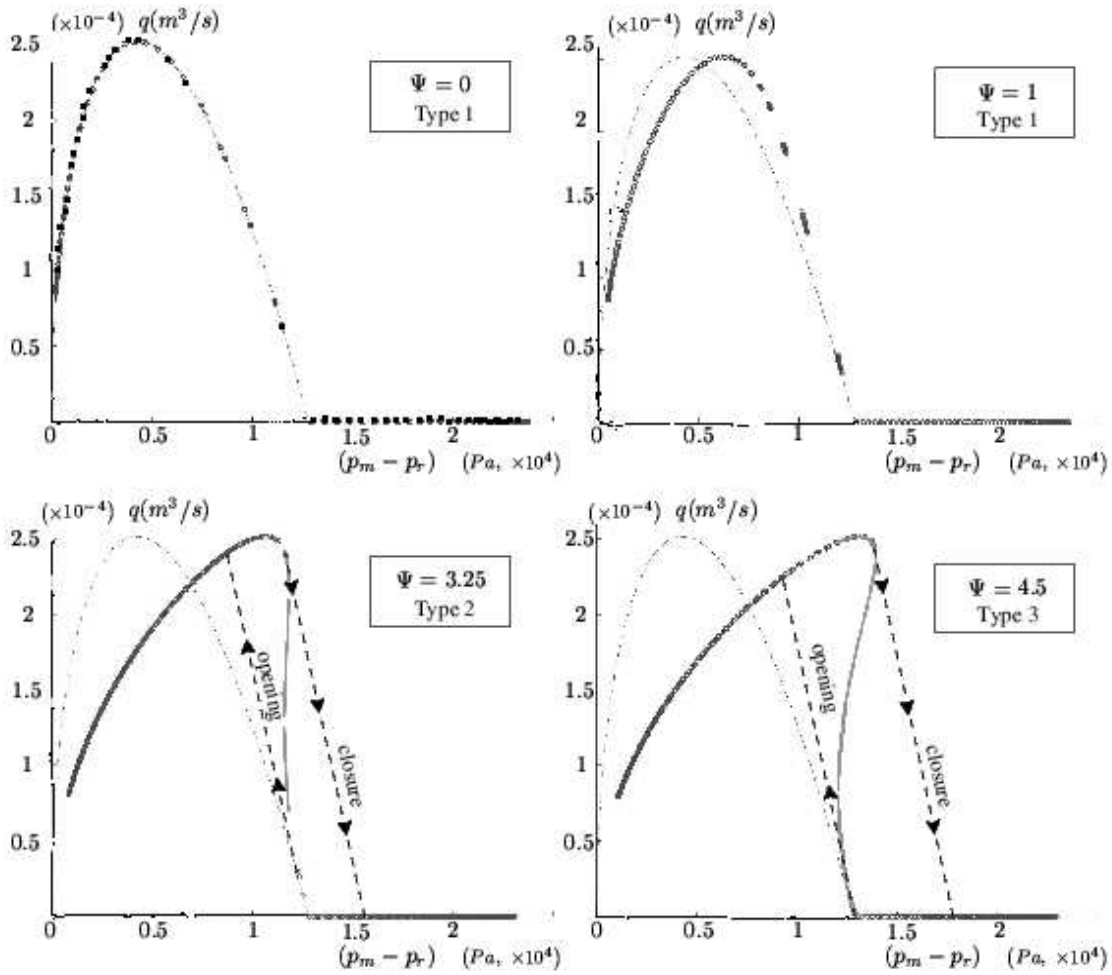

FIG. A.7 – Tracer de caractéristiques d'anche double pour différentes valeurs de Ψ (symbole $\circ$). Comparaison avec les résultats le cas $\Psi = 0$ (trait pointillé).

1. Le cas $\psi = 0$ permet bien de retrouver une caractéristique conforme au modèle de clarinette.
2. Le tracé des cas $\psi = 3,25$ et $\psi = 4,5$ suggère l'existence d'un hystérésis. En effet, il existe une plage de valeur de $p_m - p_r$ où l'étude analytique présente trois valeurs de débit distinctes. On peut dès lors supposer que l'anche pour certaines valeurs de $p_m - p_r$ sera fermée si elle est dans un cycle d'ouverture ou pleinement ouverte si elle est dans un cycle de fermeture. La solution intermédiaire ne serait pas considérée, le passage de l'anche ouverte à fermée (et inversement) étant quasi instantané dès le passage d'une valeur seuil de $p_m - p_r$ (phénomène

vérifiable expérimentalement).

Annexe B

Calibration des capteurs de pression

B.1 Introduction

B.1.1 Position du problème

Il s'agit de calibrer des capteurs de pression afin de déterminer à partir de mesures en tension acquises sous LabView les pressions effectivement mesurées lors de l'acquisition. La carte d'acquisition n'acceptant en entrée que des tensions comprises entre -2.8 et 2.8 volts, il a fallu placer en sortie des capteurs un gain variable qui permet de faire coïncider au mieux la plage de pression attendue avec la bande (-2.8v, 2.8v). On parlera de gain imposé par l'utilisateur g_u, que l'on réglera avant le début de chaque acquisition et que l'on considèrera comme constant durant toute sa durée.
Nous verrons qu'il n'est pas nécessaire de determiner précisément g_u pour obtenir les valeurs de pression effectivement acquises si l'on connaît une valeur de pression du système à un instant donné et la valeur de tension correspondant à cette pression sur l'acquisition. La donnée de ce couple (P, V) est évidemment nécessaire lors de chaque acquisition et ce, pour chacun des capteurs.
Il suffit donc de déterminer avec exactitude pour chacun des capteurs sa réponse en tension à toutes valeurs de pression pour connaître précisément les valeurs de pression mesurées durant l'acquisition. Nous cherchons donc à définir ce que nous appellerons la **caractéristique** de chaque capteur, c'est à dire sa réponse en tension à sa sortie en fonction de la pression qu'il mesure.
Toute la difficulté sera :

- de s'affranchir du gain imposé g_u, ainsi que de la température et de l'humidité. En effet, nous recherchons une caractéristique à laquelle nous

pourrions nous reporter quelles que soient les conditions de l'acquisition.
– de déterminer cette caractéristique qui présente un caractère non linéaire que l'on ne peut pas négliger.

B.1.2 Mise en évidence du caractère non linéaire de la caractéristique

On superpose les acquisitions de deux capteurs différents pour une même variation de pression d'un système. On suposera ce dernier sans fuites (ie même pression mesurée aux mêmes instants par les deux capteurs). Afin de rendre lisible cette superposition, on prend soin de faire coïncider les deux tensions dites de repos (i.e. pour une mesure de pression nulle).

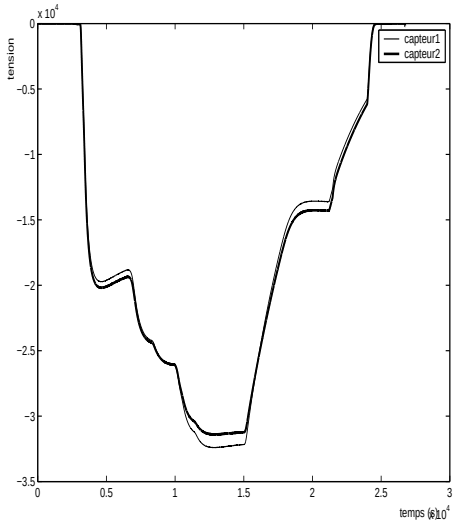

FIG. B.1 – Acquisition recalée en zero ie $V = 0$ pour $P = 0$

La non-superposition des deux courbes peut provenir à la fois du gain imposé à la sortie du capteur g_u qui peut différer selon que l'on observe le capteur 1 ou le capteur 2, mais aussi du fait que les deux capteurs peuvent présenter des caractéristiques différentes. Cependant, le fait que la réponse en tension du capteur 2 (courbe en gras) soit successivement au dessous, puis au dessus de la réponse du capteur 1 implique nécessairement l'existence d'une caractéristique non-linéaire pour au moins l'un des deux capteurs.

B.1.3 Nécessité de la prise en compte du caractère non linéaire de la caractéristique

Nous allons faire ici l'hypothèse que la réponse en tension des capteurs à une excitation en pression est de la forme affine :

$$V = Vo + \lambda \times P, \qquad \lambda > 0 \tag{B.1}$$

A partir des acquisitions présentées en figure B.1, voyons les conséquences d'une telle hypothèse sur les courbes de pression que l'on peut déduire.

On peut déterminer aisément sous Matlab Vo et λ à l'aide respectivement des fonctions `min()` et `max()`, à condition d'avoir préalablement pris soin durant l'acquisition de relever la valeur de la pression minimale (dans l'exemple, une pression nulle) et maximale auquelle le système est soumis durant l'acquisition (on pourra utiliser un manomètre pour mesurer ces dernières). Cette méthode permet ainsi s'affranchir de la connaissance du gain g_u.
On peut visualiser sur la figure B.2 les courbes ainsi obtenues. On remarque que même si nous avons utilisé ici le maximun et le minimum de pression imposés par l'utilisateur que l'on a fait correspondre au maximum et au minimum de la courbe d'acquisition, la connaissance de deux couples (P, V) quelconques situés dans la plage des valeurs acquises seraient aussi efficace. Cependant, la méthode des maximas permet de s'affranchir de la mesure des tensions correspondantes aux deux valeurs de pression relevées et de s'assurer du bon calibrage du capteur pour des pressions nulles.

On remarque en figure B.2 que les deux courbes ne se superposent pas alors qu'elles sont censées représenter les mêmes variations de pression du système. L'impression qu'elles coïncident à leur base et à leur maximum n'est dû qu'au fait que la méthode choisie impose leur valeur minimale (pression nulle) et leur valeur maximale (pression mesurée au manomètre) comme communes.

B.1.4 Conclusion

L'hypothèse selon laquelle la caractéristique de chacun des capteur est une fonction affine induit un décalage en pression du capteur 2 sur le 1 pour les pressions intermédiaires. **Il est donc nécessaire de tenir compte des caractéristiques de chacun des capteurs, dont au moins une des deux est non linéaire.**

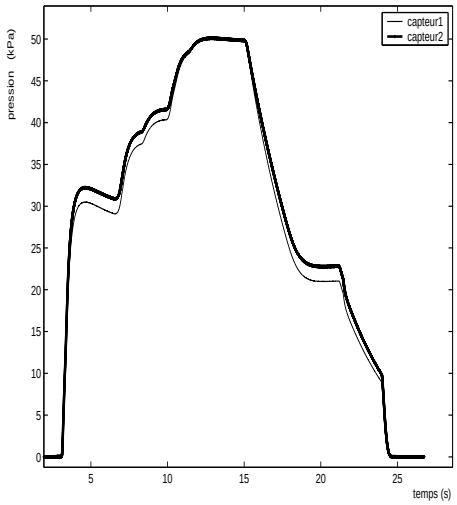

FIG. B.2 – Valeurs de pression en fonction du temps

B.2 Etablissement de la caractéristique sous Matlab

B.2.1 Position du problème

L'établissement de la caractéristique d'un capteur se fait par interpolation à partir de la donnée de N couples de mesures (P, V) où P est la pression effectivement mesurée par le capteur et V la tension acquise correspondante. On peut légitimement se poser la question du choix du gain g_u que l'on va imposer lors de la mesure des N couples de mesure. Par la suite, on note g_{uc} la valeur du gain g_u imposé lors de la mesures des couples.

L'idée sera de trouver une méthode qui s'affranchira de la connaissance de g_{uc} lors de l'utilisation de la caractérisique (cf chapitre B.3.1). On peut donc régler g_{uc} de telle sorte que l'on obtienne un maximum de précision, c'est à dire étaler au mieux la plage des pressions qui nous intéressent sur la bande d'acquisition [0v,2.8v].

A partir des N mesures, on peut interpoler la caractéristique par un polynôme de degré inférieur ou égale à N-1 à l'aide de la fonction `polyfit()` qui va prendre en entrée les couples (P,V) et le degré souhaité (N-1 étant le plus haut degré que l'on peut demander) et qui renvoie les coefficients du polynôme ainsi calculé. On peut au choix demander à matlab d'établir une interpolation de la fonction $P = f(V)$ ou de son inverse $V = g(P) = f^{-1}(P)$. Cependant, dans la mesure où nous désirons déterminer une valeur de pres-

sion à partir d'une donnée de tension acquise, il est préférable d'interpoler la fonction f.

Evidemment, il est nécessaire que chaque valeur de tension corresponde à une unique valeur de pression qui lui est propre sur la caractéristique interpolée, en d'autres termes, cette dernière doit être bijective sur la plage des pressions qui sont susceptibles de nous intéresser.

Un premier moyen simple de s'assurer graphiquement de cette bijectivité est de tracer le polynôme sur la bande de tension [0v,2.8v] et de constater qu'il est bien strictement croissant.
Un autre moyen, plus rigoureux existe. Nous le verrons au paragraphe B.3.2.

B.2.2 Mesure des N couples (P, V)

De la précision des couples de mesure (P, V) va dépendre la précision de la caractéristique, et donc la qualité de la calibration.
Compte-tenu du materiel à notre disposition, deux difficultés se présentent :

1. La difficulté d'obtenir une pression stabilisée dans le système de part la présence de légères fluctuations de débits au niveau de l'arrivée d'air qui est reliée à un compresseur.
2. L'imprécision de la mesure de pression et de tension induite respectivement par le manomètre et le voltmètre.

Voici le mode opératoire choisi afin de réduire au mieux ces imprécisions : On fait une acquisition sous LabView de la tension en sortie du capteur (amplifié selon g_{uc}) pour une pression qui croit selon N paliers (cf figure B.3).

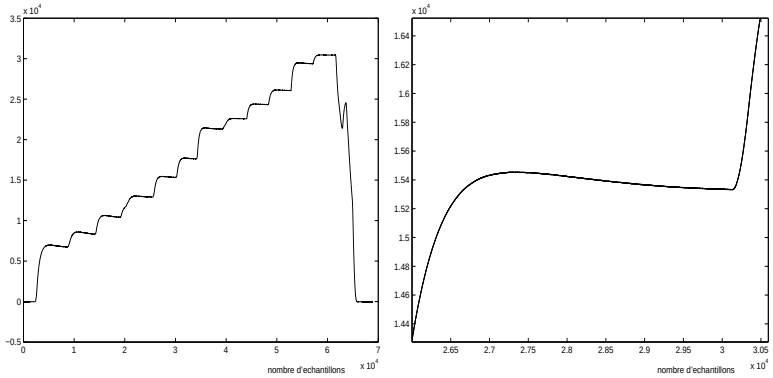

FIG. B.3 – Acquisition pour 13 paliers et zoom du 6^{eme} palier

Un zoom permet de verrifier que c'est paliers ne sont pas constants (hormis le paliers correspondant à une pression nulle).

L'idée est de noter la valeur de pression maximum lue au manomètre pour chaque palier et de l'associer à la valeur maximum de la fenêtre de l'acquisition correspondante (dans le cas particulier du palier correspondant à $P = 0$, on choisira plutôt la valeur moyenne).

Une conversion peut être nécéssaire afin de passer des valeurs acquises à des valeurs de tension, notamment si l'on utilise des fichiers en sortie codés en binaire (c'est à dire des entiers compris entre -2^{15} et 2^{15}). En effet, on rappelle que les tensions en entrée de la carte d'acquisition et les entiers en sortie sont reliés par un coefficient de proportionnalité qu'il faut déterminer.
Pour cela, il suffit de relier la carte d'acquisition à un générateur de tension et de faire une acquisition de la tension délivrée. Pour une meilleur estimation, on peut faire différents paliers successifs de tension (en prenant soin de rester sous la tension de saturation). En faisant correspondre la tension delivrée et la valeur moyenne de l'acquisition qui correspond, et ce pour chaque palier, on peut estimer ce coefficient avec précision (on fait la moyenne sur le nombre de paliers réalisés).
On note qu'une fois ce coefficient estimé, on peut déterminer la tension de saturation de la carte d'acquisition (que l'on sait proche de 2.8v).

Dans notre cas, la carte d'acquisition utilisée présente un coefficient $\beta = 8.612e-05$ et une tension de saturation estimée à 2.852v.

B.2.3 Choix du degré d'interpolation

Si la fonction `polyfit()` permet de calculer un polynôme de degré N-1 qui se raproche au mieux des N couples de mesures, il n'est pas systématiquement conseillé de choisir le plus haut degré possible. En effet, la caractéristique étant quasi affine pour les basses et moyennes pressions considérées, trop augmenter le degré de l'interpolation tend à s'éloigner de ce comportement.

On peut visualiser figure B.4 l'interpolation correspondante aux 13 couples de mesures présentés précedemment de degré 3 et de degré 8 (en gras). Si le résidu moyen est plus faible pour le polynôme de degré 8, on constate que :

1. La caractéristique obtenue ne satisfait pas le critère de subjectivité.
2. Le comportement de l'interpolation de degré 3 se rapproche plus aux pressions de la plage [0kPa,40kPa] d'un comportement linéaire que l'interpolation de degré 8.

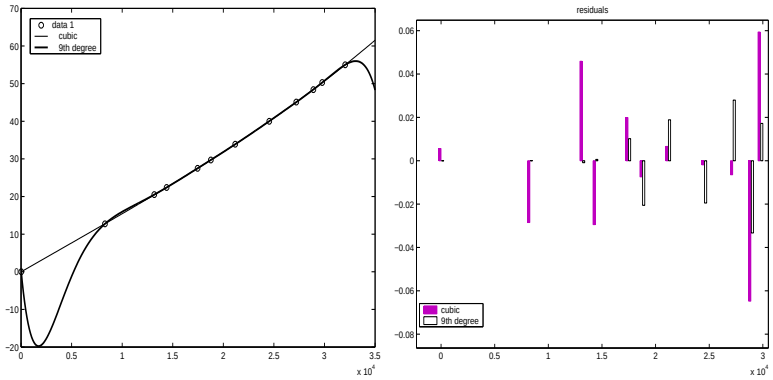

FIG. B.4 – Interpolation degré 3 et 9

3. Le résidu moyen sur la plage de pression [20kPa,40kPa] est plus faible pour l'interpolation de degré 3.

Par la suite, il semblera donc indiqué de choisir un degré d'interpolation approprié aux plages de pression que l'on souhaite mesurer.

B.3 Passer d'une tension mesurée à une pression

Il va s'agir, à partir des tensions acquises durant la mesure, de se reporter à la caractéristique du capteur utilisé (préalablement établie) afin d'en déduire la valeur de pression mesurée correspondante. Sachant que le gain g_u imposé lors de l'acquisition n'est pas forcément identique au gain g_{uc} utilisé lors de la mesure de la caractéristique, on va chercher à ramener les tensions acquises aux tensions équivalentes sur la caractéristique, et ce sans pour autant avoir à estimer g_u ou g_{uc}.

B.3.1 S'affranchir des différences de gain imposé, de l'humidité, de la température

La tension acquise dépend directement du gain imposé par l'utilisateur. Celui-ci va être fixé selon les plages de pression que l'on désire mesurer : le signal en sortie d'un capteur sera donc plus amplifié ($g_u > g_{uc}$) s'il a pour vocation de mesurer de petites valeurs de pression et inversement.
Dans une moindre mesure, la réponse en tension du capteur va être sensible à la température et à l'humidité. Comme pour le gain g_u, on ne cherchera pas à chiffrer l'influence respective de ces deux paramètres.

Soit V une tension réellement acquise et P la pression mesurée par le capteur qui lui correspond. On recherche donc la tension de la caractéristique interpolée V_c telle que :

$$P = f^{inter}(V_c) \tag{B.2}$$

La relation qui relie V et V_c n'est pas une relation de proportionalité. **On fera l'hypothèse que le boitier d'amplification qui impose g_u, ainsi que toute modification des conditions hygrométriques ou de température induisent en plus d'un gain constant un décalage de la tension de repos.** Ainsi, on a :

$$V = a * V_c + b \tag{B.3}$$

Nous avons besoin de deux couples de valeurs (P, V) mesurés lors de l'acquisition afin de déterminer les constantes a et b. On pose (P_{obs1}, V_{obs1}) et (P_{obs2}, V_{obs2}) deux couples de mesures correspondants à l'acquisition.
On pose V_{c1} et V_{c2} les valeurs en tension de la plage [0v,2.8v] telles que :

$$V_{ci} = f^{inter-1}(P_{obsi}) \tag{B.4}$$

On en déduit les valeurs de a et b :

$$a = \frac{V_{obs1} - V_{obs2}}{V_{c1} - V_{c2}} \tag{B.5}$$

$$b = \frac{V_{obs2} * V_{c1} - V_{obs1} * V_{c2}}{V_{c1} - V_{c2}} \tag{B.6}$$

Il s'agit donc pour chaque valeur de tension acquise, d'appliquer la relation suivante :

$$P = f^{inter}\Big(\frac{V - b}{a}\Big) \tag{B.7}$$

Outre les divers problèmes de précision concernant la mesure des couples (P_{obs}, V_{obs}), cette méthode a l'inconvénient d'appeller la fonction inverse de f^{inter}. Le paragraphe B.3.2 est consacré à ce problème. Cependant, force est de constater que cette méthode permet de déterminer les valeurs de pression effectivement mesurées, et ce indépendamment de g_u et de g_{uc}.

B.3.2 Inversion de la caractéristique

Le problème est d'estimer $Vc = f^{inter-1}(P_{obs})$.

Une première façon serait d'utiliser à nouveau la fonction `polyfit()` en interpolant la fonction $g = f^{-1}$ qui calculerait un polynôme de degré N-1 au plus. On pourrait alors déterminer V_c à l'aide de la commande :

$$\texttt{polyval(}g^{inter}\texttt{,}P_{obs}\texttt{)}$$

La fonction `polyval()` donnant la valeur d'un polynôme en un point donné. Cependant, il serait faux de croire que l'on a $g^{inter} = (f^{inter})^{-1}$ car f^{inter} et g^{inter} sont des interpolations. Un moyen d'illustrer ce propos est de tracer :

$$f^{inter}\Big(g^{inter}(P)\Big)$$

pour deux caractéristiques fictives f_1 et f_2 qui associent toutes deux à des tensions comprises entre 0.5v et 2.8v des valeurs de pression allant de 0 à 40 kPa. On présente ci dessous leur interpolation sous matlab (cf figure B.5), réalisées à partir de 5 couples de valeur (P, V). On vérifiera leur caractère bijectif respectif.

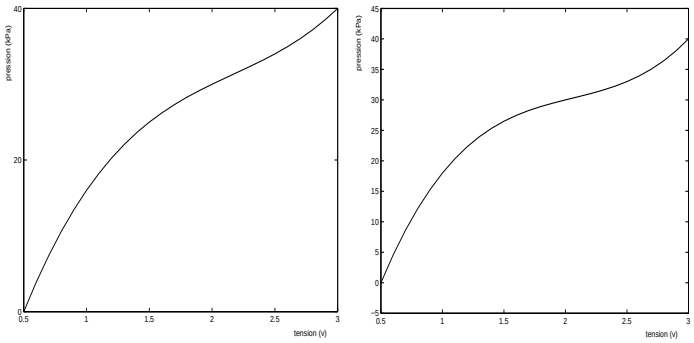

FIG. B.5 – Caractéristiques f_1^{inter} (gauche) et f_2^{inter} (droite)

De même, on va tracer les interpolations g_1^{inter} et g_2^{inter} qui sont obtenues à partir des mêmes couples de valeurs (cf figure B.6).

