Analyse de la variabilité et prédiction des mouvements d'un chef d'orchestre

par
Victoire Louis

Sciences de la gestion
Intelligences d'Affaires

Mémoire présenté en vue de l'obtention
du grade de maîtrise ès sciences
(M.Sc.)

Juillet 2018

Remerciements

Je voulais dire un grand merci à toutes les personnes qui ont contribué de près ou de loin à l'élaboration de ce mémoire. Il représente l'aboutissement d'un travail mais aussi de mes années passées à HEC Montréal.

Je voudrais donc tout naturellement remercier mon école et ses professeurs qui m'ont permis de me développer intellectuellement, professionnellement et personnellement au cours des six dernières années.

Plus personnellement, je souhaite remercier mes deux co-directeurs Aurélie Labbe et Benoît Ozell qui m'ont suivie et encouragée dans ce projet. Aurélie m'a donné un cadre pour développer mes capacités mathématiques et un espace pour laisser libre cours à ma créativité. Merci de m'avoir écoutée et conseillée dans des moments difficiles. Benoit m'a accueillie à bras ouverts à Polytechnique. Je lui suis reconnaissante pour cette riche collaboration.

Merci à Paolo Bellomia, chef d'orchestre, professeur et co-directeur du programme de direction d'orchestre de la faculté de musique de l'université de Montréal, d'avoir pris le temps de me faire découvrir l'univers de la direction d'orchestre, me permettre d'assister à ses cours de direction et ses répétitions comme une vraie privilégiée.

Merci à Jean-Francois Rivest, chef d'orchestre, professeur et co-directeur du programme de direction d'orchestre de la faculté de musique de l'université de Montréal, pour sa collaboration et sa disponibilité.

Merci à Tram Dang d'avoir pris le temps de répondre à mes questions de novice en matière de direction d'orchestre.

Merci à Edouard Labbe Guerard de m'avoir écoutée et encouragée à aller frapper à la porte du laboratoire de Benoit à Polytechnique.

Merci à Laurent Charlin d'avoir pris le temps de me recevoir et de m'écouter.

Merci à Danilo Dantras pour son intérêt dans mon projet.

Merci à Marc Fredette d'avoir cru en moi et de m'avoir offert de nombreuses opportunités au cours des deux dernières années. Je n'aurais peut-être jamais fait de statistique si je n'avais pas croisé sa route.

Merci à Julie Lemonde pour avoir alimenté mon intérêt et mon goût pour les nouvelles technologies.

Merci à tous les professeurs du département qui ont rendu l'expérience de cette maîtrise aussi agréable. Je souhaite à tous d'avoir une expérience aussi enrichissante que celle dont j'ai bénéficié.

Merci à tous mes proches et tous mes amis sans qui ce mémoire n'aurait sûrement pas pu voir le jour.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

1.1 Contexte

1.1.1 L'industrie du divertissement : toujours plus technologique

L'industrie du divertissement a subi de grandes transformations depuis les dix dernières années. Depuis l'avènement des nouvelles technologies, comme les téléphones mobiles ou les ordinateurs portables, les habitudes de consommations du public ont complétement changé. Deloitte, dans son rapport sur l'industrie des média et du divertissement de 2018 (18), constate une augmentation de la personnalisation des expériences à travers les différentes sources de contenu, publicité ou marques. Le défi des entreprises sera donc encore pour les années à venir de créer des expériences qui donnent le sentiment au consommateur d'être adaptées à ses besoins et ses envies. Deloitte note également que la réalité virtuelle (VR) et la réalité augmentée (AR) continuent d'attirer une attention grandissante des acteurs du divertissement qui sont toujours à la recherche de nouveaux moyens de se différencier. En 2016, les investissements VR/AR réalisés par les entreprises de médias ont augmenté de 137% par rapport à 2015 aux États-Unis. De plus, les applications majeures de la VR et AR seront probablement celles qui utilisent ces technologies pour améliorer l'expérience des consommateurs et magnifier la narration (*storytelling*), plutôt que de fournir la totalité de l'expérience sur ces supports.