La bijectivité de g_1^{inter} semble assurée à l'inverse de g_2^{inter}, ce que l'on peut confirmer en traçant $f^{inter}\Big(g^{inter}(P)\Big)$ pour les deux interpolations (cf figure B.7).

Si la méthode d'interpolation des fonctions f et g semblent vérifier de manière satisfaisante la relation $g^{inter} = (f^{inter})^{-1}$ pour la caractéristique 1, on ne peut s'en contenter pour la caractéristique 2. Toute la difficulté réside dans le fait qu'il faut s'assurer de la bijectivité des deux interpolations. Nous allons voir qu'il est possible d'obtenir V_c à la seule condition que l'interpolation de f soit bijective.

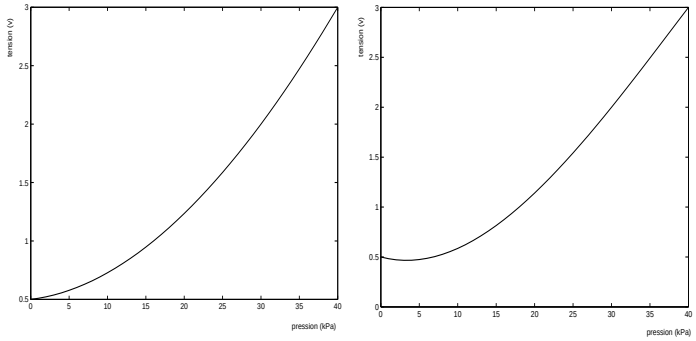

FIG. B.6 – Caractéristiques g_1^{inter} (gauche) et g_2^{inter} (droite)

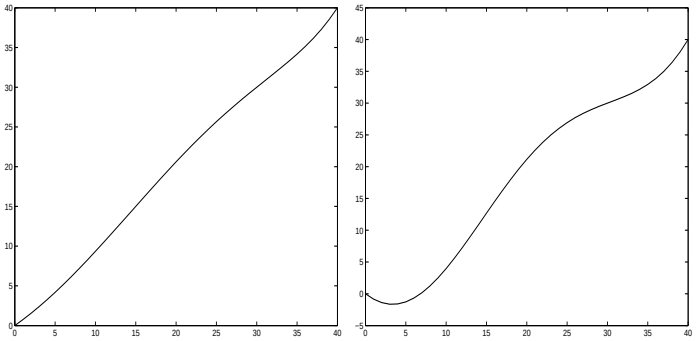

FIG. B.7 – $f^{inter}\Big(g^{inter}(P)\Big)$ pour les caractéristiques 1 (gauche) et 2 (droite)

En effet, une autre méthode possible est de chercher à résoudre :

$$f^{inter}(V_c) = P_{obs}$$

En effet, ceci revient à rechercher les racines du polynôme $f^{inter} - P_{obs}$ à l'aide de la commande `roots()` et de choisir dans les n racines calculées (où n est le degré d'interpolation de f choisi), l'unique à valeur réelle positive comprise entre 0 et 2.8v. A priori, l'unicité de cette dernière solution est bien une garantie de la bijectivité sur la bande [0v,2.8v] de l'interpolation de la caractéristique. Il existe donc un double contrôle de cette bijectivité, graphique en un premiers temps, et par la détermination de V_c ensuite. Ainsi, avant d'extraire V_c des racines du polynôme $f^{inter} - P_{obs}$, il est conseillé de toutes les afficher afin de s'assurer de l'unicité de la solution.

B.3.3 A propos de (P_{obs}, V_{obs})

Nous avons vu au paragraphe B.3.1 qu'il était nécessaire de connaître deux couples de valeurs (P, V), c'est à dire deux valeurs de pression du

système pour deux valeurs de tension en sortie. Et ce, afin de déterminer a et b. Si l'on ne peut s'affranchir de la mesure au manomètre de P_{obs} et donc de tous les risques d'imprécision que cela comporte, on préfèrera au voltmètre, une lecture de V_{obs} directement sur l'acquisition.
L'idée est donc de choisir deux valeurs de pression de telle sorte que leur tension respective V_{obs} soient facilement identifiables sur l'acquisition.

On choisit $P_{obs1} = 0$

L'idée est de laisser, durant l'acquisition, le système soumis à une pression nulle ; et ce, à un instant facilement reconnaissable lorsque l'on va visualiser les données acquises (par exemple, au début ou à la fin de l'acquisition). Ainsi, il sera possible de définir une fenêtre sur laquelle V_{obs1} correspond à la valeur moyenne des tensions acquises.
Ce choix de (P_{obs1}, V_{obs1}) a le double avantage de présenter une pression qu'il est facile de stabiliser, mais aussi de donner une estimation plus précise du décalage de la tension de repos qu'avec n'importe quel autre couple(P_{obs}, V_{obs}).

On choisit P_{obs2} quelconque

En effet, sur le même principe qu'au paragraphe B.3.3, il est tout à fait possible de choisir P_{obs2} quelconque. A condition cependant de pouvoir définir une fenêtre de l'acquisition où l'on sait que V_{obs2} représente la valeur moyenne ou le maximum.
L'idée est donc de soumettre le système à une pression mesurée au manomètre, et ce à un instant reconnaissable sur l'acquisition (on peut par exemple réaliser un plateau en début ou en fin d'acquisition). **Pour les raisons vues au paragraphe B.2.2, il nous est difficile de stabiliser la pression à laquelle est soumi le système, on choisira donc dans `carac()` la valeur maximum de la fenêtre.**

On choisit $P_{obs2} = P_{max}$

A condition que l'on connaîsse la valeur maximum de pression à laquelle le système eut été soumis durant l'acquisition, on peut choisir cette valeur pour P_{obs2} et lui faire correspondre pour V_{obs2} la tension maximum acquise.
Ce choix de P_{obs2} a l'avantage d'éviter à l'utilisateur de définir une seconde fenêtre, en plus de celle qui définit V_{obs1}.

B.3.4 Obtention des valeurs de pression à l'aide de la caractéristique

On peut donc maintenant déterminer les valeurs de pression correspondantes aux tensions acquises. En effet, le paragraphe B.3.2 nous permet de déterminer V_{c1} et V_{c2}, et donc a et b. Il ne reste plus qu'à appliquer la relation (B.7) à chacune des valeurs acquises en utilisant la fonction `polyval()`.
On obtient ainsi une courbe des variations de pression à la même fréquence d'échantillonage que l'acquisition de départ.

On remarquera que les valeurs acquises utilisées dans la relation(B.7) sont des tensions et non des entiers comme c'est le cas pour des fichiers codés en binaire. On rappelle que l'on passe de l'un à l'autre par un coefficient multiplicatif.

B.4 Programme sous Matlab

On cherche à définir un programme qui permette de convertir un fichier acquisition en un double tableau contenant les valeurs de pression mesurées par le capteur et les temps correspondants.

B.4.1 Structure du programme

On définit donc une fonction `carac()` qui accepte en entrée :
- Un tableau ligne ou colonne contenant les données acquises par le capteur A.
- La fréquence d'échantillonnage de l'acquisition.

Et qui renvoie en sortie :
- Un tableau ligne ou colonne contenant les valeurs de pression (en kPa) effectivement mesurées par le capteur A.
- Un tableau ligne contenant les temps (en s) correspondants à chaque valeur de pression.

On choisit ici le cas particulier où les données acquises sont codées en binaire. Il sera nécessaire de multiplier chacune d'entre elles par le coefficient $\beta = 8.612e-05$ pour obtenir l'équivalent en tension.

Afin que le fonction `carac()` puisse être utilisée de la manière la plus large possible (c'est à dire qu'il ne soit pas nécessaire de la modifier systématiquement avant utilisation), il sera demandé à l'utilisateur de préciser lui même :
- Le capteur utilisé lors de l'acquisition.

– Le degré d'interpolation souhaité pour la caractéristique puisque le choix du degré est propre à chaque acquisition.
– Les valeurs des deux couples (P_{obs}, V_{obs}).

B.4.2 Programme

```
function [aP,t]=carac(aV,fe)
```

avec :
`aV` =le tableau ligne ou colonne contenant les données acquises.
`fe` =la fréquence d'échantillonnage.
`aP` =le tableau ligne ou colonne contenant les valeurs de pression.
`t` =le tableau ligne contenant les itérations temporelles.

On charge la caractéristique du capteur utilisé. Pour cela, on possède un fichier .txt contenant un tableau à deux colonnes :
– La première contenant N valeurs de pression en kPa.
– La seconde contenant les N valeurs acquises correspondantes (cf paragraphe B.2.2).

```
[capt,capt_path]=uigetfile('*.txt','choisir caracteristique capteur')
C=load([capt_path capt]);
X=C(:,2);
Y=C(:,1);
clear C;
```

Les N valeurs acquises étant codées en binaire, il convient de les convertir en tensions (on fera de même avec `aV` si nécessaire) :
`beta=8.6124e-05;  X=X*beta;  aV=aV*beta;`

Puis on affiche les couples de valeurs afin de choisir au mieux le degré d'interpolation :

```
figure;plot(X,Y,'o');
n=input('degre d''interpolation souhaite n=')
```

Enfin, on estime f^{inter} :

```
[P1,S1]=polyfit(X,Y,n);
```

On définit les couples (P_{obs2}, V_{obs2}) :

```
q=input('methode Pmax (1) ou methode par etalonnage (2) ???');
```

```
if (q==1);

  Pobs2=input('valeur pression maximum observee (en kPa) Pmax=');
  Vobs2=max(aV);

 else

  figure; plot(aV)
  taille_echantillon=length(aV)

  Pobs2=input('valeur pression etalonee (en kPa) P=');
  m=input('fenetre d''etalonnage, borne inferieure m=');
  M=input('fenetre d''etalonnage, borne superieure M=');
  Vobs2=max(aV(m:M));

end
```

On remarquera qu'il est nécessaire dans la méthode par étalonnage de visualiser `aV` afin de déterminer la fenêtre. De plus, il est conseillé d'afficher le nombre d'échantillons (`length(aV)`) qui donne une idée de l'ordre de grandeur des bornes de la fenêtre, la lecture de la puissance de 10 par laquelle il faut multiplier les abscisses sur les figures Matlab n'étant pas toujours tres lisible.
Ces mêmes remarques s'appliquent à la détermination de (P_{obs1}, V_{obs1}) :

```
figure; plot(aV)
taille_echantion=length(aV)

m=input('fenetre d''etalonage du zero, borne inferieure m=');
M=input('fenetre d''etalonage du zero, borne superieure M=');
Vobs1=mean(aV(m:M));
```

Il s'agit maintenant de déterminer V_{c1} et V_{c2} (cf équation (B.4)) :

```
r=roots(P1)
Vc1=raci(r);

P2=P1;
P2(1,n+1)=P2(1,n+1)-Pobs2;
r=roots(P2)
Vc2=raci(r);
```

On notera la présence de la fonction `raci()` qui pour n racines d'un polynôme, renvoie l'unique comprise entre -0.2 et 3v (c'est à dire appartenant à la plage de définition de la caractéristique à 0.2v près). On prendra soin d'afficher au préalable l'ensemble de ces racines afin de s'assurer de l'unicité de la solution, et donc de la bijectivité de la caractéristique :

```
function a=raci(r);
l=length(r);

for j=1:l;

      if ((r(j)==real(r(j))) & (-0.2<r(j)) & (r(j)<=2.860))
      a=r(j);
      clear r;
      return
      end

end
error('pas de racines!!!!!!!!!!')
clear r;
```

On peut dorénavant déterminer a et b (cf équations (B.5) et (B.6)) :

```
a=(Vobs2-Vobs1)/(Vc2-Vc1);
b=Vobs2-a*Vc2;
```

Et donc déterminer `aP` : `aP=polyval(P1,(1/a)*(aV-b));`
ainsi que l'échelle de temps :
`l=length(aP);  t=[0:1/fe:(l-1)/fe];`

B.4.3 Exemple

Nous allons prendre un exemple similaire au cas du chapitre B.1.1, c'est à dire l'acquisition de deux capteurs différents, mesurant les mêmes variations de pressions d'un système (cf figure B.8). Durant l'acquisition, nous avons pris soin d'effectuer deux paliers équivalents à une pression nulle (début et fin de l'acquisition), ainsi qu'un palier d'étalonnage à 28.45 kPa qui correspond au dernier pic de pression. Enfin, nous avons également noté la pression maximum imposée au système qui est de 52.9 kPa.

On peut visualiser figure B.9 les caractéristiques interpolées pour le capteur 1 (de degré 2) et le capteur 2 (de degré 3).

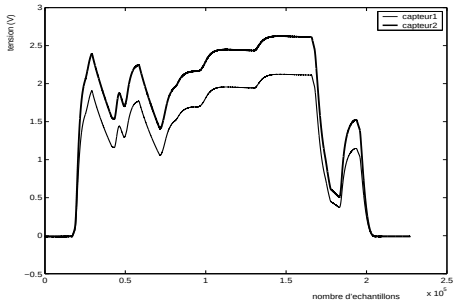

FIG. B.8 – Acquisition capteur 1/ capteur 2 (en gras)

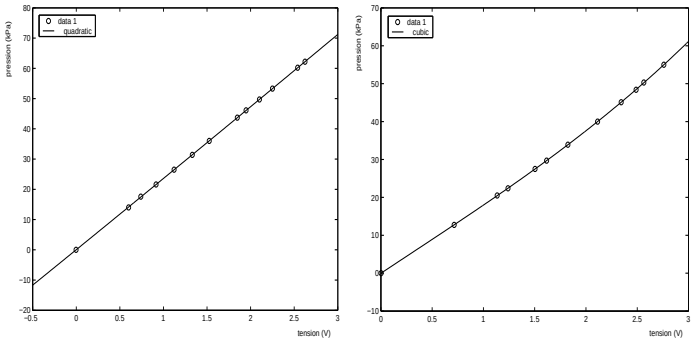

FIG. B.9 – Caractéristiques capteur 1 (gauche)/ capteur 2 (droite)

On peut ainsi comparer la superposition des mesures effectuées par chacun des capteurs selon que l'on considère la réponse en tension linéaire ou que l'on se soit reporté aux caractéristiques interpolées.

Nous constatons sous l'hypothèse de linéarité de la caractéristique (cf figure B.10 (gauche)) qu'il existe des écarts entre les deux courbes de pression de l'ordre de 2.2 kPa (soit environ 4%). Lorsque l'on se rapporte aux interpolations des caractéristiques, on constate pour un étalonnage à 28.45 kPa un écart maximum de l'ordre de 180 Pa (soit environ 0.4%). Ces écarts se trouvent exclusivement aux hautes pressions. Concernant l'étalonnage par le maximum de pression (cf figure B.10 (droite)), on retrouve des écarts du même ordre de grandeur que pour l'étalonnage à 28.45 kPa, mais localisés aux pressions intermédiaires.

On pourra aussi s'assurer de la bonne estimation de la tension de décalage en zoomant sur le début des deux courbes de pression (cf figure B.11).

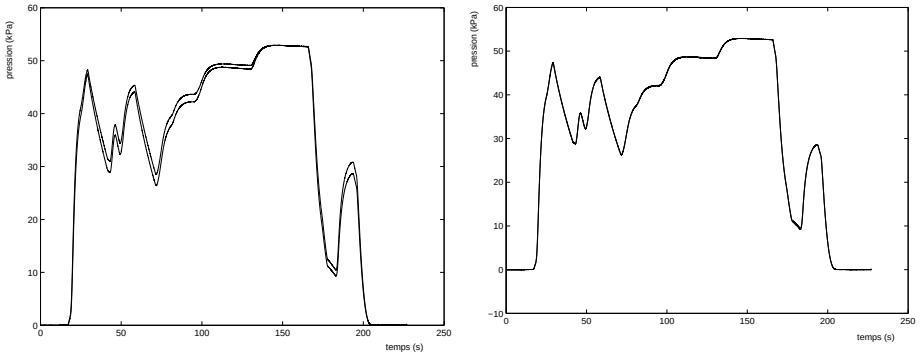

FIG. B.10 – Pour une caractéristique linéaire (gauche) / interpolée (droite)

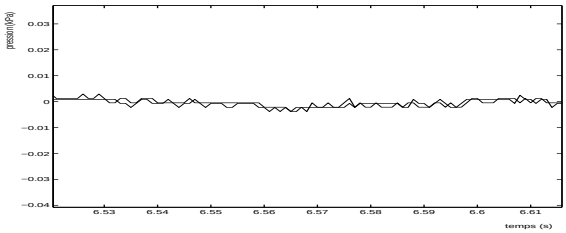

FIG. B.11 – Valeurs autour de $P = 0$

B.5 Interfaces graphiques sous Matlab

Si la fonction `carac()` ne possède seulement que deux arguments d'entrée que sont le fichiers d'acquisition et sa fréquence d'echantillonnage, elle demande à l'utilisateur de préciser au cours du code :

- Le capteur utilisé au cours de l'acquisition, sa caractéristique et le degré de son interpolation.
- Le couple (P_{obs2}, V_{obs2}) qui va dépendre de la méthode d'étalonnage choisie. Ainsi que la fenêtre d'étalonnage qui va permettre d'estimer V_{obs1}.

Le caractère interactif du code nous donne à réfléchir aux avantages que présenterait l'utilisation d'interfaces graphiques sous Matlab (module `GUI`). En effet, Matlab prévoit un ensemble de commandes et de fonctions dont l'utilisation est relativement aisée pour la création et la manipulation d'objets graphiques dont les plus couramment utilisés sont :

- les fenêtres.
- les menus
- les boutons de contrôle du type poussoir, cases à cocher, barre de défilement, zone de texte statique ou éditable (cf figure B.12).

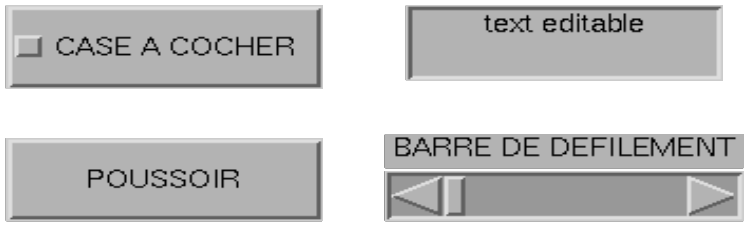

FIG. B.12 – Exemples de contrôle sous Matlab

B.5.1 A propos des contrôles sous Matlab.

Pour créer un contrôle, on utilisera la fonction `uicontrol()`. On utilise cette fonction en respectant la syntaxe suivante :

```
h=uicontrol(hfig,'NomPropriété','ValeurPropriété');
```

On rappelle que chaque objet graphique possède un identifiant unique ("handle" en anglais) qui équivault à ce que l'on pourrait qualifier de poiteur. Le pointeur d'une fenêtre est un entier affiché par défault dans la barre de titre de celle ci. Quand au pointeur d'un contrôle, c'est un flottant qu'il est préférable de stocker dans une variable.
Lors de la création d'un contrôle, on a :

- `h` : pointeur sur le contrôle créé.
- `hfig` : pointeur de la fenêtre sur laquelle on veut créer le contrôle.
- `'NomPropriété'` : corresponds au nom de la propriété que l'on veut définir telle le type de contrôle (`'Style'`), sa position sur la fenêtre (`'position'`), ou bien sa fonction lorsque le contrôle est actionné (`'callback'`).
- `'ValeurPropriété'` : peut être un nombre, un tableau ou une chaîne de caractères selon le type de proprité.

B.5.2 Position du problème.

On distingue deux problèmes :

1. La fonction `carac()` étant déja écrite, le problème principal sera de la partitionner en plusieurs actions toutes contrôlées séparément.
2. La fonction `carac()` agit de manière ordonnée. C'est à dire qu'elle va réaliser une suite d'opérations sur les données acquises, et ce dans un ordre précis qu'il faudra conserver. Or l'utilisation de contrôles graphiques implique que chaque activation de l'un d'entre eux entraîne une action. Le problème sera de s'assurer que :

– l'activation répettée d'un contrôle n'entraine pas d'erreur de calcul en effectuant deux fois de suite la même opération.
– l'activation dans le désordre des contrôles n'entraîne pas le calcul de solutions incohérentes.

B.5.3 Solutions proposées.

Bien évidemment, il ne s'agit pas de reécrire un nouveau programme, mais bel et bien de modifier l'ossature de la fonction `carac()` en la partionnant en plusieures fonctions que l'on contrôle séparément.

Partition de la fonction carac().

On peut diviser la fonction `carac()` en un premier temps en trois parties au rôle bien distinct :

1. L'acquisition et le traitement des PARAMETRES DE L'ACQUISITION : Il s'agit de faire l'acquisition de la caractéristique du capteur utilisé, de choisir un degré d'interpolation et de transformer les données acquisises en valeurs de tension dans le cas où le fichier serait codé en binaire.
2. L'ETALONNAGE DES MESURES : Cette partie fait l'acquisition des deux couples de valeurs (P_{obs}, V_{obs}) qui vont dépendre de la méthode d'étalonnage choisie (par un palier ou par le maximum).
3. La restitution de la COURBE EN PRESSION en fonction du temps : Il s'agit naturellement de calculer les valeurs de pression et l'echelle de temps et de les restituer dans un tableau double colonnes.

L'ordre d'appel des contrôles.

La fonction `carac()` réalise une suite d'opérations sur un tableau colonne de données. Nous avons vu au chapitre B.5.2 que l'ordre de ces opérations a son importance. Afin d'illustrer le type d'erreurs qui pourraient subvenir, regardons deux opérations successives réalisées par la fonction `carac()` sur le tableau de données :

– Opération 1 : Le passage des données codées en binaire à des valeurs de tensions en les multipliant par un coefficient β constant.
– Opération 2 : L'acquisition des couples de valeur (P_{obs}, V_{obs}).

On peut ainsi illustrer deux types d'erreurs dans le cas où ces deux opérations seraient actionnées par des contrôles distincts.

- **Une erreur de répetition** dans le cas où l'opération 1 serait répettée deux fois. Les données seraient alors multipliées par β^2. En effet, on rappelle que la ligne de commande de la fonction `carac()` qui réalise l'opération 1 s'écrit :
  `beta=8.6124e-05;  aV=aV*beta;`
- **Une erreur d'ordre d'appel** dans le cas où l'opération 2 serait appellée avant l'opération 1. En effet, l'opération 2 consiste à chercher le maximum d'un tableau de données et à le restituer sous V_{obs} en tant que valeur de tension.Cette valeur serait fausse si l'opération 1 n'a pas été réalisée.