Dans les grandes tendances de l'industrie des médias et du divertissement, nous pouvons aussi remarquer l'importance des personnes qui montrent leur appréciation envers les marques pour les différents acteurs de l'industrie (39), (27). Nous appelons ces personnes des *followers*. Il devient crucial d'engager ces *followers*, de les connaître pour mieux les cibler. Mais ils deviennent aussi les premiers promoteurs des artistes et des contenus de divertissement. Les acteurs de l'industrie sont donc en mesure de promouvoir leur contenu à travers un nombre croissant de personnes qui ont moins de *followers* que les grandes célébrités des réseaux sociaux, mais qui sont beaucoup plus engagés. Ces admirateurs sont récompensés pour leur comportement, renforçant l'amour de leur marque, de leurs artistes et suscitant des réponses vraiment authentiques.

Cette explication de l'industrie illustre bien les difficultés que peut rencontrer la musique symphonique. En effet, face à la multitude d'offres récréatives, la musique symphonique ne trouve que peu de fervents supporters. Insérer des nouvelles technologies à des concerts de musique symphonique permet donc d'attirer un nouveau public et d'éclairer ce genre sous un nouveau jour.

1.1.2 Les technologies et la musique classique à travers le monde

Le désintérêt pour la musique classique et symphonique est mondial. Des initiatives technologiques sont donc prises à travers le globe pour attirer un public plus jeune. Depuis 2016, l'orchestre symphonique

de Boston (BSO) prête des Ipads avec du contenu multimédia adapté au thème de la soirée aux spectateurs qui font partie d'une section particulière de la salle. De plus, ils sont placés derrière des écrans plats qui leur permettent de voir le chef d'orchestre comme les musiciens le voient (7), (49). Dans un autre style, en 2008, Asimo, un robot humanoïde créé par Honda a dirigé l'orchestre symphonique de Détroit (48). En 2017, un autre robot, Yumi, a été entraîné en Suisse pour diriger le ténor Andrea Bocelli et l'orchestre philharmonique de Lucques à Pise (34). Même si la performance de Yumi a été meilleure que celle d'Asimo, Yumi est loin de remplacer un chef d'orchestre. Il n'a pas d'émotion et ne peut pas improviser. Ce sont les musiciens qui doivent s'adapter au robot. Or, la logique des choses veut qu'il y ait une interaction entre le chef et les musiciens.

A montréal, la tendance est la même. Dans ce cadre, la faculté de musique de l'Université de Montréal organise des partenariats avec l'école Polytechnique pour redynamiser ses concerts de musique symphonique. Les deux universités collaborent pour créer des concerts plus intéractifs mais également faire évoluer l'apprentissage de la conduite d'orchestre grâce aux nouvelles technologies. Ce mémoire s'inscrit dans le contexte de ces collaborations.

1.2 Objectifs du mémoire

Les deux objectifs de ce travail sont d'analyser la variabilité des mouvements d'un chef d'orchestre et de parvenir à prédire le type de mouvements que le chef réalise à partir de la position de son corps. Pour cela, nous avons utilisé des données collectées lors d'une expérience réalisée à l'Université de Montréal avec un chef d'orchestre. Des capteurs ont été posés sur ses bras lorsqu'il dirigeait des musiciens. Ces capteurs enregistraient différents types de données que nous avons ensuite traitées. Nous nous sommes également appuyés sur les informations extraites de la partition de musique pour certaines de nos analyses.

1.3 Organisation du mémoire

Dans le chapitre 2, nous expliquerons plus en détails le projet et la nature des données à notre disposition. Dans le chapitre 3, nous traiterons le premier objectif, c'est-à-dire, l'analyse de la variabilité des mouvements. Dans le chapitre 4, nous développerons un modèle qui prédit le type de mouvements que le chef réalise basé sur les positions de ses mouvements. Le chapitre 5 sera une discussion sur les résultats de ce mémoire. Le chapitre 6 comprendra les annexes. La bibliographie clôturera ce travail.

Chapitre 2

Les données de capteurs de mouvements et le chef d'orchestre

2.1 Le projet

La musique symphonique à Montréal est particulièrement en perte de vitesse car il existe un vaste choix d'offres culturelles. La musique symphonique n'est pas nécessairement le premier choix du public s'il n'a pas été initié auparavant. Le but du projet initial a donc été d'allier les nouvelles technologies à la musique symphonique pour rajeunir l'image des spectacles et attirer un nouveau genre de public. Ce projet, qui a débuté en 2010, est réalisé conjointement entre l'Ecole Polytechnique de Montréal, plus particulièrement le laboratoire de réalité virtuelle et infographie du département de génie informatique, à travers Benoit Ozell et Francois-Raymond Boyer, et le département de musique de l'Université de Montréal (UdeM) avec Paolo Bellomia, professeur de direction d'orchestre. Depuis 2010, ils ont réalisé différents spectacles dans lesquels ils ont utilisé la réalité augmentée pour ajouter un intérêt au concert et venir susciter davantage la curiosité du spectateur sans prendre le dessus sur l'essence du spectacle (4).

Dans le cadre de ce projet, dès 2010, les chercheurs ont tenté de reproduire l'hologramme du chef d'orchestre. Ainsi, des capteurs de mouvements ont été posés sur les bras de Paolo Bellomia lorsqu'il dirigeait la pièce "Arcana" composée par Edgar Varèse entre 1925 et 1927 comme nous pouvons le voir sur la figure 2.1. Grâce aux données récoltées, ils ont pu reproduire l'hologramme de Paolo Bellomia en 3 dimensions. L'hologramme a été utilisé lors de différents spectacles. Les chercheurs espèrent que l'hologramme pourra être utilisé dans le cadre éducatif pour permettre aux apprentis chef d'orchestre d'observer plus facilement les mouvements d'un chef dans de multiples directions et de pouvoir répéter cette visualisation à volonté (36). Ces données, obtenues lors de cette expérience, ont été analysées dans le cadre de ce mémoire.

FIGURE 2.1 – Le chef d'orchestre lors d'un concert muni des capteurs de données

2.2 L'art de la conduite d'orchestre ou l'art de se rendre inutile

Avant de passer à la partie technique de mon sujet, voici quelques informations sur la fonction du chef d'orchestre et son importance pour l'orchestre.

2.2.1 Concrètement, que fait un chef d'orchestre ?

Nous pouvons nous demander ce que fait un chef d'orchestre, à quoi servent ses mouvements et toutes ses gesticulations. Pour les novices, tout cela parait bien étrange et très aléatoire. Mais sans surprise, aucun des mouvements n'est dû au hasard.

Dans le processus de direction, le chef d'orchestre commence par étudier la partition avec minutie. Il apprend tous les détails de cette dernière. Il pourrait être qualifié d'expert de la partition comme le dit David Robertson dans (41). Dans un deuxième temps, il choisit l'interprétation qu'il veut faire de la partition. Il fait les choix musicaux et artistiques qui rendent son interprétation unique. Comme le metteur en scène, au théâtre, qui décide de mettre en valeur certaines répliques plus que d'autres, du ton des acteurs... Le chef d'orchestre met en scène la partition avec les instruments de l'orchestre : lors des répétitions, il fait jouer certains instruments parfois plus forts, parfois moins forts, accélère ou ralentit le rythme pour parvenir au plus près de l'idée qu'il se fait de la partition.

Les choix musicaux se font mesure par mesure. D'après Paolo Bellomia, ce sont les notes et les commentaires du compositeur écrits sur la partition qui guide le chef à choisir une interprétation musicale d'une mesure plutôt qu'une autre. Ce sont la qualité de ces choix et la capacité de les mettre en pratique qui permettent notamment de distinguer les chefs d'orchestre.