Un moyen d'éviter au mieux ces types d'erreur est de travailler à partir de variables globales. Il s'agit de définir comme tel toutes les variables succeptibles d'être modifiées par le code. Toutes ces variables représenteront une opération effectuée sur les données et seront initialisées à zero au début du code.
Reprenons l'exemple des opérations 1 et 2. Posons les variables `a` et `aV` définies comme globales et qui représentent respectivement les données acquises brutes et en tension.
Si l'on définit l'opération 1 comme :

```
beta=8.6124e-05;  aV=a*beta;
```

Et l'opération 2 comme une opération effectuée sur la variable `aV` nécessairement non nulle, nous constatons que :

- **il est impossible de faire une erreur de répetition** sur l'opération 1 puisqu'à chaque actionnement du contrôle correspondant, la variable `aV` sera écrasée puis redéfinie.
- **il est possible d'éviter les erreurs d'ordre** en imposant un test avant l'opération 2 qui verrifie que la variable `aV` est non nulle. En effet, si l'opération 1 n'a pas été réalisée avant l'opération 2, il est possible si `aV==0` d'afficher un message d'erreur qui va rappeller à l'utilisateur les actions oubliées (cf figure B.13).

Bien evidemment, l'exemple d'erreur ci dessus que l'utilisation d'interfaces graphiques sous matlab peut entrainer n'est pas exhaustif. Il serait cependant trop laborieux de détailler la rédaction du nouveau code. Le plus simple est encore de montrer un exemple d'utilisation de l'interface.

B.5.4 L'utilisation de l'interface.

La figure B.14 présente le résultat final de l'interface graphique principale de calibration des mesures. On retrouve la partition du code (présentée au

FIG. B.13 – Messages d'erreur affichés lorsque a==0 (gauche) ou aV==0 (droite)

paragraphe B.5.3) selon les parties :

- PARAMETRES DE L'ACQUISITION.
- ETALONNAGE DES MESURES.
- COURBE EN PRESSION

Ces parties sont bien evidemment à suivre dans l'ordre chronologique, les PARAMETRES DE L'ACQUISITION et l'ETALONNAGE DES MESURES devant être définis avant le calcul permettant d'obtenir les courbes de pression en fonction du temps.

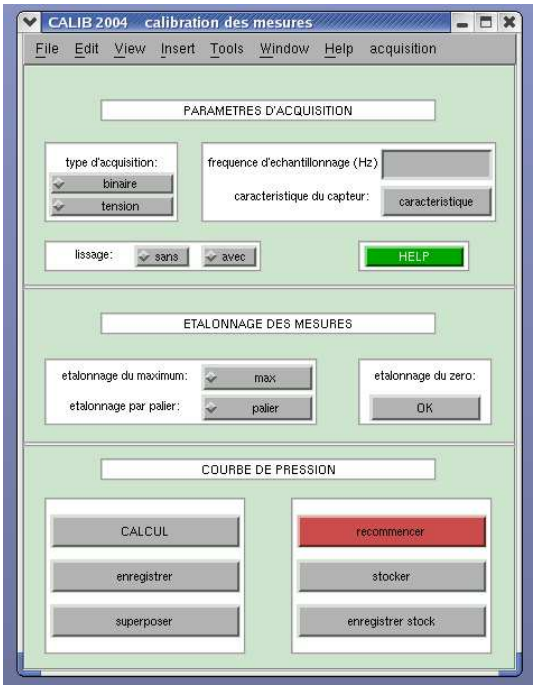

FIG. B.14 – Interface graphique principale

Nous allons suivre les étapes d'une calibration de mesures à l'aide de l'interface graphique en détaillant le type de message d'erreur qu'elle pourrait rencontrer.

L'acquisition de données.

On actionne le menu **acquisition** qui va proposer les trois sous menus **voie 0**, **voie 1** et **voie 2**. S'affiche alors à l'ecran une fenêtre "explorateur" qui propose de selectionner un fichier .txt contenant un tableau possèdant autant de colonnes que de voies enregistrées lors de l'acquisition. Le code n'enregistrera que les données de la colonne correspondant à la voie selectionnée dans le menu **acquisition**.
Dans l'exemple de la figure B.15, les données se situent dans un répertoire **acquisitions**.

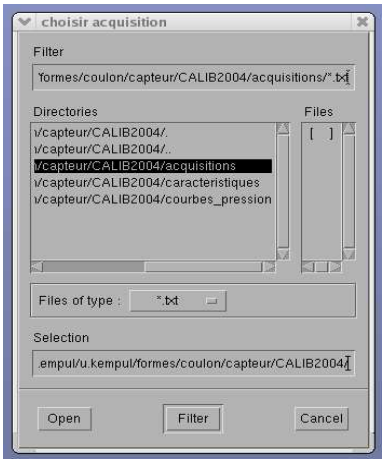

FIG. B.15 – Fenêtre "explorateur" pour l'acquisition de données.

Partie : PARAMETRES DE L'ACQUISITION.

On dénombre quatres actions spécifiques propres à cette partie :

- Le choix du **type d'acquisition** par un contrôle de type radio au choix exclusif. Dans le cas où les données seraient du type binaire, une fenêtre s'affiche à l'ecran (figure B.16) proposant une valeur de β par défault que l' on peut modifier (contrôle du type texte éditable).
- Le choix de la **fréquence d'échantillonnage** dans une fenêtre éditable.

FIG. B.16 – Valeur par défault du coefficient β

– Le choix de la **caractéristique du capteur** qui va afficher une fenêtre "explorateur" (figure B.17) afin de selectionner un fichier .txt contenant N couple de valeurs (P, V) qui caractérisent le capteur correspondant à la voie choisie dans le menu.
Dans l'exemple de la figure B.17, les fichiers .txt se situent dans un répertoire **caractéristiques**.

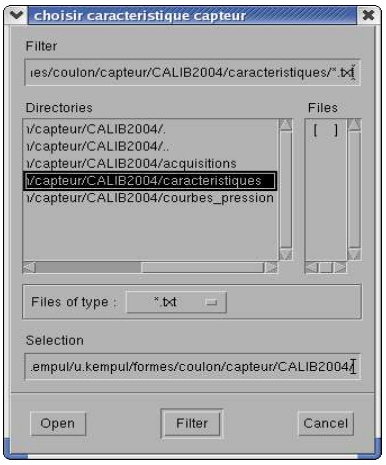

FIG. B.17 – Fenêtre "explorateur" pour l'acquisition de la caractéristique.

Le code va afficher à l'écran une fenêtre présentant les N couples de points (P, V) (figure B.18).
L'option `basic fitting` proposée par le menu des fenêtres Matlab permet de tracer les interpolations et de choisir le degré le plus approprié (l'exemple de la figure B.18 présente l'interpolation de degré 5 et de degré 10).
Le choix du degré d'interpolation est définitif lorsque sa valeur est rentrée sur la fenêtre éditable et l'action validée par le contrôle ' OK '.

– Le choix du **lissage** : C'est une fonction qui n'était pas sur `carac()`.

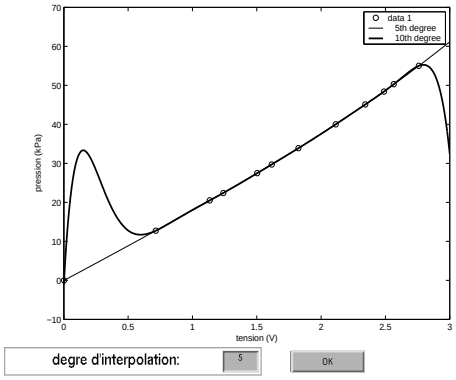

FIG. B.18 – Choix du degré d'interpolation de la caractéristique.

Elle permet de lisser les courbes de données lorsque celles-ci présentent un signal bruité. Pour cela, on fait la moyenne sur n valeurs consécutives de la courbe que l'on attribue au centre des n points. On effectue cette opération tous les $n/2$ points de la courbe et on relie les les valeurs par des segments de droite de $n/2$ points de telle sorte que la courbe lissée possède le même nombre de valeur que la courbe initiale.
Cette fonction est contrôlée par deux boutons ratio au choix exclusif. Dans le cas où l'utilisateur souhaite un lissage, une fenêtre apparaît où sont superposées la courbe initiale et la courbe lissée (figure B.19). Un contrôle de type slider permet de faire varier la valeur de n, un bouton ' OK ' pour la valider.

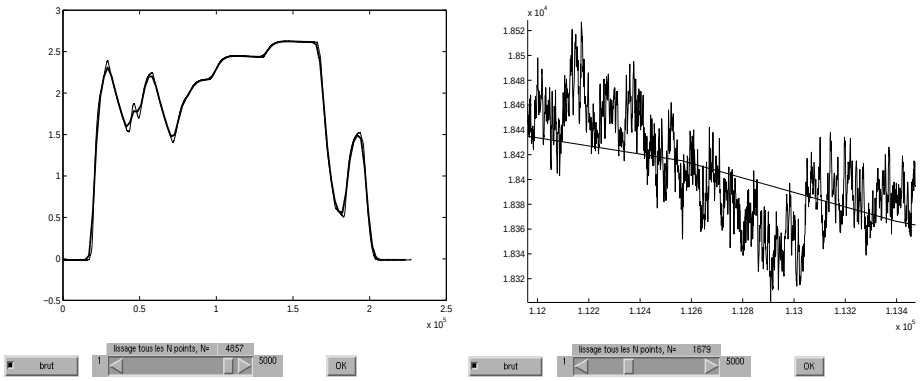

FIG. B.19 – Cas d'une courbe de données standard (gauche) Cas d'un signal bruité (droite).

– Un bouton **HELP** qui lance la commande `helpwin` et qui affiche ainsi un message d'aide à l'utilisation de l'interface.

Partie : ETALONNAGE DES MESURES.

Cette partie a pour but de déterminer les deux couples de valeurs (P_{obs}, V_{obs}).

1. **Le couple** (P_{obs1}, V_{obs1}) : il s'agit **d'étalonner le zero** de l'acquisition, c'est à dire de déterminer une fenêtre sur laquelle la valeur moyenne des tensions sera considérée comme la tension acquise pour une pression nulle. Le contrôle ' OK ' fait apparaître à l'écran le graphique des données, le bouton de gauche de la souris permettant de zoomer sur la fenêtre désirée, le contrôle ' Retourner la fenêtre ' de la valider (cf figure B.20).

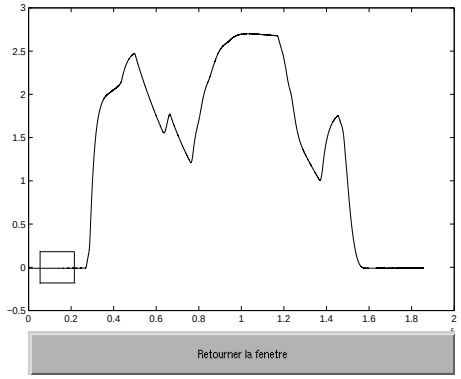

FIG. B.20 – Choix de la fenêtre d'étalonnage du zero de l'acquisition.

2. **Le couple** (P_{obs2}, V_{obs2}) : comme nous l'avons vu au paragraphe B.3.3, nous distinguons deux méthode d'étalonnage :
   - par le maximum de pression. Un fenêtre s'affiche alors à l'écran demandant à l'utilisateur d'entrer la valeur maximale de pression observée durant l'acquisition (cf figure B.21).

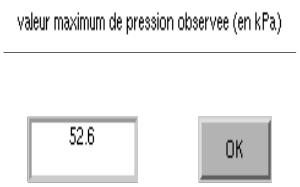

FIG. B.21 – Etalonnage par le maximum de pression observée.

– par une valeur de pression quelconque, le code demandant successivement la valeur de cette pression et la fenêtre de l'acquisition correspondante (cf figure B.22).

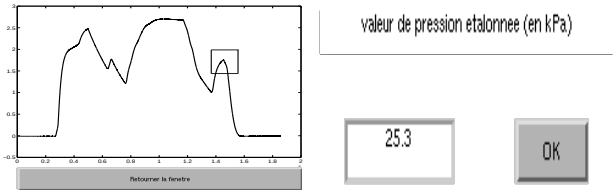

FIG. B.22 – Etalonnage par une valeur quelconque de pression observée durant l'acquisition.

On remarquera que le choix de la méthode d'étalonnage concernant (P_{obs2}, V_{obs2}) est evidemment exclusif. De plus, il existe un certain nombre de messages d'erreur, notemment dans le cas où le type d'acquisition n'aurait pas été précisé.

Partie : COURBE DE PRESSION.

On distingue six fonctions au sein de cette partie :

1. La fonction **CALCUL** qui finit de calibrer les mesures acquises.
2. La fonction **enregistrer** qui ouvre une fenêtre "explorateur" et qui enregistre dans un fichier .txt les données en pression sous formes d'un tableau à deux colonnes, la première contenant les valeurs de pression calculées, la seconde les temps itérés correspondants.
3. La fonction **superposer** qui affiche une autre interface graphique (figure B.23). Cette interface permet de superposer (`hold on`) ou d'afficher séparément (`subplot`) plusieures courbes afin de les comparer. Pour cela, on charge succéssivement les fichiers .txt en précisant la couleur d'affichage désiré dans un menu qui apparaît à l'écran (contrôle **charger données**), qu'on valide systématiquement avec le contrôle **ajouter**.
4. La fonction **recommencer** qui relance l'application. Cela a pour conséquences de fermer toutes les fenêtres et de réinitialiser tous les paramètres. Dans le cas où l'utilisateur désirerait calibrer deux courbes acquises distinctes, deux possibilités s'offrent à lui, une fois la première calibration réalisée :
   – faire l'acquisition dans le menu du second fichier de données qui présente l'avantages de conserver les paramètres définis lors de la

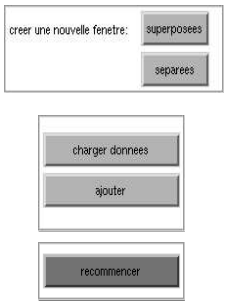

FIG. B.23 – Interface graphique de la fonction **superposer**.

première calibration, telle que la fréquence d'échantillonnage, le choix de la caractérisitique et son degré d'interpolation. Cependant, cela présente le risque d'omettre de modifier l'un d'entre eux si tel était nécessaire et de lancer un calcul faux. En effet, les paramètres possédant une valeur, le calcul se réalisera sans afficher de message d'erreur. Une manière de répondre à ce problème est de désactiver systématiquement lorsque l'on charge un fichier de données tous les contrôles du type ratio (choix du type d'acquisition, lissage etc...) et d'afficher en rouge sur l'interface les paramètres qui possédent déja une valeur et qu'il est peut-être nécessaire de modifier.

- une autre méthode est d'utiliser la fonction **recommencer** qui réinitialise systématiquement toutes les variables du code.

5. Les fonctions **stocker** et **enregistrer stock** qui permettent d'enregistrer les valeurs en pression de plusieures acquisitions dans un même fichier .txt sous forme d'un tableau de n colonnes. La première ligne contenant le numéro de la voie, la seconde la fréquence d'échantillonnage afin que l'on puisse réstituer l'echelle de temps, et le reste les valeurs de pression.

Annexe C

Traitement d'images sous Matlab

C.1 Introduction

C.1.1 Position du problème

Il s'agit d'utiliser les outils de traitement d'images disponibles sous Matlab (**image processing toolbox**.) afin de déterminer l'aire, la longueur et la largeur de l'ouverture d'une l'anche double.
L'intérêt est de pouvoir réaliser ce calcul

- en post-traitement, c'est à dire à partir d'images isolées ou de séquences d'images.
- lors de mesures en régime statique.

On travaillera à partir d'images ou de séquences obtenues à l'aide d'une caméra DV du commerce placée en face de l'ouverture.

C.1.2 Les images sous Matlab

On rappelle que Matlab considère une image comme un quadrillage de pixels. A chaque pixel va correspondre une ou plusieurs valeurs contenues dans un tenseur d'ordre 2 ou 3. Toutes les images ne sont donc pas codées de la même manière. Dans ce qui va suivre, on distinguera essentiellement les images en couleur des images en noir et blanc.

1. **Les images en couleur ou RGB** sont codées sous la forme de tenseurs du troisième ordre (ou matrices cubiques), à chaque pixel étant associées trois valeurs correspondant au degré de vert, de rouge et de bleu.

2. **Les images en noir et blanc** qui peuvent être codées
   - comme des images couleurs, avec pour chaque pixel des valeur de degré de vert, de bleu et de rouge identiques.
   - sous forme de matrices (deux dimensions), dont chaque élément présente une valeur comprise entre 0 (le noir) et 1 (le blanc) correspondant au niveau de gris du pixel.

En résumé, une image composée de m*n pixels sera codée sous Matlab par un tenseur de dimension m∗n∗3 si elle est en couleur, et par un tenseur de dimension m∗n∗3 ou une matrice de dimension m∗n si elle est en noir et blanc.

Par la suite, on parlera d'images en niveau de gris plutôt que d'images en noir et blanc, réservant cette dernière appellation aux images codées par une matrice dont chaque élément possède comme valeur 1 ou 0.

C.1.3 Comment Matlab calcule-t-il une aire

Comme nous l'avons vu au paragraphe C.1.2, Matlab va travailler à partir de tenseurs. **Modifier une image, c'est donc modifier la valeur des éléments d'un tenseur ou ses dimensions.**

Matlab ne peut donc donner directement une valeur d'aire mais un nombre de pixels. Pour cela, on travaillera à partir d'images noir et blanc (la valeur des éléments est 1 ou 0), que l'on appelle ausi des **images binarisées**. On peut visualiser un exemple en figure C.1. Cet exemple est volontairement de petites dimensions afin que l'on puisse aisément distinguer les pixels.

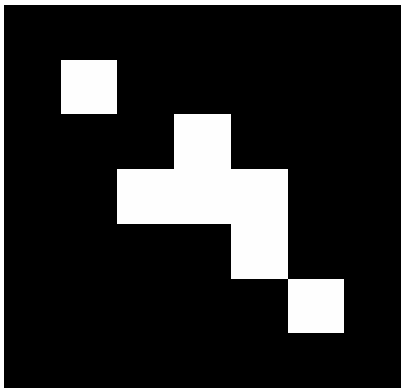

FIG. C.1 – Exemple d'image en noir et blanc

Sous Matlab, cette image est codée par la matrice :

```
image =

     0     0     0     0     0     0     0
     0     1     0     0     0     0     0
     0     0     0     1     0     0     0
     0     0     1     1     1     0     0
     0     0     0     0     1     0     0
     0     0     0     0     0     1     0
     0     0     0     0     0     0     0
```

L'idée est de séparer les différentes taches (en blanc) que contient une image binarisée et de déterminer le nombre de pixels respectifs qui les composent. Dans l'exemple précédent, on distingue clairement deux taches : une en haut à gauche composée d'un unique pixel, une seconde au centre composée de six pixels.

On peut séparer et numéroter les différentes taches de l'image à l'aide de la fonction Matlab `bwlabel()`. Cette fonction renvoie un entier correspondant au nombre de taches distinguées et une matrice de mêmes dimensions que l'image. A chaque pixel de l'image va correspondre sur la matrice crée :

- un zero si le pixel ne correspond pas à une tache.
- l'entier correspondant à la tache à laquelle appartient le pixel.

Reprenons l'exemple précédent :

```
>> [taches,nombre_tache]=bwlabel(image)
taches =

     0     0     0     0     0     0     0
     0     1     0     0     0     0     0
     0     0     0     2     0     0     0
     0     0     2     2     2     0     0
     0     0     0     0     2     0     0
     0     0     0     0     0     2     0
     0     0     0     0     0     0     0

nombre_tache =
     2
```

Matlab ne se contente pas uniquement de distinguer les différentes taches qui composent une image binarisée. En effet, il est capable de donner un certain nombre d'informations sur ces dernières à l'aide de la fonction `regionprops()` qui va renvoyer une structure contenant notamment le nombre de pixels qui composent chacune des taches de l'image (propriété `Area`). C'est lors de l'appel de la fonction `regionprops()` que l'on indique quels types d'informations l'on désire extraire.

```
>> Proprietes=regionprops(taches,'Area')

Proprietes =
2x1 struct array with fields:
    Area

>> Proprietes(1).Area
ans =
     1

>> Proprietes(2).Area
ans =

     6
```

L'indice de la structure correspond à l'identifiant de la tache considérée.

Ce sera à l'utilisateur de convertir ce nombre de pixels en une valeur d'aire en déterminant l'équivalent en dimension d'aire d'un pixel. Nous verrons cela plus en détails par la suite dans le paragraphe C.3.2.

C.2 Binarisation de l'image

Le paragraphe C.1.3 nous indique qu'il est nécessaire de passer de l'image d'origine (en couleur ou niveau de gris) à une image binarisée.

C.2.1 Choix du seuil de binarisation

On binarise une image sous Matlab à l'aide de la fonction `im2bw()`. Celle-ci accepte en entrée une image codée en niveau de gris (ie une matrice dont les valeurs sont comprises entre 0 et 1) et un réel compris entre 0 et 1 qui correspondra au seuil de binarisation de l'opération (les valeurs au dessus du seuil sont remplacées par des 1, les valeurs au dessous par des 0). C'est en

jouant avec ce seuil que l'on fait apparaître les taches qui sont susceptibles de nous intéresser. L'exemple de la figure C.2 illustre l'importance de ce choix, la tache correspondant à l'ouverture de l'anche double pouvant disparaître si ce dernier n'est pas judicieux.

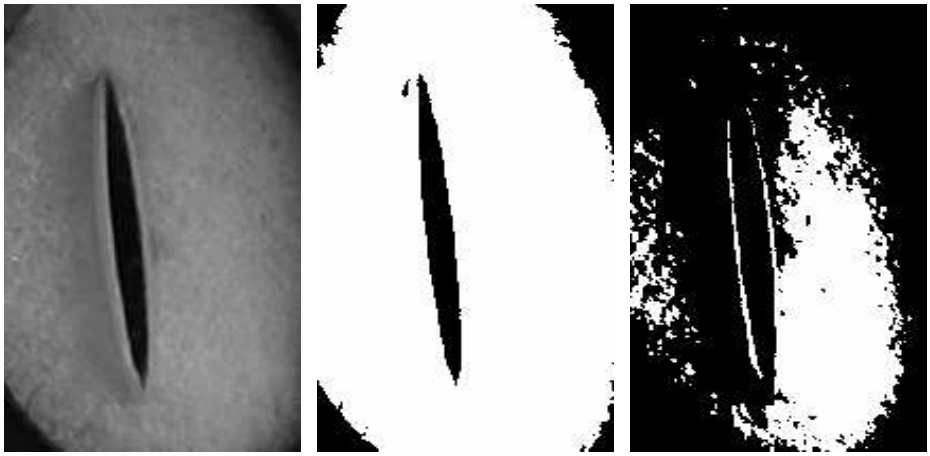

FIG. C.2 – A gauche : image d'origine. Au centre et à droite : exemples de binarisation pour deux seuils différents.

On remarquera :

- **qu'il peut être nécessaire de passer au préalable d'une image codée en RGB (tenseur d'ordre 3) à une image en niveau de gris avant de passer à la binarisation** (fonction `rgb2gray()`).
- **qu'il peut être nécessaire d'inverser la binarisation**, c'est à dire de changer les 0 en 1 et inversement. En effet, nous avons vu que Matlab considérait comme taches les zones où les pixels sont blancs. Ainsi, sur l'exemple de la figure C.2, il est nécessaire que la tache qui nous intéresse soit de couleur blanche (en l'occurence dans l'exemple l'ouverture de l'anche). On inverse la binarisation en utilisant la fonction `imcomplement()` qui transforme une image binarisée en son inverse. La figure C.3 illustre ce propos.

C.2.2 Filtrage de l'image

Nous avons vu au paragraphe C.2.1 qu'il est nécessaire de définir une tache dont l'aire est la plus proche possible de l'aire physique réelle de l'ouverture de l'anche. Dans cette optique, le choix du seuil de binarisation est essentiel. Cependant, d'autres outils peuvent selon certains cas venir s'ajouter au choix d'un seuil judicieux afin de faire ressortir la tache qui représente l'ouverture de l'anche. On parlera de **fonctions de filtrage**.

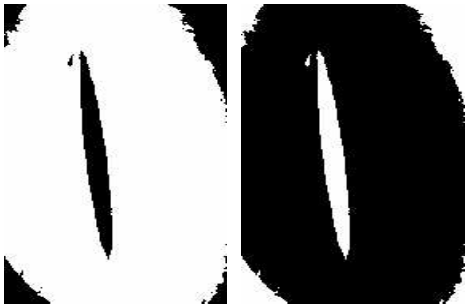

FIG. C.3 – Exemple d'inversion de binarisation.

En plus de faire ressortir encore plus nettement la tache principale, elles ont l'avantage de réduire parfois considérablement le nombre de taches sur l'image binarisée (en pratique, de plusieurs centaines à quelques dizaines), ce qui représente un gain de temps non négligeable lors de l'utilisation de la fonction `bwlabel()` (on rappelle que cette dernière extrait de l'image la totalité des taches, que ce soit la tache principale ou tout pixel blanc isolé, d'où l'intérêt d'éliminer au maximum ces derniers).

Voici une liste non exhaustive de ces fonctions de filtrage disponibles sous Matlab, que nous avons utlisées.

Ouverture et fermeture morphologique.

Avant de définir ce que représente les actions d'ouverture et de fermeture morphologique, il convient de se pencher en un premier temps sur les opérations d'**érosion** et de **dilatation** en matière de traitement d'image.

Ces deux opérations s'appliquent à des images binarisées. Elles agissent sur ces dernières par l'intermédiaire d'un **élément structurant** (qui peut être un cercle, un rectangle, un losange etc...) dont les dimensions sont définies par l'utilisateur (cf fonction `strel` sous Matlab). Cet élément structurant va se déplacer sur chacun des pixels de l'image et modifier sa valeur (0 ou 1) selon sa position par rapport au pixel noir le plus proche.

Les difinitions de ces opérations de morphologie mathématique sont les suivantes :

Erosion :

- Ensemble $X \subset \mathbb{R}^2$ (en pratique une image)
- Elément structurant $B \subset \mathbb{R}^2$ centré en x : B_x (une boule de centre x)
- L'érodé de X par B est défini par $Y \triangleq \{x | B_x \subset X\}$
- Notation : $Y = E^B(X)$

Dilatation :

- Ensemble $X \subset \mathbb{R}^2$ (en pratique une image)
- Elément structurant $B \subset \mathbb{R}^2$ centré en x : B_x (une boule de centre x)
- Le dilaté de X par B est défini par $Y \triangleq \{x | B_x \cap X \neq 0\}$
- Notation : $Y = D^B(X)$

Le plus simple est encore d'observer un exemple afin de mieux comprendre l'action respective de ces deux fonctions. La figure C.4 présente un exemple d'érosion et de dilatation sur une image binarisée pour un élément structurant d'un pixel.

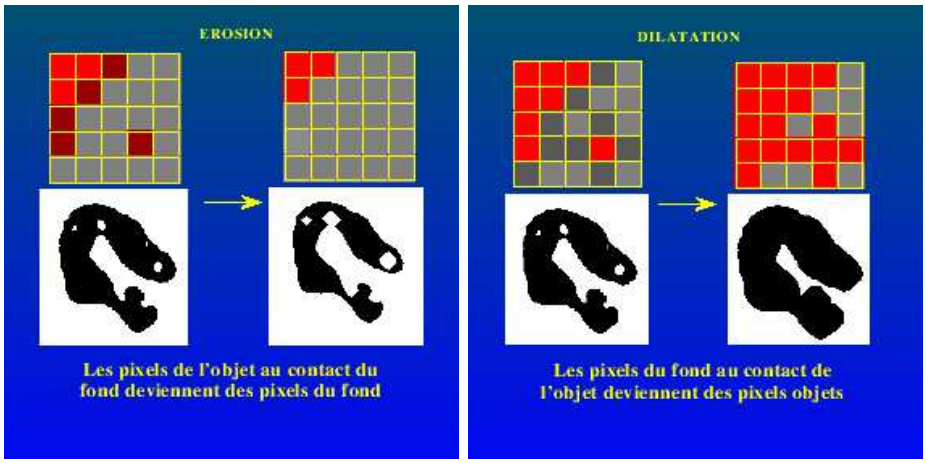

FIG. C.4 – Exemple d'une érosion (gauche) / dilatation (droite)

- **Une ouverture morphologique est la suite d'une opération d'érosion puis de dilatation de l'image binarisée.** Elle est assurée sous Matlab par la fonction `imopen()` qui va opérer l'ouverture morphologique sur une image binarisée, et ce à partir d'un élément structurant commun aux deux opérations. La définition de cette opération de morphologie mathématique est la suivante :
  - Ensemble $X \subset \mathbb{R}^2$ (en pratique une image)
  - Elément structurant $B \subset \mathbb{R}^2$ centré en x : B_x (une boule de centre x)
  - L'ouverture de X par B est définie par le dilaté par la transposée de B de l'érodé de X par B
  - Notation : $Y = O^B(X) \triangleq D^{B^T}\left(E^B(X)\right)$

  La figure C.5 présente comment l'utilisation successive de l'érosion et de la dilatation assure un filtrage de l'image binarisée.

FIG. C.5 – Principe de l'ouverture morphologique.

– **Une fermeture morphologique est la suite d'une opération de dilatation puis d'érosion de l'image binarisée.** Elle est assurée sous Matlab par la fonction `imclose()` qui va opérer la fermeture morphologique sur une image binarisée, et ce à partir de mêmes types d'éléments structurants que ceux utilisés par la fonction `imopen()`.

  La figure C.6 présente comment l'utilisation successive de la dilatation et de l'érosion assure un filtrage de l'image binarisée.

FIG. C.6 – Principe de la fermeture morphologique.

Il est évident que l'ouverture et la fermeture morphologique ont des actions contraires que l'on peut relier par une inversion de binarisation. On remarquera que ces deux fonctions ont l'intérêt de :

– netoyer les régions principales, que ce soit en supprimant les zones noires de petites dimensions situées au milieu de taches blanches ou en supprimant les taches blanches de petites dimensions isolées dans une zone noire.
– relier entre elles des taches blanches principales ou des zones noires principales.

Lorsque l'on parle de petites dimensions, cela sous-entend que l'on oberve des dimensions comparables aux dimensions de l'élément structurant choisit.

Remarque : les exemples d'opérations morphologiques présentés par les figures C.4, C.5 et C.6 considèrent comme des taches les pixels de couleur

noir. On remarquera que Matlab considère comme tache les pixels blancs. Ainsi l'exemple d'ouverture morphologique de la fgure C.5 correspond à une fermeture sous Matlab. Il en va de même pour les autres exemples présentés : la fermeture présentés figure C.6 correspond à une ouverture sous Matlab, l'érosion à une dilatation et inversement.

La fonction imfill

La fonction `imfill()` permet de convertir des zones noires (valeurs 0) en blanc (valeur 1). Pour cela, il est nécessaire de donner en paramètre d'entrée au moins un point appartenent à la zone que l'on désire convertir. Evidemment, l'inversion de la binarisation de l'image en amont, puis en aval de la fonction `imfill()` permet d'obtenir l'action inverse (conversion de zones blanches en zones noires).

Puisque l'enjeu majeur du filtrage est la diminution du nombre de taches blanches, c'est l'utilisation conjointe de la fonction `imfill()` et de l'inversion de la binarisation que nous privilegierons. L'exemple de la figure C.7 rentre dans ce cadre d'utilisation et montre le cas où l'on rentre en entrée l'image binarisée inversée ainsi qu'un point de l'image situé à coté de la tache principale (en bas, dans le coin gauche de l'image).

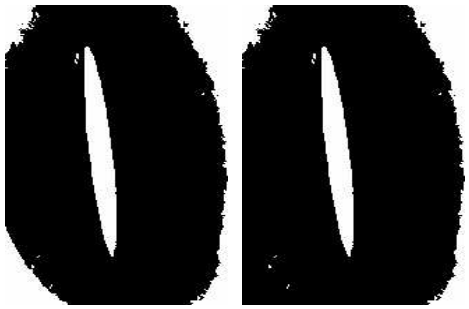

FIG. C.7 – Image originale non inversée (gauche) et résultat par l'utilisation de la fonction imfill après inversion(droite).

Si ces fonctions permettent de mieux distinguer la tache représentant l'ouverture de l'anche du reste de l'image, elles ont surtout l'avantage de permettre de réduire considérablement le nombre total de taches et donc le temps de calcul.
Ce gain de temps sera très utile par la suite, lorsqu'il s'agira de calculer l'aire d'ouverture sur une séquence d'images ou lors de l'acquisition en temps réel.

C.3 Le calcul de l'aire

Le chapitre C.2 nous indique comment faire apparaître la tache qui permettra de déterminer l'aire d'ouverture de l'anche, ainsi que les moyens de réduire le nombre total de taches de manières significatives.
Deux problèmes se posent alors avant de pouvoir déterminer l'aire réelle d'ouverture :

1. **Il est nécessaire de distinguer la tache qui nous intéresse des autres**. En effet, nous avons vu que Matlab va calculer l'aire (en pixels) de toutes les taches présentes sur l'image binarisée et filtrée. Il s'agira de trouver l'aire qui nous intéresse.
2. **Passer d'une aire en pixels à une aire en dimensions physiques**.

C.3.1 Distinguer une tache en particulier

Le paragraphe C.1.3 nous indique que la fonction `bwlabel` va créer une matrice `taches` de même dimension que l'image binarisée. L'idée est de déterminer quel identifiant (un entier) la fonction a attribué à la tache qui nous intéresse.
On pourra utiliser la fonction `ginput(j)` qui va renvoyer les abscisses et ordonnées de j points de la figure courante, points que l'on sélectionne avec la souris (l'origine du repère est en haut, à gauche, l'axe des ordonnées étant orienté vers le bas). On peut ainsi en cliquant sur la tache qui représente l'ouverture, déterminer l'abscisse et l'ordonnée d'un point lui appartenant. De ces valeurs, on peut déduire les coordonnées du pixel correspondant. **Attention, un pixel est repéré par la ligne, puis la colonne à laquelle il appartient dans la matrice qui code l'image. Il s'agit donc de prendre les parties entières des coordonnées précédemment obtenues et d'en inverser l'ordre.**

Une fois les coordonnées du pixel connues, la valeur de la matrice `taches` pour l'élément correspondant va donner l'identifiant. On pourra donc accéder à l'aire de la tache en créant une structure contenant les aires de toutes les taches et en regardant la valeur qui a pour indice l'identifiant.

La commande sous Matlab est :

```
imshow(image_binarise)

[x,y]=ginput(1);
x=floor(x);
y=floor(y);
```

```
[taches,n] = bwlabel(image_binarisee,4);
index=taches(y,x);

Structure=regionprops(tache,'Area');

Aire_pixel=Structure.Area(index);
```

C.3.2 Déterminer la valeur physique de l'aire

Il suffit pour cela de déterminer les dimensions physiques d'un pixel. Cela implique nécessairement une dimension de l'image qui est connue, telle que la longueur de l'anche, son épaisseur... On peut aussi envisager de placer un repère dans le champ de la caméra, mais toute la difficulté sera de le placer dans le même plan que celui de l'ouverture.
De la même manière qu'au chapitre C.3.1, on détermine les coordonnées des deux points de l'image correspondants aux limites de la dimension connue. Il n'est pas nécessaire de considérer leurs arrondis puisque nous ne nous référons pas à une matrice mais uniquement à une distance. De plus, travailler sur les coordonnées de points permet une plus grande précision concernant la dimension physique d'un pixel.

Ainsi, en posant la variable `distance_physique` comme la dimension de référence, on a :

```
imshow(image)
[X,Y]=ginput(2);

distance_image=sqrt((X(2)-X(1))^2+(Y(2)-Y(1))^2);
dimension_pixel=distance_physique/distance_image;
```

La variable `dimension_pixel` représente la longueur d'un coté du pixel exprimé dans la même dimension physique que la distance de référence (le plus souvent en mm). Il s'agira donc de multiplier par la suite les longueurs exprimées en pixels par `dimension_pixel` et les aires par `dimension_pixel^2` pour obtenir des valeurs physiques.

Attention, le raisonnement précédent repose sur l'hypothèse que le quadrillage en pixels de l'image est carré, c'est à dire que deux longueurs de même dimension seront codés par le même nombre de pixels quelque soit leur direction.
Un moyen de s'en assurer est de filmer un carré et d'étudier l'image affichée

par Matlab. Une comparaison des cotés du carré affiché nous renseignera sur la véracité de l'hypothèse précédente.

Visualisation d'un carré

A l'aide de la fonction `ginput`, on extrait les coordonnées des extrémités du carré. On peut ensuite comparer les différents côtés. **On constate alors un écart d'environ 10% entre hauteurs et largeurs**. La figure C.8 illustre ce phénomène de manière légèrement amplifiée.

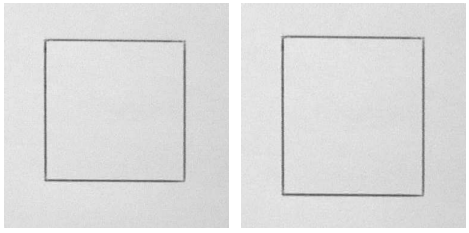

FIG. C.8 – Image filmée par la caméra (gauche) et image affichée par Matlab (droite).

Le nombre de pixels attribués à l'image dépendant de la caméra, il est certain que le quadrillage de la caméra n'est pas carré puisque cette dernière attribue plus de lignes que de colonnes pour coder un carré. **Il est donc nécessaire d'introduire une correction dans le code de calcul.**

Correction du code de calcul

Nous allons pour cela étudier le cas simple présenté en figure C.9 : la caméra associe en chaque point de l'image filmée un pixel rectangulaire de largeur l et de hauteur $0.5\,l$. L'image filmée est un carré de 4 cm^2.

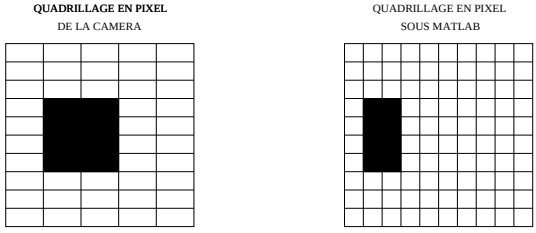

FIG. C.9 – Attribution des pixels par la caméra pour un carré (gauche) et résultat correspondant sous Matlab (droite).

Le carré est donc codé sous Matlab par une matrice $4*2$, c'est à dire par 8 pixels. Le caractère rectangulaire du quadrillage implique que l'on distingue

la dimension physique verticale d'un pixel (c'est à dire la longueur en mm d'un côté vertical : 5 mm) de la dimension horizontale (10 mm). Ainsi, chaque pixel représente une aire de $5 * 10 = 50$ mm, ce qui fait bien 4 cm^2 pour 8 pixels.

Il est donc nécessaire de distinguer la dimension physique verticale d'un pixel de sa dimension horizontale. La détermination de l'une des deux sera suffisante si l'on connait le rapport entre les deux.

On pose `dimension_pixel_vert` et `dimension_pixel_hor` respectivement la dimension physique d'un coté verticale et horizontale du pixel.
Dans cet exemple, on a `dimension_pixel_vert=0.5*dimension_pixel_hor`.

Nous savons qu'il n'est pas nécessaire de déterminer les deux dimensions si le rapport des deux est connu. Cependant, il est intéressant d'étudier leur obtention respective car chacune implique une correction différente concernant la distance de référence (`distance_physique`). En effet, cette dernière doit correspondre à une distance sur l'image exprimée en longueur de pixel (`distance_image`), qui ne sera pas la même selon que l'on cherche à déterminer `dimension_pixel_vert` ou `dimension_pixel_hor`.

Le plus simple est encore de retourner brièvement sur l'exemple de la figure C.9 : on choisit naturellement comme distance de référence le côté du carré (20 mm).

- Si on fait correspondre à cette distance un coté vertical, on aura une distance égale à 4 longueurs de pixel, le rapport `distance_image/distance_physique` étant alors égal à `dimension_pixel_vert`, c'est à dire 5 mm.
- Par contre, si on fait correspondre à cette distance un coté vertical, on aura une distance égale à 2 longueurs de pixel, le rapport `distance_image/distance_physique` étant alors égale à `dimension_pixel_hor`, c'est à dire 10 mm.

Bien évidemment, on cherche une méthode qui accepterait toute distance comme référence quelque soit sa direction et non pas nécessairement une distance verticale ou horizontale. Ceci est envisageable à condition de modifier une de deux composantes de la variable `distance_image` selon la dimension physique du pixel que l'on souhaite déterminer.

1. **La détermination de la dimension physique verticale** :

```
imshow(image)
[X,Y]=ginput(2);

distance_image=sqrt((2*(X(2)-X(1)))^2+(Y(2)-Y(1))^2);
```

```
dimension_pixel_vert=distance_physique/distance_image;
```

En ce qui concerne les calculs de distances, il sera nécessaire de multiplier les composantes verticales par `dimension_pixel_vert` et les composantes horizontales par `dimension_pixel_vert*2`.
Pour les calcules d'aires on multipliera les valeurs en pixels obtenues par `dimension_pixel_vert^2*2`.

2. **La détermination de la dimension physique horizontale** :

```
imshow(image)
[X,Y]=ginput(2);

distance_image=sqrt(((X(2)-X(1)))^2+(0.5*(Y(2)-Y(1)))^2);
dimension_pixel_hor=distance_physique/distance_image;
```

En ce qui concerne les calculs de distances, il sera nécessaire de multiplier les composantes verticales par `dimension_pixel_hor*0.5` et les composantes horizontales par `dimension_pixel_hor`.
Pour les calcules d'aires on multipliera les valeurs en pixels obtenues par `dimension_pixel_hor^2*0.5`.

Remarque

La méthode détaillée au paragraphe C.3.2 implique de connaître le rapport des dimensions physiques d'un pixel de l'image. Ce rapport est propre au type d'appareil utilisé pour faire l'acquisition de l'image. Ainsi, pour la caméra DV utilisé au laboratoire, on estime ce dernier à 1,1. Un moyen de le déterminer est de filmer un carré comme nous l'avons présenté au paragraphe C.3.2 et de faire le rapport entre le nombre de pixels attribué à la hauteur du carré et le nombre de pixels attribué à sa largeur.
Le carré présente l'avantage de permettre la vérification immédiate de la correction puisque l'aire que l'on doit trouver est parfaitement connue.

C.3.3 Calcul de la largeur et de la longueur de l'ouverture

Le paragraphe C.3.2 nous indique qu'il est relativement aisé d'obtenir la distance entre deux points quelconques de l'image. Le plus simple serait donc d'utiliser la fonction `ginput` et de selectionner les distances qui nous intéressent directement sur l'image (la figure C.10 présente un exemple d'image d'anche double, on constate qu'il est tout à fait possible de cliquer sur les extrémités qui nous intéressent).

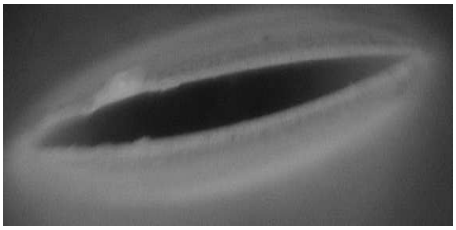

FIG. C.10 – Image filmée de face d'une anche double partiellement ouverte.

On prendra soin cependant de bien tenir compte du facteur de correction estimé au paragraphe C.3.2.

C.4 Pour une séquence d'images

L'idée est bien évidemment de créer une boucle qui calculerait sur une séquence d'images l'aire de l'ouverture de l'anche, sa longueur et sa largeur. On détaillera uniquement dans ce chapitre l'aspect calculatoire de cette démarche, le code en lui-même étant explicité au chapitre C.7.
Ainsi, on supposera qu'il est possible à l'utilisateur d'établir une boucle sur la séquence où seraient répétées systématiquement les mêmes opérations sur l'image. Tout le problème sera :

1. De définir des paramètres de calcul valables pour toutes les images de la séquence, que ce soit le seuil de binaristion, le type de filtrage de l'image et ses caractéristiques ou le choix de la tache.
2. De trouver une autre méthode de calcul pour déterminer la longueur et la largeur de l'ouverture puisque que l'on ne peut se permettre de les déterminer directement pour chaque image (cf paragraphe C.3.3), une séquence pouvant comporter jusqu'à plusieurs centaines d'images.

C.4.1 Les paramètres de calcul

On entend par paramètres de calcul, les paramètres de binarisation, de filtrage et de calcul de l'aire. L'intérêt étant de les définir de façon à ce qu'il soient valables pour toutes les images de la séquence.

La binarisation de l'image

Rien n'empèche à priori d'utiliser le même seuil de binarisation pour chacune des images de la séquence, à condition que celles-ci aient toutes été réalisées dans les mêmes conditions d'éclairage. On pensera notemment au

cas où l'on n'utilise pas une caméra DV classique mais une caméra à haute fréquence d'acquisition.

En effet, cette dernière peut être associée à un stroboscope dont la fréquence est réglée sur celle de l'acquisition. Cependant, un léger décalage au court de l'acquisition de ces deux fréquences peut entraîner une différence de contraste non négligeable entre deux images de la même séquence. La figure C.11 illustre ce propos, et les conséquence sur la binarisation des images : on peut y observer deux images d'une anche double amortie issue d'une même séquence d'images.

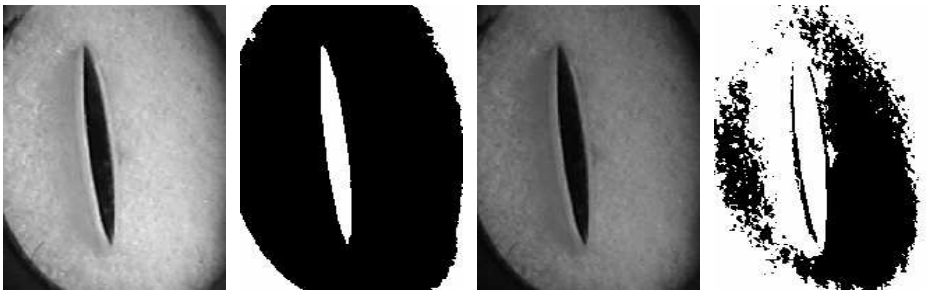

FIG. C.11 – Deux couples image/image binarisée inversée pour un seuil de 0,435.