2.2.2 Le chef d'orchestre comme allégorie du chef d'entreprise

Le chef d'orchestre est une belle allégorie du chef d'entreprise (ou le chef d'entreprise est une belle allégorie du chef d'orchestre). Un orchestre symphonique peut dépasser la centaine de musiciens. Le chef d'orchestre est donc un dirigeant comme l'est un dirigeant d'entreprise. Un bon chef d'orchestre doit être capable de connaître le potentiel de chaque instrument et idéalement de chaque musicien. A ce poste, il doit connaître les capacités, les forces et faiblesses des personnes avec qui il travaille pour anticiper les sonorités et les harmonies entre les instruments mais aussi les capacités des humains qui jouent derrière les instruments. Cette anticipation garantie au chef une meilleure cohésion entre les parties, ce qui donnera un résultat plus proche du but recherché.

Le chef, aussi appelé maestro, connaît parfaitement l'oeuvre dans les moindres détails. Sa partition rassemble toutes les partitions de tous les instruments. Le chef maîtrise toutes les notes qui sont jouées

par l'orchestre. Il a face à lui des musiciens qui sont des spécialistes de leur instrument. Ils excellent tous dans leur domaine respectif. Un des objectifs du chef d'orchestre est de mettre en valeur le talent, l'expérience et l'expertise des musiciens sans porter atteinte à l'unité de l'orchestre. Un bon chef sera capable de permettre à chacun de ses musiciens individuellement de se dépasser musicalement et de jouer du mieux qu'il peut. Il sera capable aussi d'unifier l'ensemble des parties de l'orchestre pour donner le sentiment d'un tout harmonieux et faire oublier à l'auditoire que des dizaines de musiciens, avec des vécus différents, qui jouent des instruments hétéroclites avec des façons d'être jouées complétement différentes, sont sur scène. Souvent, les chefs d'orchestre connaissent plusieurs instruments. C'est une des facultés qui leur permet de mieux comprendre les expériences des musiciens qu'ils dirigent. Cela leur donne aussi une meilleure connaissance de l'agencement des sons entre eux.

De plus, le maestro doit donner les clés nécessaires à l'orchestre pour obtenir cette unité. Il est donc logique que "le but du chef d'orchestre devrait être de se rendre inutile" comme le précise Darko Butorac d'après les mots de Herbert Von Karajan dans (11). Le chef doit communiquer avec les musiciens le plus possible sur ce qu'il veut, mais aussi sur ce que les musiciens comprennent. Ainsi, savoir jouer de plusieurs instruments permet de mieux communiquer avec chacun, mais donne également une plus grande légitimité au discours du maestro. En effet, lors des entrevues que j'ai pues avoir avec les différents chefs d'orchestre que j'ai rencontrés, le charisme et la légitimité sont des points qui reviennent très fréquemment. Tram Dang, qui a étudié à l'université de McGill en musique et a dirigé l'orchestre étudiant de McGill, résume très bien la situation : 'Tu ne peux pas arriver devant 80 musiciens qui sont excellents dans ce qu'ils font et douter de ce que tu veux. Tu n'as pas le temps et puis si tu leur dis que finalement, tu aimerais bien essayer une autre interprétation car tu as changé d'avis, ils ne te font plus confiance.' Les bons chefs d'orchestre sont rarement très jeunes, tout comme les bons dirigeants d'entreprise. Ils ont d'abord été musiciens, compositeurs puis ils se sont finalement tournés vers la direction d'orchestre. C'est avec le temps qu'ils acquièrent l'expérience nécessaire et donc la légitimité de diriger un orchestre. Il en va de même pour les dirigeants d'entreprise ou les managers. Ce sont souvent les plus expérimentés, ceux qui ont vu plusieurs côtés techniques qui arrivent à comprendre les besoins et les objectifs de chacun. Ils ont la capacité d'unifier et de faire avancer l'entreprise.

2.3 Présentation des données

Nous détaillons maintenant l'origine des données avec lesquelles nous avons travaillé pour ce mémoire. Nous avons deux sources principales de données : les capteurs qui ont été posés sur le chef d'orchestre et des données récoltées à partir de la partition de musique. Nous avons obtenu 90 variables à partir des capteurs et 16 variables à partir de la partition du morceau interprété.