Si l'on veut pouvoir utiliser le même seuil de binarisation pour chaque image de la séquence, il est nécessaire d'assurer un contraste équivalent pour chacune d'entre elles.

On propose deux méthodes qui visent à harmoniser les contrastes des différentes images. Ces deux méthodes ont en commun d'utiliser une image choisie par l'utilisateur qui servira de référence pour le reste de la séquence.

1. **L'utilisation des fonction `histeq()` et `imhist()`** : on choisit une image dont le contraste servira de référence pour le reste de la séquence. On extrait de l'image l'histogramme présentant la répartition des niveaux de gris présents à l'image à l'aide de la fonction `imhist()`, puis on fait correspondre le contraste de chacune des images de telle sorte que leur histogramme respectif s'étale sur le même domaine de niveau de gris.
   On choisira ici l'image de gauche de la figure C.11 comme référence :

   ```
   histograme=imhist(image_1);
   ```

   On peut visualiser figure C.12 l'histogramme de l'image 1 qui servira de référence pour l'image 2.

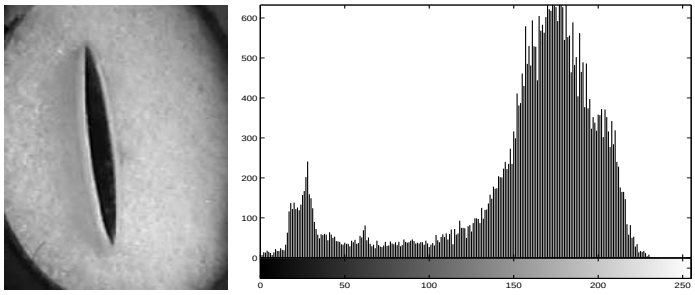

FIG. C.12 – L'histograme de la première image.

On applique la fonction `histeq()` à l'image 2 en utilisant l'histogramme de l'image 1 comme référence :

```
image_2=histeq(image_2,histograme);
```

La figure C.13 montre l'image 2 avant, puis après harmonisation des contrastes, ainsi que leur binarisation inversée respective, toujours pour un seuil à 0,435.
On constate bien qu'il est possible de conserver le même seuil de binarisation pour l'image 1 et l'image 2 sans perdre d'information sur l'aire de l'ouverture de l'anche.

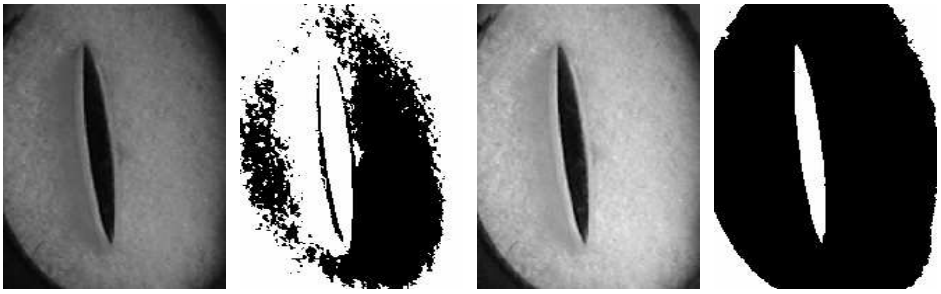

FIG. C.13 – Couple image/image binarisée inversée avant et après harmonisation du contraste.

On constate bien sur la figure C.14 que Matlab va modifier le domaine de niveau de gris de l'image 2 pour le faire correspondre à celui de l'histogramme de référence.

2. **L'harmonisation à partir d'une région de référence** : on choisit une région de l'image 1 que l'on prendra comme référence (par exemple, un rectangle sur la mousse qui amortit l'anche), et l'on modifie le contraste de l'image 2 de telle sorte que la valeur moyenne du niveau

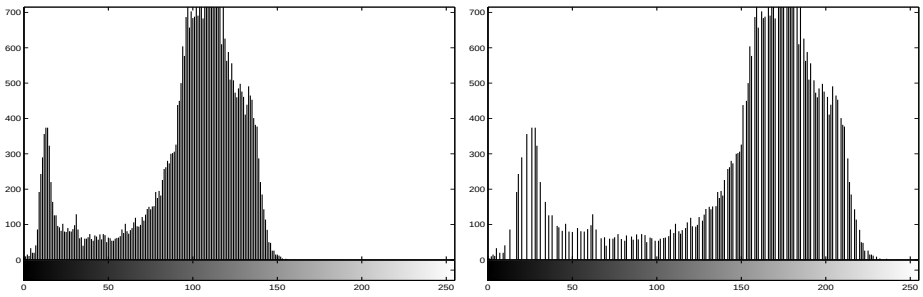

FIG. C.14 – Histograme de l'image 2 avant et après harmonisation du contraste

de gris de la même région de l'image soit égale à la valeur moyenne du niveau de gris de la région de référence.

On définit un rectangle de l'image 1 qui sera la région de référence (cf figure C.15). Un moyen pratique de la définir est de récupérer les limites des axes de la figure après avoir zoomé sur la région. On obtient ainsi un vecteur de 4 éléments noté U contenant les limites de la région en abscisse et ordonnée.

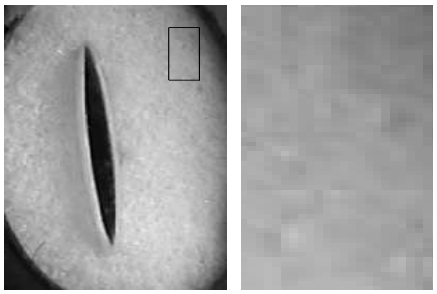

FIG. C.15 – Selection de la région de référence/ zoom sur la région.

On extrait de l'image la région de référence :

```
u=floor(U);
region=image_1(u(3):u(4),u(1):u(2));
```

On calcule alors la valeure moyenne en niveau de gris (entre 0 et 1) de la région : `moyenne=mean(mean(region));`
Il s'agit ensuite de calculer la valeur moyenne pour la même région de l'image 2 :

```
region_2=image_2(u(3):u(4),u(1):u(2));
moyenne_2=mean(mean(region_2));
```

Puis de modifier le contraste de l'image 2 :
`image_2=immultiply(image_2,moyenne/moyenne_2);`

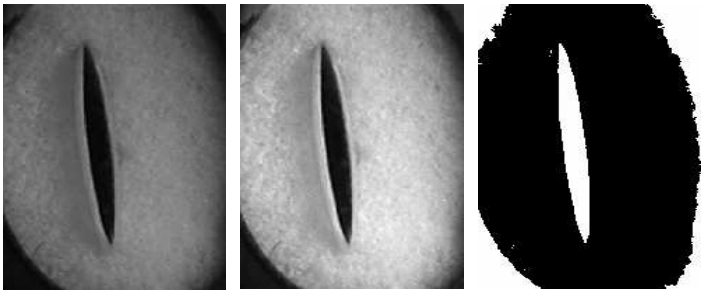

FIG. C.16 – Image 2 avant et après harmonisation/ image binarisée inversée après harmonisation

L'harmonisation des contrastes entre toutes les images de la séquence permet de retenir une unique valeur de seuil binarisation.

Remarque : Il est difficile de faire une comparaison entre les deux méthodes d'harmonisation proposées. On remarquera cependant que la méthode reposant sur l'utilisation de la fonction `histeq` apparaît comme plus "propre". En effet, et sans qu'aucune explication viennent justifier ce propos, le résultat de l'harmonisation par région donne l'impression d'avoir perdu en netteté concernant les images modifiées, le contraste semblant plus grossier. Cette remarque est cependant tout à fait subjective et en aucun cas basée sur des considérations mathématiques. On peut supposer cependant que la méthode d'harmonisation par région entraîne probablement la saturation du contraste dans certaines régions.

Toutefois, on peux se poser la question de savoir s'il n'existerait pas des cas où l'harmonisation par région serait préférable à l'utilisation de la fonction `histeq`. On pensera notamment au cas où l'aire d'ouverture joue sur le contraste de l'image : en effet, lorsque l'anche est éclairée de l'intérieur, sa fermeture peut entraîner une diminution de luminosité. Il peut en résulter que le seuil de binarité défini pour les grandes ouvertures soit inadapté pour de faibles ouvertures. Dans ce cas précis, il semble préférable d'harmoniser les images de la séquences à partir d'une région de référence indépendante de l'ouverture (comme par exemple la mousse entourant l'anche double sur la figure C.11) que plutôt d'utiliser la fonction `histeq` qui agit sur l'image dans sa globalité.

Le filtrage.

Nous avons repertorié au paragraphe C.2.2 trois fonctions de filtrage :

1. La fonction `imopen`.
2. La fonction `imclose`.
3. La fonction `imfill`.

Si l'utilisation des deux premières est compatible avec une boucle sous Matlab, la fonction `imfill` pose le problème des points de l'image en paramètres d'entrée que l'on ne peut se permettre de redéfinir pour chaque image de la séquence.

La fonction `imfill` permet cependant de stocker les points utilisés dans une variable afin de pouvoir les réutiliser par la suite :

```
[image_binarisee_1,points]=imfill(image_binarisee_1);
image_binarisee_2=imfill(image_binarisee_2,points);
```

les figures C.17 et C.18 illustrent un exemple sur deux images présentant une aire d'ouverture sensiblement différente.

FIG. C.17 – Image 1 avant et après utilisation de la fonction imfill pour un point en bas à gauche.

On peut donc définir des paramètres de filtrage que l'on peut utiliser au sein d'une boucle sous Matlab. Concernant la fonction `imfill`, on prendra soin de choisir des points relativement éloignés de la tache principale afin de ne pas la convertir sur certaines images de grandes ouvertures en une tache noire. Evidemment, cela suppose qu'il n'existe pas un grand décalage spatial de l'anche entre les images de la séquence, c'est à dire que l'anche ne bouge pas au court de la séquence.

Remarque : Toujours dans le but de réduire au maximum le nombre total de taches sur l'image binarisée et sous l'hypothèse du faible décalage spatial entre les images d'une même séquence, on soulignera l'intérêt du zoom.

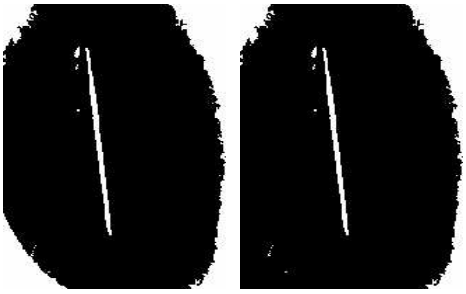

FIG. C.18 – Image 2 avant et après utilisation de la fonction imfill à partir du point utlisé pour l'image 1.

Dans une démarche analogue à celle du paragraphe C.4.1 pour définir une région de l'image, il est possible de réduire l'image à une zone contenant nécessairement l'ouverture de l'anche et ainsi gagner en temps de calcul.

Distinction de la tache principale

Le paragraphe C.3.1 nous indique qu'il est nécessaire de distinguer l'identifiant attribué à la tache qui nous intéresse par la fonction `bwlabel` pour pouvoir déterminer son aire en nombre de pixels.
Cependant, rien n'indique que cet identifiant sera le même pour chaque image de la séquence. En effet Matlab attribue de manière croissante un entier au fur et à mesure qu'il détecte une tache sur l'image binarisée. L'identifiant dépend donc de l'ordre des taches sur l'image, ordre qui diffère selon l'image.

Il s'agit donc de trouver une méthode qui va permettre de distinguer pour chaque image l'identifiant attribué à la tache qui nous intéresse.

En supposant que le décalage spatial entre deux images successives soit négligeable, on peut faire l'hypothèse que la position de la tache représentant l'ouverture de l'anche sur deux images successives de la séquence est approximativement la même.
L'idée est donc de s'appuyer sur la boucle de calcul en utilisant la détermination d'une tache pour trouver la tache de l'image suivante.

Pour cela, on va travailler à partir des centres de gravité des taches de l'image binarisée. On parlera de **centroïdes**. Le centroïde est accessible à l'aide de la fonction `regionprops()` (cf présentation au paragraphe C.1.3).

Afin d'illustrer la méthode choisie, on considère deux images successives de la séquence en supposant le centroïde de l'image 1 connu (noté `c`, il a la dimension d'une structure).

On peut alors chercher sur l'image 2 la tache dont le centroïde est le plus proche du centroïde de l'image 1 et en déduire que cette tache corresponds à l'ouverture de l'anche, et ainsi accéder à l'aire en pixels de l'ouverture sur l'image 2 :

```
[taches_2,n_2]=bwlabel(image_binarisee_2,4);

C=regionprops(taches_2,'Centroid');
A=regionprops(taches_2,'Area');

for i=1:1:n_2
   distance(i)=(C(i).Centroid(2)-c.Centroid(2))^2+(C(i).Centroid(1)- ...
                                                  c.Centroid(1))^2;
end

[X,index_2]=min(distance);
Aire_image_2=A.(index_2).Area;
```

Il est nécessaire d'initialiser la boucle. On peut pour cela déterminer le centroïde de la première image de la séquence en s'appuyant sur la démarche présentée au paragraphe C.3.1 : on détermine l'identifiant de la tache qui nous intéresse et on utilise la fonction `regionprops()` afin de connaître son centroïde.

On remarquera que cette méthode peut s'avérer inefficace lorsqu'une tache autre que celle qui nous intéresse présente un centroïde très proche du centroïde de la tache de l'image précédente. C'est le cas par exemple sur l'image présentée en figure C.19 et le résultat de sa binarisation.

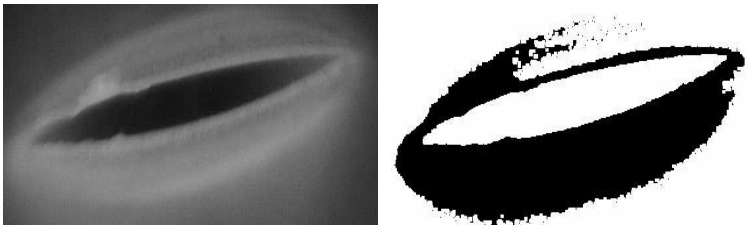

FIG. C.19 – Exemple de deux taches aux centroïdes proches

En effet, le bord de l'image (de couleur blanche) constitue pour Matlab une tache dont le centroïde est proche du centre de l'image et donc de celui de la tache qui représente l'ouverture de l'anche. Il existe donc un risque non négligeable que ce centroïde soit celui conservé lors du calcul et que l'aire ainsi déterminée ne corresponde pas à l'aire d'ouverture sur l'image.

Une solution proposée est d'utiliser la fonction imfill appliquée en un point de la tache du bord afin d'éviter toute confusion possible.

C.4.2 Le calcul de la longueur et de la largeur de l'ouverture

Il s'agit d'établir une méthode de calcul satisfaisante et compatible avec une boucle afin de déterminer la longueur et la largeur de l'ouverture de l'anche.

Une méthode fait l'objet du chapitre C.6.

C.5 Cas particulier où l'anche est presque fermée

Le chapitre C.4 montre qu'il est possible de créer une boucle qui calcule l'aire de l'ouverture de l'anche sur une séquence d'images. **Cependant, il est nécessaire de souligner que la méthode décrite peut s'avérer inefficace lorsque l'anche est presque refermée.**

En effet, la méthode a l'inconvénient de ne sélectionner qu'une unique tache qu'elle considérera comme l'ouverture. Or la figure C.20 montre que lorsque l'anche est presque fermée, son ouverture peut être représentée par plusieurs taches distinctes.

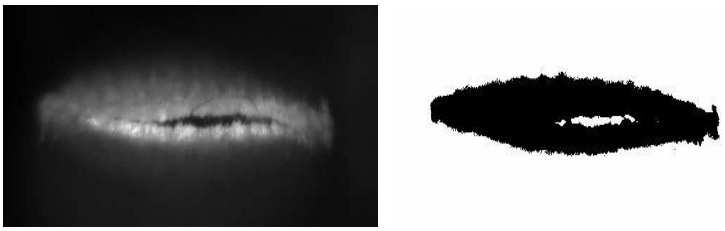

FIG. C.20 – Exemple d'anche presque fermée.

L'idée est de considérer l'ensemble des taches qui représentent l'ouverture de l'anche et de sommer leur aire respective afin d'obtenir une estimation plus juste de l'aire d'ouverture.

Pour cela, nous allons considérer deux images distinctes de la séquence :
- l'image 1 correspondant à une aire d'ouverture suffisament grande pour qu'elle soit représentée par une unique tache.
- l'image 2 correspondant à une anche presque fermée du type de celle présentée en figure C.20.

L'idée est de créer une image filtre contenant uniquement la tache qui représente l'ouverture sur l'image 1. Ceci est possible une fois l'identifiant attribué à la tache connu. Ainsi, en posant `index_1` cet identifiant et `taches_1` la matrice crée par l'opération `bwlabel`, on peut créer une image `filtre_1` contenant uniquement la tache désirée (cf figure C.21) :

```
filtre_1=(taches_1==index_1);
```

FIG. C.21 – Image 1 binarisée et image filtre correspondante.

On peut alors multiplier terme à terme les matrices `filtre_1` et `image_binarisee_2` afin d'isoler l'ensemble des taches qui représentent l'ouverture de l'anche sur l'image 2 (cf figure C.22) :

```
ensemble_taches=filtre_1.*double(image_binarisee_2);
```

FIG. C.22 – Taches de l'image 2 isolées.

Puisqu'un pixel blanc correspond à une valeur de 1 et un noir à une valeur de 0, il suffit de sommer l'ensemble des éléments de l'image obtenue pour accéder aux nombre de pixels qui constituent l'ensemble des taches :

```
Aire_image_2=sum(sum(ensemble_taches));
```

C.6 Méthode de calcul des dimensions de l'ouverture

A partir des informations que Matlab fournit sur les taches (par l'intermédiaire de la fonction `regionprops()`), voyons comment l'on peut obtenir une estimation satisfaisante de la longueur et de la largeur de la taches.

C.6.1 Ce que propose Matlab

La fonction `regionprops()` va associer à chacune des taches détectées une ellipse qui possède le même moment centré d'ordre 2 : la répartition des points par rapport au centre de gravité de l'ellipse est analogue à la répartition des pixels de la tache par rapport à son centroïde (cf figure C.23).

FIG. C.23 – Exemple de tache et d'ellipse associée.

On peut accéder aux valeurs (en pixels) des deux axes de l'ellipse (par la commande `regionprops(taches,'Majorlength','Minorlength')`). Il s'agit de savoir si ces distances sont des estimations satisfaisantes de la longueur et de la largeur de la tache.

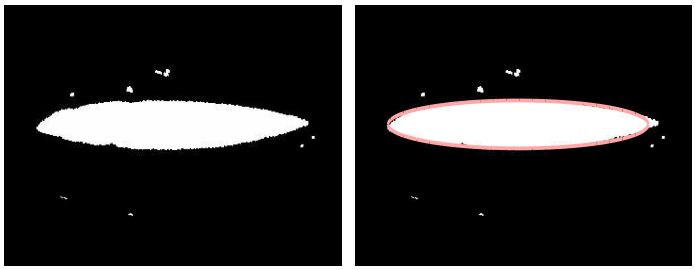

FIG. C.24 – Exemple de tache et ellipse associée.

La figure C.24 montre la superposition d'une tache et de l'ellipse associée par Matlab. Il est évident que l'on ne peut se satisfaire des dimensions de l'ellipse comme valeur de la longueur et de la largeur de la tache.

C.6.2 Solution proposée

L'idée est de tracer les deux axes de l'ellipse sur l'image binarisée et de compter le nombre de pixels appartenent à la fois :

- à la tache et au grand axe de l'ellipse pour déterminer la longueur de l'ouverture.
- à la tache et au petit axe de l'ellipse pour déterminer la largeur de l'ouverture.

cf figure C.25.

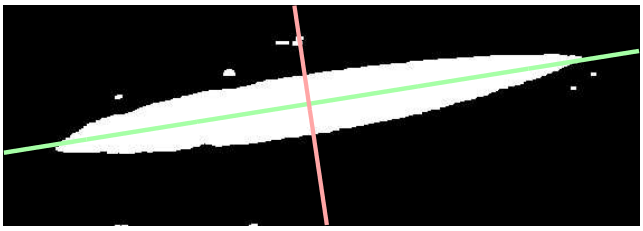

FIG. C.25 – Tracer des axes de l'ellipse sur la figure.

On peut en effet tracer ces deux axes puisque nous connaissons un point leur appartenant (le centroïde de la tache) et leur orientation. En effet, la commande `regionprops(taches,'Orientation')` permet d'accéder à l'angle en degrés que fait le grand axe de l'ellipse avec l'axe des abscisses (on en déduira aisément l'orientation du petit axe).

On distingue plusieurs étapes dans la méthode :

1. Déterminer le nombre de **pixels de l'image appartenant aux deux axes de l'ellipse**.
2. Déterminer le nombre de ces pixels **appartenant** effectivement **à la tache** représentant l'ouverture.
3. Convertir ce nombre de pixels en une distance.

C.6.3 Etape 1 : à partir du cas général

Si superposer une droite et une matrice n'a pas réellement de sens, l'idée de la méthode est de convertir l'axe en pixels, c'est à dire de déterminer les pixels de l'image appartenant à l'axe sur l'image présentant la superposition de la tache et de l'axe.
Afin d'illustrer au mieux les difficultés que la méthode engendre, **nous allons étudier le cas général d'une droite passant par un point de coordonnées connues et d'orientation θ.**

Avant cela, il est cependant nécessaire d'introduire un rappel concernant les différents types de coordonnées.

Rappel concernant les coordonnées

On distingue essentiellement les coordonnées cartésiennes (pour la droite) des coordonnées matricielles (pour les pixels). Ainsi, on cherche à déterminer à partir des points de la droite codés en abscisse et en ordonnée les coordonnées des pixels équivalents sur l'image, codés eux en indice de ligne puis de colonne.
Il semble donc pratique de considérer pour la droite un repère cartésien dont l'origine se situe en haut à gauche de l'image et dont l'axe des ordonnées soit orienté vers le bas afin d'avoir une relation directe entre abscisse et indice de colonne, ordonnée et indice de ligne.

On remarquera alors que :

1. dans un repère inversé, on devra tracer une droite d'orientation $-\theta$.
2. le point par lequel passe la droite doit être exprimé dans le repère inversé. Dans le cas qui nous intéresse, c'est à dire concernant le centroïde, nous constatons aisément que c'est le cas. En effet, on pourra vérifier qu'une tache composée d'un unique pixel situé à la i^{eme} ligne, j^{eme} colonne, possède un centroïde de coordonnées (j, i) selon la fonction `regionprops`.

On travaille donc dans le repère inversé. On pose (x_0, y_0) les coordonnées du point par lequel passe la droite et $-\theta$ son orientation.
On travaille sur une image de dimension l lignes et c colonnes.

L'équation cartésienne de la droite s'écrit :

$$-tan(\theta) = \frac{y - y_0}{x - x_0} \tag{C.1}$$

Il est nécéssaire cependant de distinguer deux cas concernant l'orientation de la droite. Cette distinction sera expliquée au paragraphe C.6.6.

Cas où $0° \leq |\theta| \leq 45°$

Dans ce cas, **on cherche pour chaque colonne de l'image, l'indice de ligne du pixel** le plus proche de la droite.
Pour cela, on va poser une matrice ligne représentant l'indice de chacune des colonnes de l'image (donc de 1 à c) et associer à chacun de ces indices la

valeurs sur la droite. L'entier le plus proche de cette valeur sera considéré comme l'indice de ligne du pixel recherché.

On travaille ici à partir d'une image uniformément noire de dimension 400×500. Le point par lequel passe la droite sera le centre de l'image $(250, 200)$ et son orientation de $\frac{\pi}{6}$.

```
image=zeros(400,500);
x=[1:1:500];
x0=250;
y0=200;
y=y0-tan(pi/6)*(x-x0);
coord=[round(y)' x'];
```

La matrice `coord` contient les coordonnées des pixels de l'image situés au dessous de la droite. Une manière de les visualiser est de les convertir en pixels blancs :

```
for i=1:1:500
image(coord(i,1),coord(i,2))=1;
end
```

La figure C.26 présente le résultat :

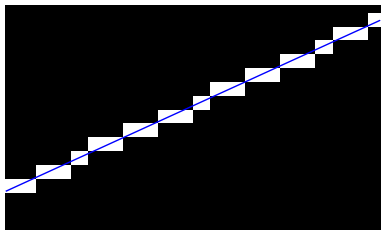

FIG. C.26 – Pixels situés sous la droite d'orientation $\frac{\pi}{6}$ (zoom)

Cas où $45° < |\theta| \leq 90°$

On travaille toujours sur la même image, mais pour une orientation de la droite de $\frac{\pi}{3}$. Dans ce cas, c'est **pour chaque ligne que l'on va chercher l'indice de colonne du pixel** situé sous la droite, et ce, selon une démarche analogue au paragraphe C.6.3 :

```
image=zeros(400,500);
y=[1:1:400];
x0=250;
y0=200;
x=x0-(1/tan(pi/3))*(y-y0);
coord=[y' round(x)'];
```

La figure C.27 présente le résultat :

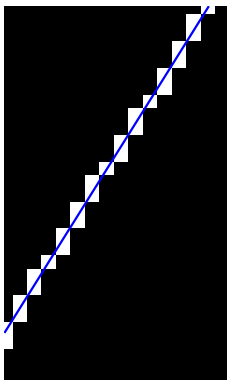

FIG. C.27 – Pixels situés sous la droite d'orientation $\frac{\pi}{3}$ (zoom)

Attention

La méthode proposée pour déterminer les indices des pixels appartenant à l'axe n'exclut pas d'obtenir des indices sans correspondance avec l'image (des valeurs négatives ou supérieures aux dimensions de l'image) que l'on ne pourra pas utiliser. Ainsi, la figure C.28 présente un cas d'orientation $0° \leq |\theta| \leq 45°$ où l'axe "sort" de l'image par les bordures horizontales. Ainsi,

- pour les premiers indices de colonne, les indices de ligne déterminés par l'opération `y=y0-tan(`θ`)*(x-x0)` seront supérieurs aux nombre total de de lignes de l'image (partie rouge de la figureC.28).
- pour les derniers indices de colonne, les indices de ligne déterminé par l'opération `y=y0-tan(`θ`)*(x-x0)` seront négatifs (partie verte de la figure C.28).

Il convient donc de modifier la matrice `coord` afin de ne garder que les valeurs correspondant à des indices de pixels existants.

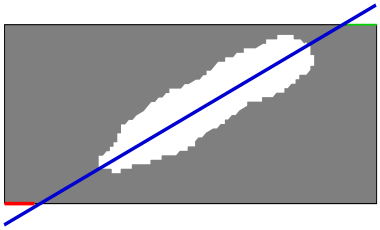

FIG. C.28 – Cas d'un axe qui "sort" de l'image par les bordures horizontales.

C.6.4 Etape 2 : déterminer quels pixels appartiennent à la tache

Le paragraphe C.6.3 nous indique comment isoler l'ensemble des pixels de l'image situés sous l'axe majeur de l'ellipse, mais aussi sous l'axe mineur en considérant une orientation de $\frac{\pi}{2} \pm \theta$.

L'étape 2 a pour but de déterminer le nombre de ces pixels appartenant effectivement à la tache qui code l'ouverture.
Un moyen est de superposer les axes à une image contenant uniquement la tache qui nous intéresse. Pour cela, on crée une image `filtre` selon la méthode vue au chapitre C.5 (cf figure C.29) :

```
filtre=(taches==index);
```

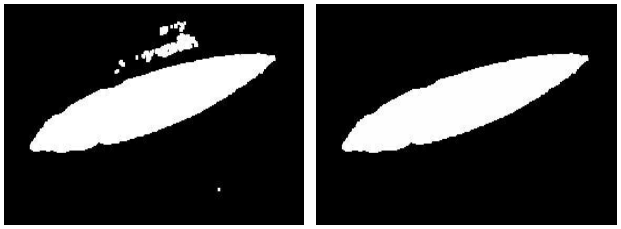

FIG. C.29 – Image binarisée et image filtre

A partir du filtre, il est aisé de déterminer combien de pixels situés sous un axe appartiennent à la tache.
En effet, le paragraphe C.6.3 nous indique comment créer une matrice `coord` contenant les coordonnées des pixels situées sous un axe. On peut alors créer une matrice ligne contenant les valeurs de ces pixels :

```
for i=1:1:length(coord)
    val_pixels(i)=filtre(coord(i,1),coord(i,2));
end
```

Les pixels appartenant à la tache contiennent la valeur 1, les autres 0. Ainsi la somme de ces valeurs donne le nombre de pixels communs à la tache et à l'axe :

```
nb_pixels=sum(val_pixels);
```

C.6.5 Etape 3 : en déduire une distance

Le nombre de pixel ainsi déterminé corresponds à la projection en pixels de la distance recherchée. Cette projection est

- horizontale si l'on est dans le cas : $0° \leq |\theta| \leq 45°$.
- verticale si l'on est dans le cas : $45° < |\theta| \leq 90°$.

On peut obtenir la valeur en dimension physique de cette projection en la multipliant par la dimension physique du pixel appropriée (cf chapitre C.3.2).

On obtient ensuite une estimation de la distance recherchée en multipliant la projection

- par $\frac{1}{cos(\theta)}$ pour le cas $0° \leq |\theta| \leq 45°$.
- par $\frac{1}{sin(\theta)}$ pour le cas $45° < |\theta| \leq 90°$.

C.6.6 Discussion

L'étape 1 va faire une distinction entre les images en fonction de l'orientatin de la tache principale, ainsi :

1. Pour une orientation $0° \leq |\theta| \leq 45°$, on va travailler à partir de tous les indices de colonne pour déterminer les indices de ligne.
2. Pour une orientation $45° < |\theta| \leq 90°$, on va travailler à partir de tous les indices de ligne pour déterminer les indices de colonne.

On peut expliquer ce choix par le fait que selon l'orientation, à un unique indice de ligne va correspondre plusieurs indices de colonne (axe d'orientation $\frac{\pi}{6}$, figure C.26) ou bien l'inverse (axe d'orientation $\frac{\pi}{3}$, figure C.27).

Afin de comprendre un peu plus le rôle de cette distinction, on observera la figure C.30 qui reprend les exemples du paragraphe précédent traités par la méthode qui ne leur est pas destinée.

On peut cependant se poser la question de la nécessité de cette distinction puisque de toute façon, nous travaillons à partir de projection. En effet, on peut se demander si il ne serait pas aussi efficace de ne plus faire cette distinction, c'est à dire par exemple d'utiliser systématiquement la méthode du cas $0° \leq |\theta| \leq 45°$, et de passer de la projection horizontale ainsi obtenue

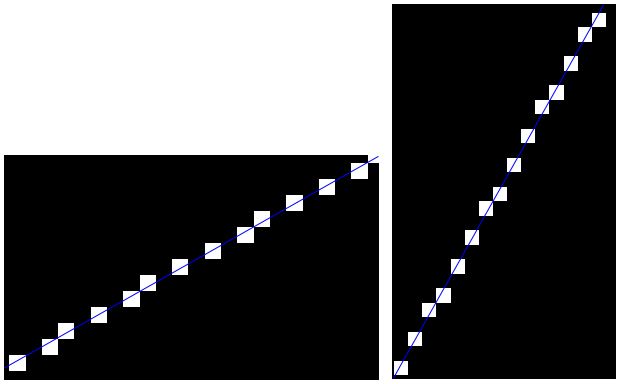

FIG. C.30 – Application de l'autre méthode pour une orientation de $\frac{\pi}{6}$ (gauche) et de $\frac{\pi}{3}$ (droite)

à une longueur par le facteur multiplicatif $\frac{1}{cos(\theta)}$. En effet, les exemples de la figure C.30 ne semble pas aller contre cette avis.

Un moyen de comprendre en quoi cette distinction est nécessaire est de se rapprocher des orientations 0 ou $\frac{\pi}{2}$. On devine aisément que plus l'on se rapproche des ces orienations et plus l'une des deux projections va tendre vers zero (la vertical pour $\theta = 0$ et l'horizontale pour $\theta = \frac{\pi}{2}$).

La figure C.31 présente la superposition de deux taches de même orientation $\frac{\pi}{2} - \epsilon$, et de dimensions sensiblement différentes. Si l'on applique à ces taches la méthode prévu pour le cas $0° \leq |\theta| \leq 45°$, nous constatons que la méthode ne retient que 5 pixels, dont seulement trois appartiennent aux taches. Ainsi, deux taches deux longueurs différentes présentent la même projection horizontale.

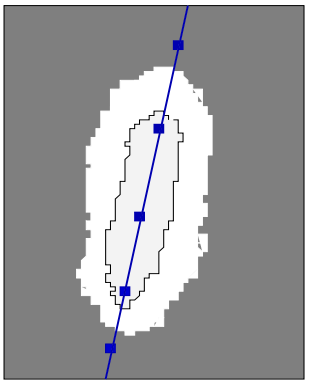

FIG. C.31 – Cas de deux taches presentant la même projection horizontale

Il est évident que l'important est donc de travailler avec la projection la plus grande, ce que la méthode de distinction permet. En effet, le fait de travailler à partir de nombre entier de pixel implique une erreur de `dimension_pixel_hor`$*\frac{1}{cos(\theta)}$ ou de `dimension_pixel_vert`$*\frac{1}{sin(\theta)}$ selon la méthode. Distinguer les deux types d'orientations, c'est s'assurer de minimiser cette erreur.

C.6.7 Validation de la méthode

Pour valider la méthode présentée, le plus simple est encore de la confronter à la méthode présentée au paragraphe C.3.3 qui repose sur l'utilisation de la fonction `ginput`.

On constate que l'on est aisément en dessous des 1% d'écart lorque l'on compare les deux méthodes, et ce sur toute la plage des orientations possibles.

On remarquera cependant que cet écart augmente lorsque l'on traite des dimensions faibles puisque l'on résonne à partir de nombre entier de pixels pour l'une (méthode du chapitre C.6), de coordonnées pour l'autre (méthode du chapitre C.3.3), la première perdant en précision.

Cependant, la méthode du chapitre C.6 a l'avantage d'être compatible avec un calcul sur une séquence d'images. On prendra soin afin de gagner en précision de traiter des images dont les dimensions ont un niveau de résolution convenable.

C.6.8 La fonction sous Matlab

Présentons la fonction `calcul_longueur` qui traduit tout ce qui a été exposé précédemment. Nous constatons qu'elle accepte en entrée la matrice résultante de l'opération `bwlabel`, l'indice de la tache dont le centroïde `c` est le plus proche du centroïde tampon et la matrice `filtre` qui correspond à la tache retenue isolée des autres comme nous l'avons vu au paragraphe C.6.4.

On remarquera la variable globale `correction` qui contient le coefficient de correction des dimensions physiques du pixel (cf paragraphe C.3.2). De plus, nous pouvons voir que la fonction tient compte de la remarque du paragraphe C.6.3 en supprimant les indices négatifs ou superieurs aux dimensions du filtre (cf variables `indlong` et `indlong2`).

```
function [long]=calcul_longeur(taches,index,filtre)
global c;
global correction;
```

```
Or=regionprops(taches,'Orientation');

size_filtre=size(filtre);

if(-45<=Or(index).Orientation)&(Or(index).Orientation<=45)
absc=[1:size_filtre(2)];
I=ones(size(absc));
ord=round(c.Centroid(2)*I-tan(Or(index).Orientation*pi/180)*(absc- ...
                                                    c.Centroid(1)*I));
indlong=find(ord>=0);
ord=ord(indlong);
indlong2=find(ord<=size_filtre(1));
ord=ord(indlong2);absc=absc(indlong2);
taille=size(indlong2);
coef=1/abs(cos(Or(index).Orientation*pi/180))*(correction);

else
ord=[1:size_filtre(1)];
I=ones(size(ord));
absc=round(c.Centroid(1)*I-1/(tan(Or(index).Orientation*pi/180))*(ord- ...
                                                    c.Centroid(2)*I));
indlong=find(absc>=0);
absc=absc(indlong);
indlong2=find(absc<=size_filtre(2));
ord=ord(indlong2);absc=absc(indlong2);
taille=size(indlong2);
coef=1/abs(sin(Or(index).Orientation*pi/180));

end

for j=1:1:taille(2)

  longeur(j)=filtre(ord(j),absc(j));
end
long=sum(longeur)*coef;
```

C.7 Le code de calcul pour une séquence d'images

Il n'est pas question ici de retranscrire de manière abrupte le code de calcul mais seulement de l'expliciter en reprenant les arguments développés dans les paragraphes précédents.

En effet, si nous avons démontré jusqu'ici qu'une boucle sous Matlab calculant l'aire d'ouverture de l'anche, sa longueur et sa largeur était envisageable, il reste cependant quelques points à éclaircir afin de la rendre réalisable.

C.7.1 Se reporter à une séquence d'images

Une séquence d'images n'est en soit qu'un répertoire contenant plusieurs images au même format (jpg, png, etc...) et au nom qui différe.
La question est de savoir comment on peut indiquer à Matlab d'aller chercher à chaque itération une image dans le répertoire en s'assurant que :

- Il respecte l'ordre de numérotation des images lorsque ces dernières en possèdent une.
- Il charge toutes les images, et ce une seule fois.

L'ordre de la séquence

Bien évidemment, une séquence d'image résulte souvent d'une acquisition dans le temps. L'ordre des images acquises, respectueux de la chronologie, a son importance.

Nous verrons que l'on peut conserver cet ordre à condition de bien nommer les images, c'est à dire que l'ordre alphanumérique des fichiers respecte l'ordre chronologique.

Charger les images

L'idée est de créer une structure contenant la liste des fichiers image.
Pour cela, on crée une variable `adresse` qui va contenir le chemin menant au répertoire contenant la séquence :

```
adresse='\home\sequence_images'
```

On crée alors une première structure contenant la liste des fichiers contenus dans le répertoire **sequence images**. Par exemple, pour un répertoire contenant quatre images en .jpg, ainsi qu'un fichier texte et une image en .png :

```
list_totale=dir(adresse);
list_totale=
    8x1 struct array with fields:
        name
        date
        bytes
        isdir
```

La liste contient donc huits éléments : les six fichiers présents ainsi que les chemins : '.' et '..'. On constatera que ces fichiers sont classés naturellement dans l'ordre alphanumérique :

```
liste_totale.name=
.
..
diver.txt
image.png
image_1.jpg
image_2.jpg
image_3.jpg
image_4.jpg
```

Il s'agit maintenant de créer une nouvelle liste contenant uniquement les fichiers qui nous intéressent. On définit une variable `fin` qui va servir de référence pour extraire les fichiers recherchés. Ici :

```
fin='.jpg';
```

Puis on parcourt la liste afin de détecter tous les fichiers dont le nom contient la variable `fin` que l'on stoque dans une nouvelle liste :

```
 liste_jpg=cellstr('');

 for i=1:length(liste_totale)
   if (length(strfind(liste_totale(i).name,fin))~=0)
    liste_jpg=vertcat(list_jpg,cellstr(liste_totale(i).name));
   else
   end
 end
```

On obtient ainsi une liste des noms de la séquence d'images, classés dans l'ordre alphanumérique. Si ce dernier respecte l'ordre chronologique de l'acquisition (ce qui est souvent le cas avec les caméras classiques), la séquence

sera traitée de manière chronologique.
On peut ainsi créer une boucle qui possède un nombre d'itérations égale à la taille de `liste_jpg` et qui vérifie :

```
image=imread([adresse liste_jpg{iteration}]);
```

C.7.2 Choix du type de calcul d'aire

Les chapitres C.4 et C.5 montrent qu'il est nécessaire de distinguer les cas où l'anche est suffisament ouverte, des cas où elle est presque fermée. A cela doit s'ajouter les cas où l'anche est tolalement fermée et ceux où l'on ne constate aucune tache sur toute l'image.

Il s'agit donc de mettre en place des tests afin de déterminer dans quel cas nous nous trouvons et d'y répondre de manière adéquate.

Dans la suite du paragraphe, nous nous situerons au moment où l'image de la séquence en cours a été binarisée et filtrée. L'opération qui suit directement est la distinction des taches par la fonction `bwlabel` :

```
[taches,n]=bwlabel(image_binarisee_filtree);
```

Si aucune tache n'est détectée sur l'image

C'est naturellement le cas le plus simple. On peut faire un test sur `n` : si ce dernier est nul, alors aucune tache n'est détectée ; l'aire, la longueur et la largeur de l'ouverture sont nécessairement nulles.

On suppose dorénavant `n` différent de zero.

Anche ouverte, presque fermée ou fermée ?

Nous allons voir que les cas "anche fermée" et "anche presque fermée" peuvent être traités de manière équivalente.

Il s'agit donc d'établir un test permettant de distinguer les cas " anche ouverte" et "anche fermée".

Pour cela, supposons que :
- Nous traitons la i^{eme} image de la séquence.
- L'image au rang $i-1$ correspondait à un cas "anche ouverte".
- Nous avons conservé dans les variables `taches_tampon` , `index_tampon` et `centroid_tampon` respectivement le résultat de l'opération `bwlabel`, l'identifiant et le centroïde de la tache représentant l'ouverture au rang $i-1$.

L'idée du test est de considérer une dimension de l'anche ouverte comme référence. Cette dimension doit bien évididemment être quasi invariante quelque soit l'aire d'ouverture lorsqu'il s'agit de cas "anche ouverte", c'est à dire une ouverture d'anche représentée par une unique tache.
On choisira comme référence la longueur de l'ouverture, ou plutôt une approximation, c'est à dire la taille de l'axe majeur de l'ellipse.

Ainsi, on a :

```
data=regionprops(taches_tampon,'All');
dist_ref=data(index_tampon).MajorAxisLength;
```

Pour déterminer dans quel cas se situe l'image de rang i, on va comparer la dimension de l'axe majeur de la tache la plus proche du centroïde de rang $i-1$ avec la distance de référence (voir aussi paragraphe C.3.1).
On doit donc déterminer la tache dont le centroïde est le plus proche du centroïde du rang précédent :

```
c=centroid_tampon;

data_i=regionprops(taches,'All');
C=data_i.Centroid;

for i=1:1:n_2
   distance(i)=(C(i).Centroid(2)-c.Centroid(2))^2+(C(i).Centroid(1)- ...
                                                  c.Centroid(1))^2;
end

[X,index_i]=min(distance);
```

Puis, on compare l'axe majeur de la tache trouvée avec la distance de référence. Sachant, qu'il est peu probable que ces dernières soient identiques, on fixe un rapport minimal de 80 % entre les deux pour considérer le rang i comme un cas "anche ouverte" :

1. Si la valeur de `data_i(index_i).MajorAxisLength/dist_ref` est supérieure ou égale à 0,8, alors on considère le rang i comme un cas d'anche ouverte. On pourra déterminer l'aire de l'ouverture, la longueur et la largeur par les méthodes décrites respectivement aux chapitres C.3 et C.6. Dans ce cas, et en vue du traitement de l'image du rang $i+1$, on redéfinie les valeurs tampons :

   ```
   taches_tampon=taches;
   index_tampon=index_i;
   centroid_tampon=C;
   ```

2. Dans le cas contraire, on considère le rang i comme un cas d'anche presque fermée. On accéde à la valeur de l'aire d'ouverture par la méthode décrite au chapitre C.5 en utilisant la matrice `taches_tampon` pour créer la matrice `filtre`. **Les variables tampons que l'on utilisera au rang $i+1$ restent celles définies au rang $i-1$.** Il n'existe cependant dans ce cas aucun moyen de donner une estimation des dimensions de l'ouverture qui aurait un sens, le calcul de ces dernières reposant sur l'idée d'une tache unique représentant l'ouverture de l'anche.

Il est nécessaire de faire plusieures remarques concernant le test décrit précédemment :

1. **La nécéssité d'initialiser la méthode**, et ce à partir d'une image d'anche bien ouverte. Nous avons vu au paragraphe C.4.1 les raisons pour lesquelles cette initialisation est nécessaire (nécessité d'un centroïde de référence) auxquelles s'ajoutent les définitions des variables tampons.
2. **La possibilité de définir une distance de référence commune** : en effet, on peut envisager de considérer une unique distance de référence pour le test et non la redéfinir à chaque itération. On pourrait ainsi la déterminer à partir de l'image qui servira à initialiser la boucle. Cela présente le double avantage de gagner en temps de calcul et d'éviter le risque que cette distance diminue fortement au court du calcul (dans le cas de plusieurs rapports successifs autour de 0.8), rendant le test moins efficace.
3. **L'équivalence des cas "anche fermée" et "anche presque fermée"**. En effet, le risque était de voir des taches sans rapport avec l'ouverture être comptabilisées car non éliminées par le filtre. Cependant, la boucle traitant les images de manière chronologique et redéfinissant tant que l'anche est ouverte une matrice `taches_tampon`, on peut se dire que le filtre utilisé pour calculer les aires d'ouverture faibles sera le plus serré possible, réduisant ainsi le risque de tenir compte de tache de bord sans rapport avec l'ouverture.

C.7.3 Le code sous Matlab

Voici le code sous Matlab qui traite une séquence d'images en tenant compte des différents problèmes présentés dans ce document. On a pris soin de choisir des noms de variables relativement explicites afin d'en améliorer la compréhension. Cependant, il convient de préciser le rôle de certaines d'entres elles, notamment celles dont le nom se termine par `_active` qui peuvent être

égale à 1 ou à 0. De manière générale si la variable `fonction_active` est égale 1, on applique à l'image une fonction de traitement ; dans le cas contraire, on ne fait rien.

On soulignera aussi le rôle de la vatiable `typ_image` qui indique si l'image a déjà de la dimension d'une matrice (cas des .bmp) ou si un passage en niveaux de gris est nécessaire.