2.3.1 Les données extraites des capteurs

Les capteurs ont été posés sur le chef d'orchestre pendant une durée d'environ 40 minutes lors d'une répétition. Le jeu de données brut avec lequel j'ai travaillé contenait 151 737 observations pour chaque variable relevée. Cependant, seule une partie de ces 40 minutes nous intéresse pour ce projet. En effet, Paolo Bellomia n'effectue pas pendant 40 minutes des gestes de direction d'orchestre. Il dirige des parties de la pièce puis arrête l'orchestre pour donner ses commentaires. Or, nous cherchons à analyser des mouvements précis du chef d'orchestre. Nous avons donc besoin d'une séquence de mouvements de direction d'orchestre conséquentes. De plus, pour être capable de prédire les mouvements précis du chef, nous devons être capable d'associer les données des capteurs avec le type de mouvements réalisé et donc les données de la partition. La partie de l'enregistrement qui nous intéresse ici dure 11 minutes et 45 secondes. Nous avons une vidéo de cette partie. De plus, je me suis appuyée sur le travail de Francois-Raymond Boyer, professeur

à l'Ecole Polytechnique de Montréal, qui a créé une base de données permettant de suivre au cours des 11 minutes 45 secondes l'avancée dans la partition. J'ai assemblé cette petite base de données avec celle sur les capteurs. C'est grâce à la colonne du temps que j'ai pu retrouver les 11 minutes 45 secondes pertinentes à mon projet à l'intérieur des 40 minutes. J'ai ensuite réduit la base de données des capteurs pour n'obtenir que 44 467 relevés pour chaque variable.

Plus techniquement, le taux d'échantillonnage nominal des capteurs est de 64 relevés par seconde, ce qui correspond à un relevé chaque 15,6 millisecondes environ. Chaque relevé contient les informations de 6 capteurs qui sont sur les bras : 1, 2, 3 sont pour la main, l'avant-bras et le bras droit, respectivement, et 4, 5, 6 la même chose pour le bras gauche. Chaque capteur donne, selon 3 axes X, Y et Z, les accélérations en mètres par seconde carrée, les vitesses de rotation en radians par seconde, et le nord magnétique (sans unités). L'axe X varie d'avant en arrière : l'avant est vers le positif et l'arrière vers le négatif car le système d'axe tracé est dit "main droite". L'axe Y varie de gauche à droite : le bras droit est du côté négatif et le bras gauche du côté positif. L'axe Z varie de haut en bas : la tête correspond au côté positif et les pieds au côté négatif. A partir de ces valeurs, les valeurs de la série d'angles d'Euler yaw, pitch et roll ont été calculées. Les données de positions des capteurs selon les 3 axes (X, Y, Z) ont aussi été enregistrées. Ce sont principalement ces données que nous utilisons. Nous avons un total de 90 variables provenant des capteurs. Le tableau 2.1 donne un récapitulatif de ces variables. Il n'y a aucune valeur manquante dans le jeu de données. Cela ne sera donc pas un problème tout au long de ce travail.

2.3.2 Tableau récapitulatif des variables extraites des capteurs

TABLE 2.1 – Les 5 grandes catégories de variables extraites des capteurs

Type de variables	Capteur	Nombre de variables
Position (sans unité)	les 6 capteurs dans les 3 directions X, Y et Z	18
Accélération en m/s^2	les 6 capteurs dans les 3 directions X, Y et Z	18
Vitesse de rotation en $radian/s$	les 6 capteurs dans les 3 directions X, Y et Z	18
Nord Magnétique (sans unité)	les 6 capteurs dans les 3 directions X, Y et Z	18
Roll (sans unité)	les 6 capteurs	6
Pitch (sans unité)	les 6 capteurs	6
Yaw (sans unité)	les 6 capteurs	6