On remarquera aussi que le calul de la longueur et de la largeur de l'anche se font par l'intermédiaire des fonctions `calcul_longeur` et `calcul_largeur` déjà présentées au paragraphe C.6.8.

```
global c correction dimensions;

dimensions=[];

%chargement des paramètres de traitement:
[fichier,adresse_parametres]=uigetfile('*.mat','choisir parametres');

load([adresse_parametres fichier]);


%chargement de la sequence:
liste=dir(adresse_sequence);
liste_sequence=cellstr('');

for i=1:length(liste);

if (length(strfind(liste(i).name,fin_nom))~=0)
liste_sequence=vertcat(liste_sequence,cellstr(liste(i).name));
else
 end
end

c=centroide_tampon;


%lancement de la boucle de calcul:
for j=2:length(liste_sequence);


  %initialisation des variables internes:
  choix=[];
```

```
image_anche=imread([adresse_sequence liste_sequence{j}]);


%passage niveaux de gris:
if(typ_image)
  else
image_anche=rgb2gray(image_anche);
end


%zone d'interet:
if(zone_active)
  image_anche=image_anche(zone_interet(3):zone_interet(4), ...
                          zone_interet(1):zone_interet(2));
else
end

%harmonisation:
   %par region:
   if(harmon_region_active)
   image_region=image_anche(region(3):region(4),region(1):region(2));
   moyenne=mean(mean(image_region));
   image_anche=immultiply(image_anche,moyenne_ref/moyenne);
   elseif(harmon_histo_active)
   image_anche=histeq(image_anche,histo_ref);
   end

%binarisation:
image_anche=im2bw(image_anche,seuil_bina);

%inversion binarite:
if(inversion_active)
image_anche=imcomplement(image_anche);
else
end


%filtrage:
if(rang_imop==rang_imclo)
elseif(rang_imop<rang_imclo)
image_anche=imopen(image_anche,element_imop);
```

```
   if(rang_imclo==3)
   else
   image_anche=imclose(image_anche,element_imclo);
   end

elseif(rang_imclo<rang_imop)
image_anche=imclose(image_anche,element_imclo);

   if(rang_imop==3)
   else
   image_anche=imopen(image_anche,element_imop);
   end

end


%imfill:
if(imfi_active)
 image_anche=imcomplement(imfill(imcomplement(image_anche),points));
else
end


%separation des taches:
[taches,nb_taches]=bwlabel(image_anche,4);

if(nb_taches==0)
  dimensions=[dimensions; 0 0 0];
else


  %extraction des centroides:
  cent=regionprops(taches,'Centroid');
  %extraction des axes majeurs:
  Lmaj=regionprops(taches,'MajorAxisLength');
  %extraction des aires:
  Air=regionprops(taches,'Area');

  %centroide le plus proche:
   for i=1:1:nb_taches
```

```
        choix(i)=(cent(i).Centroid(2)-c.Centroid(2))^2+ ...
                 (cent(i).Centroid(1)-c.Centroid(1))^2;
      end

      [x,index]=min(choix);

      %test sur la dimension majeure:
      if (Lmaj(index).MajorAxisLength>=ratio*distance_ref.MajorAxisLength)
        filtre=(taches==index);
        centroide_tampon=cent(index);
        c=centroide_tampon;

        longueur=calcul_longeur(taches,index,filtre);
        largeur=calcul_largeur(taches,index,filtre);

        dimensions=[dimensions;Air(index).Area longueur largeur];

        tache_tampon=taches;
        index_tampon=index;

      else
        filtre=(tache_tampon==index_tampon);
        image_filtre=filtre.*double(image_anche);
        dimensions=[dimensions;sum(sum(image_filtre)) 0 0];
      end

  end

end

%passage aux dimensions physiques:
dimensions=[dimensions(:,1)*dimension_pixel*correction dimensions(:,2) ...
            dimensions(:,3)]*dimension_pixel;
```

La figure C.32 présente un organigramme du programme. Vous y noterez que les flèches en pointillés traduisent les directions falcultatives.

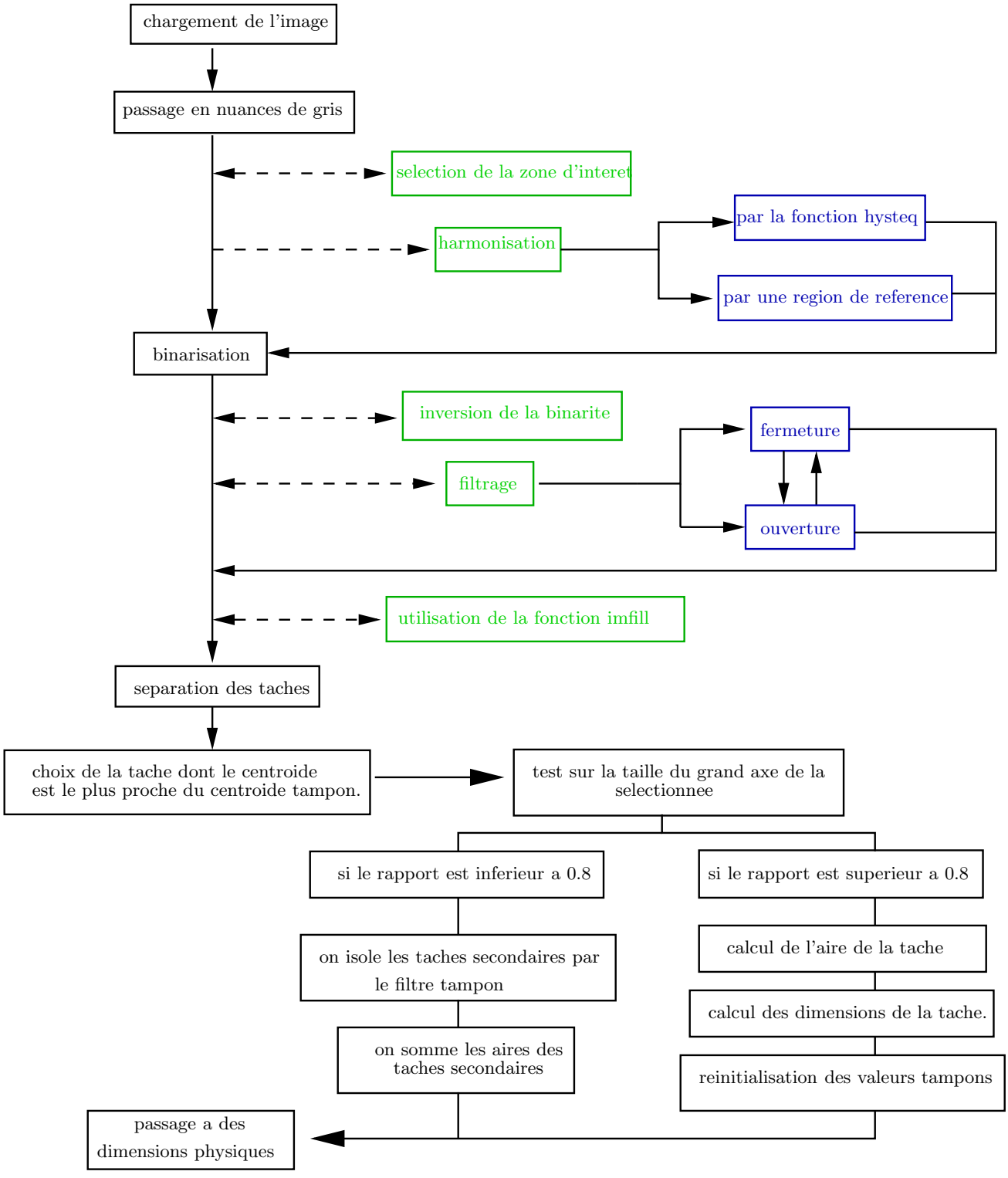

FIG. C.32 – Organigramme du code sous Matlab pour le traitement d'une séquence d'images.

C.8 Utilisation conjointe de la caméra

Les chapitres précédents présentent un moyen de calculer les dimensions de l'ouverture de l'anche double :

– à partir d'une image seule.
– sur une séquence d'images.

Le code constitue à priori un moyen d'accéder au dimensions de l'ouverture en post-traitement. Or, ces dimensions peuvent être lors de certaines expérimentations des paramètre déterminants qu'il est nécessaire de déterminer avant toutes mesures.

L'idée est donc de trouver un moyen de relier directement la caméra au code de calcul afin que ce dernier constitue un outil de réglage en temps réel lors de certaines expérimentations. Pour des raisons liées au fait que la fréquence d'acquisition de la caméra (environ 25 images par secondes) est très inférieure aux fréquences d'oscillation de l'anche, cette utilisation conjointe du code et de la caméra n'a de sens que pour des système au repos ou en régime statique.

On va chercher à s'appuyer sur les possibilités de contrôle sous Matlab de la caméra.

C.8.1 Contrôle de la camera sous Matlab

On sait commander la caméra à partir d'un terminal sous Linux par la fonction `dvgrab`. Or, rappelons que les commandes unix sont accessibles à Matlab par la fonction `unix` qui s'utilise selon :

```
[p,g]=unix('commande_unix');
```

Ainsi, il est tout à fait possible de commander la camera à partir de Matlab. La fonction `dvgrab` offre de nombreuses options qui vont nous permettre de distinguer deux types distincts d'utilisation conjointe du code et de la caméra. On rappelle que la commande `dvgrab` va enregistrer les images filmées par la caméra dans le répertoire de travail, et ce au format, à la fréquence et sur une durée imposés par l'utilisateur. Ainsi, on parlera d'acquisition d'images lorsqu'il y aura création d'un fichier image dans le répertoire de travail.

C.8.2 Calcul sur une image

Très proche dans son principe du calcul sur une image en post-traitement, l'intérêt est surtout d'essayer de réduire au maximum le temps entre l'acqui-

sition de l'image par la caméra et son traitement par Matlab, temps durant lequel les conditions de l'expérimentation pourraient varier.

On aimerait ainsi créer une fonction sous Matlab qui lancerait l'acquisition d'une image, suivie de son traitement instantané.

Pour cela, on va lancer une acquisition d'image par la camera de 60ms (option `--duration`), ce qui ne laisse le temps que d'enregistrer une unique image dans le répertoire de travail au format souhaité (option `--format`). On pensera à lui attribuer un nom afin que Matlab sache quelle image il doit lire :

```
[p,g]=unix('dvgrab --duration 60ms --format jpeg image');
     image_anche=imread('image.jpg');
```

On peut ensuite travailler sous Matlab à partir de l'image `image_anche` et effectuer un calcul de dimensions qui donne une indication directe et rapide du montage de l'expérimentation.

C.8.3 Calcul en temps réel

Le calcul sur une image présente certains inconvénients dans l'optique de donner une information rapide sur le montage :

1. La nécessité de redéfnir apres chaque acquisition d'image les paramètres du traitement.
2. L'impossibilité de régler les paramètre du montage et d'avoir un retour direct sur les dimensions.

Naturellement, nous en sommes arrivés à nous poser la question de savoir si un calcul en temps réel était envisageable, donnant en continu les informations sur chaque image acquise par la caméra, à l'image d'un calcul sur une séquence qui s'écrirait au fur et à mesure.

On peut évidemment discuter de l'appélation "temps réel", car le traitement d'une image suivi de la détermination de ses dimensions n'est pas vraiment instantané (en l'état actuel du code, pour une image traitée présentant moins de 20 taches, le calcul dure un peu moins d'une seconde). A cela, il convient de rajouter le fait que l'opération implique que Matlab et la fonction `dvgrab` fonctionnent en même temps sur la machine, ce qui risque d'alourdir un peu plus le temps de calcul.

Fixer les paramètres du calcul

C'est bien évidemment le premier problème à résoudre puisque les paramètres doivent être définis avant de pouvoir lancer un calcul en temps réel.

On pensera naturellement à ce qui a déjà été présenté aux paragraphes C.4, C.5, C.6 et C.7 concernant le calcul sur une séquence d'image.

On peux dire que le calcul en temps réel reposera sur les mêmes principes que le calcul sur une séquence d'images, que se soit concernant les paramètres de traitement ou la détermination des dimensions.

Ainsi, il est tout à fait envisageable de définir en amont du calcul en temps réel les paramètres du calcul (paramètres du traitement, variables tampons, etc...), et ce à partir d'une image du montage obtenue par la méthode décrite au paragraphe C.8.2.

La question est donc de savoir à partir de quelles images nous devons travailler, puisque nous ne pouvons pas réellement parler de séquence d'images dans le cas d'un calcul en temps réel.

Une séquence d'images dynamique

Nous avons vu que le traitement en temps réel des images sera quasi-identique au traitement d'une séquence d'images, à savoir une boucle répétant systématiquement les mêmes opérations sur des images différentes, redéfinissant au besoin des variables tampons.
Tout le problème est de savoir quelle image doit être considerée au début de chaque itération.

1. **Première idée : faire l'acquisition d'une image à chaque itération** selon la méthode présentée au paragraphe C.8.2. En effet, rien empèche de faire l'acquisition d'une durée de 60ms à chaque itération, écrasant systématiquement l'image creée au rang précédent. Cependant, **cette méthode présente l'inconvénient majeur d'être très lente.** En effet, la fonction `dvgrab` met un temps non négligeable à lancer une acquisition et à l'arréter. Il serait donc préférable d'opter pour une méthode permettant de lancer une seule fois la commande `dvgrab` et de travailler sur les images qui sont acquises au fur et à mesure.

2. **Seconde idée : faire une acquisition continue et travailler en parallèle sur la séquence dynamique ainsi créee**. Evidemment, cela sous-entend que la fonction `dvgrab` crée en continu des images qu'elle numérote dans l'ordre alphanumérique (cela est réalisable en retirant l'option `--duration`). Cette méthode est cependant difficilement réalisable car

(a) nous avons vu au paragraphe C.7.1 que la boucle peut se déplacer sur une séquence à condition de posséder dans une structure le nom de toutes les images présentes dans le répertoire indiqué, opération réalisée avant que la boucle de calcul ne soit lancée.

(b) dans l'hypothèse où le problème exposé précédemment serait levé, on peut se demander quelles seraient les conséquences d'une différence entre le temps séparant deux acquisitions d'image successive t_{acq} et le temps de traitement t_{calc} :
   - **si** $t_{acq} > t_{calc}$: on risque un retard entre les dimensions affichées par la boucle et l'ouverture effectivement filmée par la caméra.
   - **si** $t_{acq} < t_{calc}$: on risque de travailler avec des images qui n'existent pas.

L'argument (a) implique qu'il est délicat d'essayer de travailler à partir d'une séquence. En effet, cela nécessiterait de répertorier en début de chaque itération la totalité des images de la séquence (par l'utilisation de la fonction `dir`), ce qui représente une opération lourde en temps. Mais surtout, l'argument (b) souligne bien l'importance de traiter la dernière image acquise par la caméra et non de suivre une séquence, ce qui permetrait d'avoir un retard ou une avance du calcul sur l'acqusisition systématiquement réduit. On remarquera à ce sujet que l'utilisateur possède un certain contrôle sur le temps t_{acq} puisque l'on peut imposer la fréquence d'acquisition (plafonnée à 25 images par seconde).

3. **Troisième idée : faire une acquisition continue sur une image qui s'écrase et se régénère.** Cela est rendu possible par l'utlisation de l'option `--overwrite` de la fonction `dvgrab`. Ainsi, le calcul s'éffecturait systématiquement sur la même image à chaque itération. Les problèmes d'avance/retard serait réglé puisque :
   - **si** $t_{acq} > t_{calc}$: le code traite systématiquement la dernière image acquise par la caméra, les images acquises entre les deux itérations n'étant pas traitées.
   - **si** $t_{acq} < t_{calc}$: le code traite l'image même si cette dernière ne change pas, et ce jusqu'à ce qu'une nouvelle image soit acquise.

   Bien que réalisable, cette idée présente un inconvénient non négligeable : on risque de travailler à partir de fichiers image partiellement écrits, ce qui conduirait à des résultats incohérents. En effet, l'écrasement d'un fichier n'étant pas instantané, il existe des laps de temps durant lesquels le fichier ne sera que partiellement fini. C'est durant ces laps de temps qu'il faut empécher le code de considérer le fichier et de le traiter (cf figure C.33).

FIG. C.33 – image originale (gauche), exemple d'images partiellement écrites (milieu et droite).

Il est délicat de savoir exactement de quelle manière Matlab va réagir face à ces fichiers incomplets lors de l'appel de la fonction `imread` : on a observé que le plus souvent, l'image va être accéptée par Matlab qui va cependant indiquer par un message d'attention que l'image n'est pas complète. Il existe plusieurs types de messages renvoyés par Matlab ce qui sous-entend qu'il existe plusieurs types de fichiers incomplets. Voici un exemple de message envoyé à l'utilisateur :

```
Warning: Premature end of JPEG file
```

On notera cependant qu'il existe des cas où l'image est refusée par Matlab qui va alors interrompre le calcul et ce, sans que l'on ne comprenne pourquoi.
Mais l'important n'est pas tant de comprendre mais surtout d'éviter que les images incorrectes efffectivement lues par Matlab soient traitées par la suite, car elles donneront à coup sur des résultats numériques faux. On peut donc envisager un test sur les fichiers image qui rejetterait les incomplètes avant le traitement. On pourra utiliser les messages d'attention accessibles par le fonction `lastwarn` comme indicateur de l'état du fichier.

Le code sous Matlab

Il s'agit en un premier temps de créer une acquisition continue et régulière de l'image qui enregistrerait dans un unique fichier qui s'écrase et se régénere à chaque acquisition :

```
unix('dvgrab --every 10 --format jpeg --jpeg-overwrite image &');
```

On remarquera que le choix du nombre d'images par seconde considérées a son importance : un nombre trop faible donnera une indiquation sur l'image avec un temps de retard important, un nombre trop grand augmentera la

fréquence des cas d'images incomplètes, retardant ainsi le traitement. Paradoxalement, un nombre trop important risque donc d'engendrer aussi un retard.

Avant de faire un test sur les messages d'attention, il convient naturellement de prévenir les interruptions du calcul. On utilisera pour cela un `try` qui lance une action tant que cette dernière entraîne une interruption du calcul :

```
try
    image_anche=imread('image.jpg');
catch
end
```

On peut ensuite réaliser un test sur l'image dont le principe est de renouveller l'opération de lecture du fichier jusqu'à ce que l'opération n'entraîne aucun message d'attention :

```
while (length(lastwarn)>0)
  lastwarn('');
  try
    image_anche=imread('image.jpg');
  catch
  end
end
```

On notera qu'il nécessaire de réinitialiser systématiquement le message d'attention par la commande `lastwarn('')`. Evidemment, la lecture de l'image, le test et le traitement doivent s'effectuer au sein d'une boucle `while` qui va fonctionner en parallèle de la commande unix qui fait l'acquisition de l'image.

Un traitement en temps réel est donc tout à fait envisageable. On prendra soin cependant de bien définir les paramètres de traitement afin de réduire au maximum les temps de calcul, ainsi que de bien choisir le nombre d'images acquises par la caméra en une seconde à fin qu'il s'accorde bien avec le traitement.

C.9 Interface graphique

Il est possible d'associer le traitement d'image sous Matlab à une intrface graphique qui présenterait à l'utilisateur une palette des différentes

applications de traitement vues précédemment. Un retour direct à l'écran du résultat, pouvant associer plusieurs type de filtrage, permettrait de choisir au mieux les paramètres de calcul (seuil de binarisation, élément structurant pour les opérations morphologiques, etc...).

La figure C.34 présente l'interface réalisée. Survolons de manière succinte les possibilités offertes par l'interface.

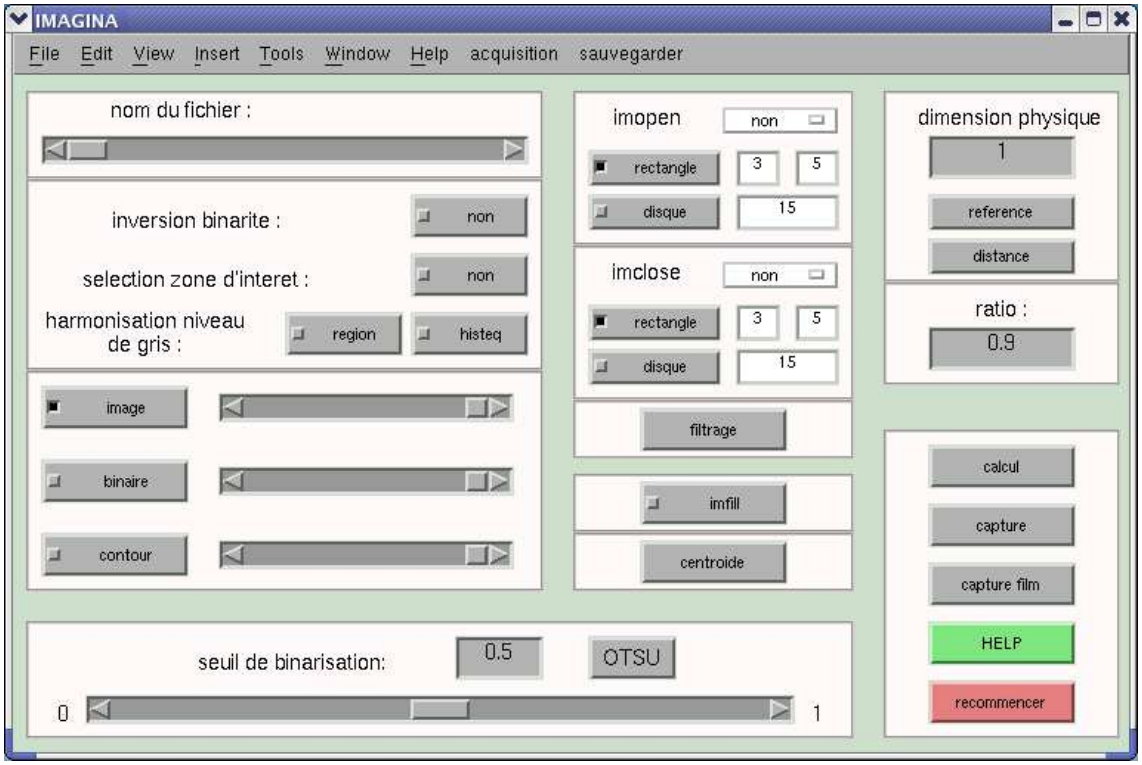

FIG. C.34 – Interface du programme de traitement d'image.

C.9.1 L'acquisition de données

L'interface peut faire l'acquisition :
- d'une image seule.
- d'une séquence d'images.
- de paramètres de traitement.

C.9.2 Visualisation

Une première barre de défilement permet de se déplacer le long d'une séquence d'images afin de pouvoir comparer le résultat d'un même traitement sur plusieures images distinctes de la séquence. Il est possible de jouer avec les transparences afin de pouvoir superposer une image originale et sa version traitée, ce qui peut aider au choix du traitement.

C.9.3 Traitement de l'image

L'ensemble des fonctions présentées dans le document ici présent sont accessibles depuis l'interface :

- Une barre de défilement contrôle le seuil de binarisation (une fenêtre éditable permet cependant de rentrer directement une valeur numérique).
- Des boutons à choix unique contrôle l'inversion de la binarisation, le zoom, l'harmonisation, les opérations morphologiques et la fonction imfill (le choix des points se fait directement sur l'image).
- Des fenêtre éditables permettent de choisir les dimensions de l'élément structurant.

C.9.4 Le calcul

L'interface doit permettre le calcul des dimensions de l'ouverture sur une image seule, comme sur une séquence. Pour cel, il est nécessaire de définir au préalable les paramètres du calcul :

- **Définition des valeurs tampons :** Un bouton **centroïde** permet de choisir sur n'importe quelle image la tache qui nous intéresse. Le programme en déduira le centroïde, le filtre et la distance de référence.
- **Définition d'une dimension de référence :** à partir de deux points de l'image et d'une valeur en dimension physique.
- **Définition du ratio :** c'est à dire du rapport minimum (souvent 0,8) utilisé dans le test sur la longueur de référence.

L'interface propose plusieurs type de calcul :

- **Le calcul d'une distance quelconque :** à partir de deux points quelconques de l'image, le code affiche dans la fenêtre Matlab la valeur de la distance.
- **Le calcul des dimensions de l'ouverture :** par l'option **calcul**, qui distinguera le cas d'une image isolée du cas d'une séquence.

C.9.5 Application en temps réel

A condition bien évidemment que la machine soit reliée à un apareil du type caméra que l'on puisse commander sous unix, l'interface permet le calcul de l'aire d'ouverture de l'anche en temps réel (on ne calcul que l'aire et non la totalité des dimensions afin de gagner en temps de calcul).

L'option **capture** lance l'acqisition d'une image qui s'affiche à l'écran. C'est à partir de cette dernière que seront réglés les paramètres du calcul en temps réel. Ensuite,l'option **capture film** lance une acquisition continue (par défaut de 2 images à la seconde) et affiche sous graphique l'évolution de la valeur calculée en temps réel.

C.9.6 Sauvegarde

L'interface offre la possibilité de sauvegarder :

- Un tableau 3 colonnes contenant les dimensions de l'anche sur une séquence (aire, longueur, largeur).
- Les paramètres du traitement ainsi que la localisation de la séquence sous la forme d'un fichier .mat qu'il est possible d'ouvrir à partir de l'interface.

Annexe D

Problème de visco-élasticité

On cherche à déterminer les constantes de temps permettant de caractériser le caractère visco-élastique d'une anche double.

D.1 Caractère visco-élastique de l'anche

L'anche possède une position de repos caractérisée par son aire H_0. Lorsque la surface des lames de l'anche double sont sollicitées en pression, l'anche se referme jusqu'à une aire d'ouverture H.

Si l'on libère l'anche de toute solicitation, cette dernière va s'ouvrir brusquement :

1. Pour osciller avec une faible amplitude autour d'une aire d'ouverture H_0' inférieure à H_0, et ce durant un temps τ_1.
2. Puis, tendre de manière continue vers l'aire d'ouverture H_0 durant un temps τ_2 bien supérieur à τ_1.

L'idée est de voir dans quelle mesure le traitement d'image sous Matlab et la méthode de calcul des dimensions de l'ouverture peut nous permettre d'estimer τ_1 et τ_2. Cependant, la très faible valeur de τ_1 (de l'ordre de quelques dizièmes de secondes alors que τ_2 est de l'ordre de plusieures dizaines de minutes) ainsi que la grande fréquence d'oscillation autour de H_0' impliquent de posséder une caméra capable de faire l'acquisition d'image à une fréquence très élevée... ce qui n'est pas le cas du laboratoire d'acoustique musicale de l'IRCAM. Nous verrons plus en détail au paragraphe D.4 pourquoi l'estimation de τ_1 est impossible avec une simple caméra DV.

D.2 Protocole expérimental

Nous avons à notre disposition une caméra DV, une arrivée d'air, une lentille grossissante et une bouche artificielle. Cette dernière consiste en une cavité fermée directement reliée à l'arrivée d'air ainsi qu'à un manomètre afin que l'on puisse contrôler la pression à l'intérieur.

L'anche double est à la fois à l'intérieur de la bouche (du coté des lames afin que l'on puisse exercer une pression à leur surface) et à l'exterieur (en sortie de l'anche afin que l'on puisse créer un écoulement dans le cas où ce serait nécéssaire). La transparence de la paroie de la bouche permet de pouvoir filmer l'ouverture de l'anche au court de la manipulation (voir le protocole experimental détaillé figure D.1).

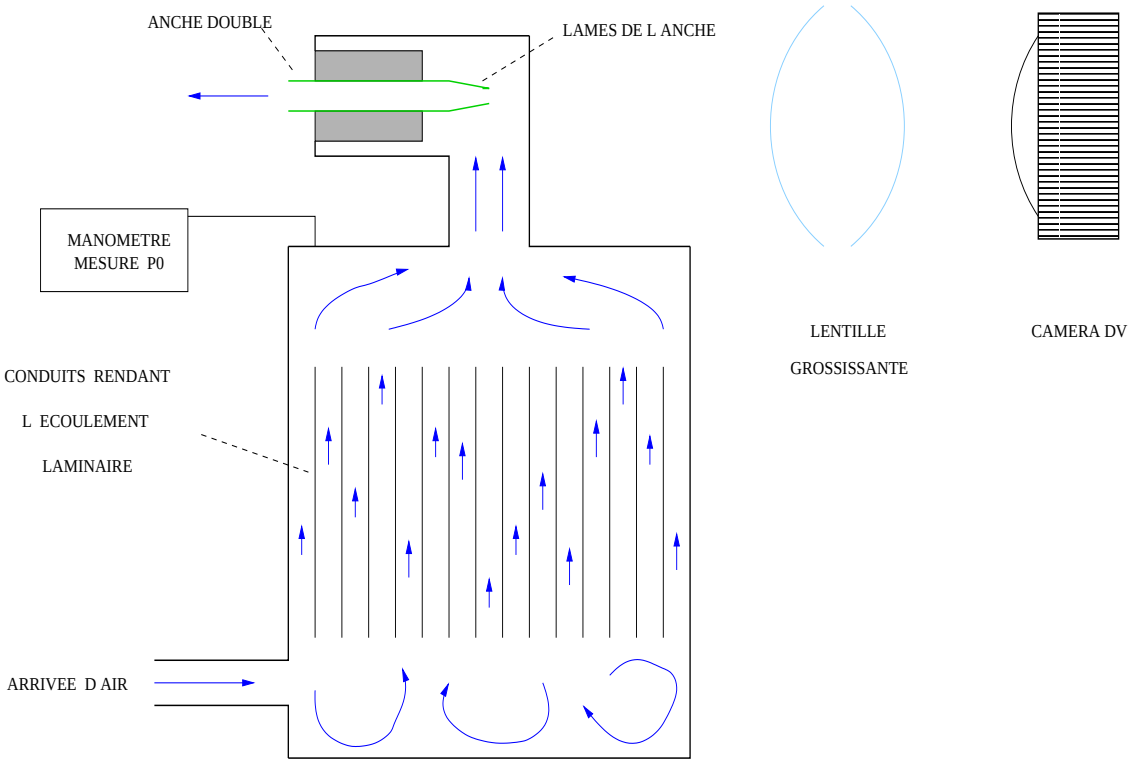

FIG. D.1 – Protocle expérimental (cas anche ouverte).

L'expérience se déroule selon les étapes suivantes :

1. On soumet les lames de l'anche à une pression P_0 de l'ordre de 40 kPa durant une quinzaine de minutes. Les caractéristiques mécanique de l'anche double font que les lames se ressèrent au maximum, maintenant

l'anche fermée. Le système {anche + bouche artificielle + arrivée d'air} est alors sans fuite. La pression P_0 y est stabilisée.

2. On commence à filmer l'anche totalement fermée.
3. On retire brusquement l'arrivée d'air, faisant décroître la pression dans la bouche jusqu'à ce qu'elle atteigne la valeur de la pression atmosphérique.
4. On continue de filmer l'anche ouverte durant environ 60 minutes.

D.3 Estimation de la constante τ_2

Comme nous l'avons précisé au paragraphe D.1, le protocole expérimental décrit précedemment nous permet uniquement l'estimation de τ_2. Ce qui suit décrit la démarche utilisée.

D.3.1 Choix de la séquence d'images

Nous avons vu précédemment qu'il était nécessaire de considérer l'évolution du système sur une échelle de temps de plusieurs dizaines de minutes pour pouvoir estimer τ_2. Nous considérerons donc naturellement une séquence s'étendant sur les 60 minutes de film. Cependant, la caméra enregistrant environ 25 images à la seconde, il serait maladroit de considérer la totalité des images enregistrées (et aussi inutile compte-tenu de la très faible vitesse d'évolution de l'aire d'ouverture sur le film).
Nous nous contenterons de retenir une image toutes les deux secondes, formant ainsi une séquence de plus de 1800 images pour un film de 60 minutes, ce qui est largement suffisant pour estimer τ_2.

Traitement des images

La figure D.2 présente une image extraite de la séquence, son image binarisée pour un seuil de 0,594, ainsi que le résultat après traitement de l'image. On propose comme traitement une inversion de la binarisation, puis une ouverture avec comme élément structurant un cercle de rayon 2 pixels, et enfin l'application de la fonction `imfill` en un seul point du rebord entourant l'anche.

Nous constatons que le résultat du traitement en figure D.2 est pleinement satisfaisant et laisse entrevoir un temps de calcul convenable.

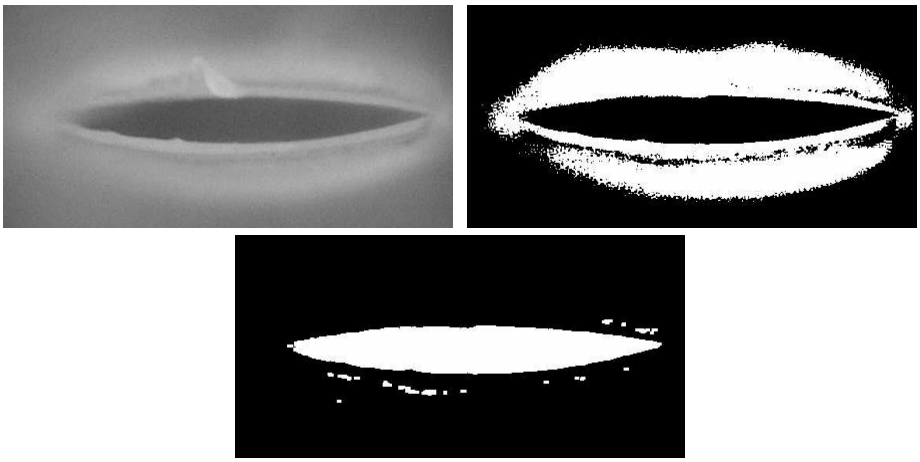

FIG. D.2 – Image de la séquence en nuance de gris (haut gauche) puis traitée (haut droit), puis traitée (bas centre).

Résultats

La figure D.3 présente l'évolution au court des 60 minutes de l'aire d'ouverture de l'anche double, à raison d'un calcul d'aire toutes les 2 secondes (c'est à dire toutes les 50 images acquises par la caméra).

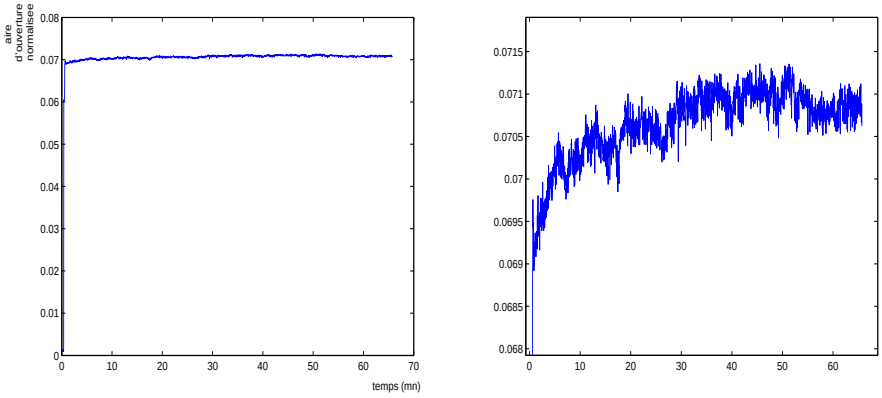

FIG. D.3 – Evolution de l'aire d'ouverture de l'anche au court du temps (gauche) et zoom sur la partie fluctuante (droite).

On remarquera que la transition entre la position fermée et ouverte est abrupte, la séquence ne contenant pas d'images présentant une ouverture intérmédiaire. Cela résulte du temps relativement long qui sépare deux images de la séquence (environ 2 secondes), largement supérieure au temps d'ouverture.

D.3.2 Interprétations

Il s'agit d'élaborer un modèle traduisant le comportement visco-élastique de l'anche, puis de le confronter aux valeurs calculées précédemment.

Pour cela, on va chercher sous Matlab les réels A, B et τ tels que la variation de l'aire d'ouverture durant l'expérience soit la plus proche possible de la forme :

$$aire = A + Be^{-\frac{t}{\tau}} \tag{D.1}$$

Cela revient donc à chercher sous Matlab la valeur de la variable X qui minimise la fonction :

$$f(X, data) = \sum_{i=1}^{n} \left(data(i) - X(1) - X(2)e^{-\frac{t_i}{X(3)}} \right)^2 \tag{D.2}$$

avec $data$, un vecteur de taille n contenant les variations de l'aire d'ouverture. On définit donc sous Matlab la fonction f de deux variables à l'aide de la commande `inline` :

```
f=inline('sum((data-X(1)-X(2)*exp(-[1:1:length(data)]/X(3)).^2)','X','data');
```

puis on détermine la variable X qui minimise f pour $data = aire$:

```
fminsearch(f,[0.06 0.01 100],[],aire)
```

où [0.06 0.01 100] est la valeur de X qui initialise la recherche. On obtient ces valeurs par lecture graphique ; elles correspondent aux ordres de grandeurs des valeurs que l'on recherche. On remarquera que cette méthode ne fonctionne qu'à condition de considerer les données d'aire tronquées de leurs premières valeurs de telle sorte qu'au temps zero, l'aire soit proche de 0.06.

La figure D.4 présente la solution proposée par Matlab, soit :

$$aire = 0.071 - 0.0016e^{-\frac{t}{659}} \tag{D.3}$$

On peut donc donner un ordre de grandeur du temps caractéristique τ_2 qui est d'environ 658 secondes, soit environ 11 minutes.

D.4 Estimation de la constante τ_1

L'estimation de la constante τ_1 se fait à partir de l'étude des toutes premières images de l'acquisition. Contrairement à l'estimation de la constante τ_2 (cf paragraphe D.3.1), nous ne pouvons nous contenter d'une image toutes les 2 secondes, tant le temps d'ouverture caractérisé par τ_1 est bref.

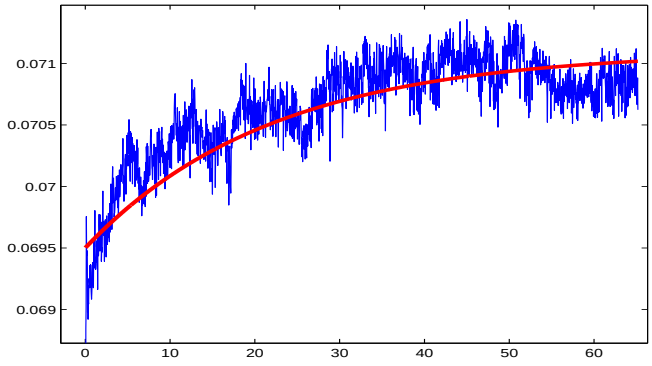

FIG. D.4 – Superposition des aires calculées par l'interface et de l'équation $0.071 - 0.0016e^{-\frac{t}{658,2956}}$

Nous considérerons donc une séquence contenant toutes les images acquises par la caméra durant le début de l'ouverture (environ 25 images par seconde), soit une séquence de plus de 570 images équivalent à environ 23 secondes.

Concernant le traitement des images, il ne diffère évidemment pas de ce qui a été proposé pour τ_2.

D.4.1 Résultats

La figure D.5 présente l'évolution de l'aire d'ouverture de l'anche au court des 20 secondes entourant le saut de pression, à raison de 25 calculs par seconde.

D.4.2 Exploitation des résultats

De toute évidence, ces résultats ne sont pas exploitables car :

1. les oscillations ne sont pas observables car elles se produisent à des fréquences de l'ordre du kHz.
2. certaines images de la partie transitoire qui suit juste le saut de pression semblent donner des valeurs d'aire incohérentes.

D.4.3 Explications

La remarque 1 explique pourquoi il est impossible de déterminer τ_1 avec une caméra DV classique, puisque la fréquence d'acquisition y est de 25Hz.

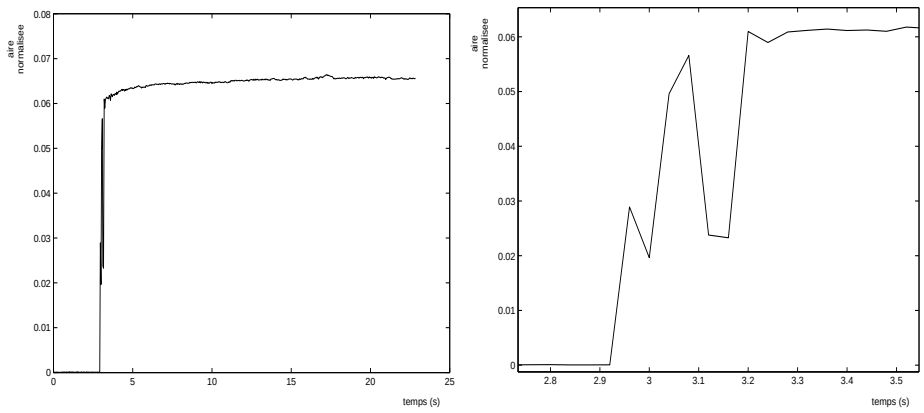

FIG. D.5 – Evolution de l'aire d'ouverture de l'anche sur les 20 premières secondes (gauche) et zoom sur la partie fluctuante (droite).

Concernant la remarque 2, un bref aperçu de la séquence permet d'extraire les images qui posent problème. On distinguera deux cas :

1. **Lorsque l'anche est dans la phase d'ouverture abrupte** qui suit directement le saut de pression. La figure D.6 présente l'image acquise par la caméra qui suit la dernière image d'anche fermée de la séquence (soit juste après le saut de pression).

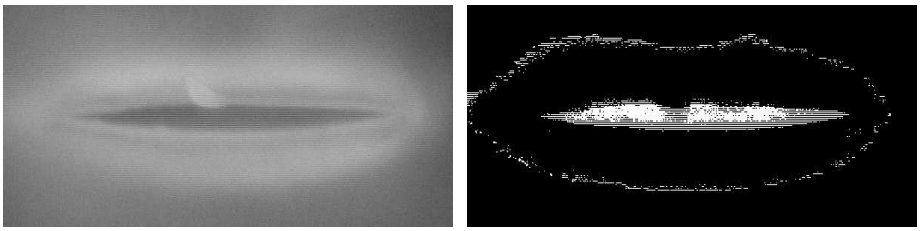

FIG. D.6 – Image de l'anche en phase d'ouverture abrupte, avant traitement (gauche) et après traitement (droite).

   On constate que la vitesse d'ouverture est si grande que la caméra superpose plusieurs ouvertures ce qui donne ces surimpressions des bords des lames sur l'ensemble de l'ouverture.

   La figure D.6 présente le résultat du traitement d'image et met ainsi en lumière l'erreur effectuée sur le calcul de l'aire d'ouverture.

2. **Lorsque l'anche est dans la phase d'oscillation** qui suit directement l'ouverture abrupte. La figure D.7 présente une image acquise par la caméra correspondant à cette phase d'oscillation.

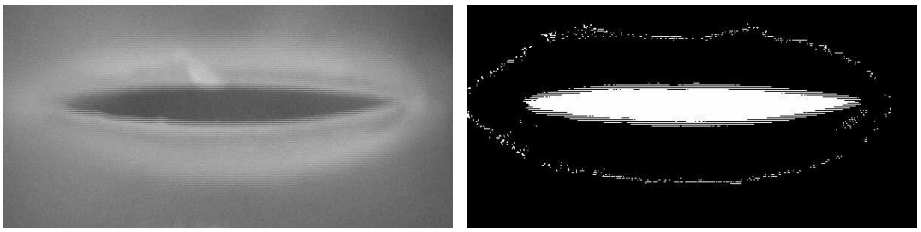

FIG. D.7 – Image de l'anche en phase d'oscillation, avant traitement (gauche) et après traitement (droite).

On constate que la vitesse d'oscillation est si grande que la caméra superpose plusieurs bords de lame.

La figure D.7 présente le résultat du traitement d'image et met ainsi en lumière l'erreur effectuée sur le calcul de l'aire d'ouverture.

Ces deux types de problème correspondent à des moments distincts de l'acquisition mais se traduisent par un même effet de surimpression qui modifie l'image de la tache et rend impossible le calcul de ses dimensions.

Un moyen d'éviter une telle surimpression est de pouvoir diminuer le temps d'exposition de la caméra, ceci n'étant pas possible avec une caméra DV classique.

Résumé

Le stage effectué à l'Institut de Recherche et Coordination Acoustique/Musique sous la tutelle de M. Christophe Vergez, se superpose au travail de thèse de M. André Almeida sur le fonctionnement physique du hautbois. Il traite d'un point particulier de ce travail : la détermination expérimentale des paramètres physiques de l'anche.

La durée de stage se décompose en trois parties distinctes. En un premier temps, une approche générale de l'étude expérimentale et du traitement de données par la réalisation sous Matlab d'un outil de calibration de capteur de pression. Puis, une recherche plus approfondie sur le traitement d'images sous Matlab et les méthodes de calcul des dimensions de l'ouverture de l'anche double. Et ce, dans le but de réaliser une interface de traitement d'images utilisable en post-traitement ou en temps réel avec une caméra DV. Enfin, l'utilisation des outils précédemment élaborés afin de déterminer les paramètres physiques de l'anche double du hautbois tel que son caractère visco-élastique.

Abstract

The practicum was realized at IRCAM under the supervision of Mr. Christophe Vergez. It was a complementary work to the thesis of Mr. André Almeida about the physical behavior of the oboe. The subject of the practicum points on a particular aspect of the thesis: the mecanical coupling between the double reed and the resonator.

We can decompose this practicum in three parts. At first, a general approach of the experimental work and of data post-processing by the realisation with Matlab of a calibration tool of pressure sensors. Next, an research about image processing with Matlab and methods to estimate the reed dimensions and position during the experiment. The aim of this part of the practicum is to write a program with a graphical interface allowing the user to perform image post-processing or real-time tasks on a DV camera output. Finally, practical applications have been investigated using the tools developped during the practicum. For example, the visco-elastic behavior of the double reed of an oboe has been studied experimentally.