2.3.3 Les données extraites de la partition

Comme explicité précédemment, les chefs ne font pas des gestes au hasard. Bien au contraire. Les gestes sont très précis. Chaque geste existe pour une raison définie. Chaque geste est là pour transformer certaines idées du compositeur en musique comme l'explique Darko Butorac dans (11). Grâce à Paolo Bellomia, j'ai eu la chance d'assister à des cours de conduite d'orchestre pour chefs débutants et avancés et donc à l'apprentissage de ces différents mouvements à différents niveaux. La première grande distinction à faire dans les gestes d'un chef d'orchestre est entre le bras droit et le bras gauche. Le bras droit bat la mesure, il est donc constamment en mouvement. Le bras gauche, quant à lui, exprime les crescendos, decrescendos, les entrées de certains musiciens dans la musique, et donne aux musiciens des indications pour s'ajuster à l'interprétation du maestro. Les mouvements du bras droit sont donc très réguliers et prévisibles. Les mouvements du bras gauche le sont beaucoup moins, ils dépendent de la volonté du chef, de ses émotions et de l'interprétation des musiciens le jour de la répétition ou du concert. Dans ce mémoire, je me concentre donc principalement sur les mouvements du bras droit où des répétitions de mouvements sont visibles. Les gestes effectués avec le bras gauche sont plus anecdotiques et peu répétés sur une pièce d'une durée de 11 minutes 45 secondes.

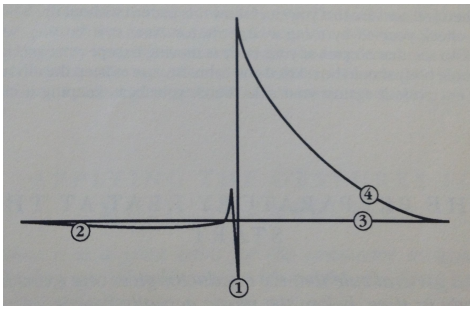
Dans (42), Max Rudolf explique les principaux mouvements de base et donne les graphiques des mouvements correspondants. Les mouvements du bras droit sont basés sur les temps indiqués sur la partition pour chaque mesure. De plus, un mouvement correspond à une mesure. Il y a 267 mesures dans le morceau, Paolo Bellomia a donc réalisé 267 mouvements avec la main droite. Avec Paolo Bellomia,j'ai relevé chaque mouvement qu'il effectue lors de chaque mesure. Dans le tableau 2.2 nous pourrons voir ces différents mouvements ainsi que le nombre d'occurrences de chacun.

TABLE 2.2 – Nombre d'occurrences des différents mouvements au cours de la pièce.

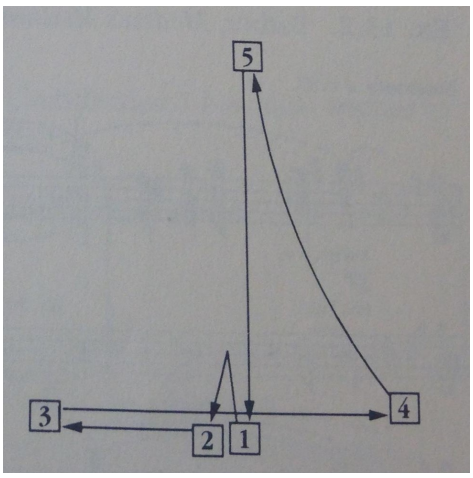
Nom du mouvement	**Nombre d'occurrences du mouvement dans le morceau**
4 temps neutre	138
3 temps neutre	42
5 temps alterné 3+2	40
2 temps neutre	25
7 temps 3+4	3
6 temps italien	3
7 temps 4+3	2
7 temps 2+2+1+2	2
12 autres types de mouvements	1 fois chacun

Au vu du grand nombre de répétitions (137/267) du mouvement 4 temps neutre, ce qui correspond à plus de 50% des mouvements, dans la dernière partie du mémoire, nous avons choisi de traiter les mouvements de façon binaire : 4 temps neutre versus le reste des mouvements.

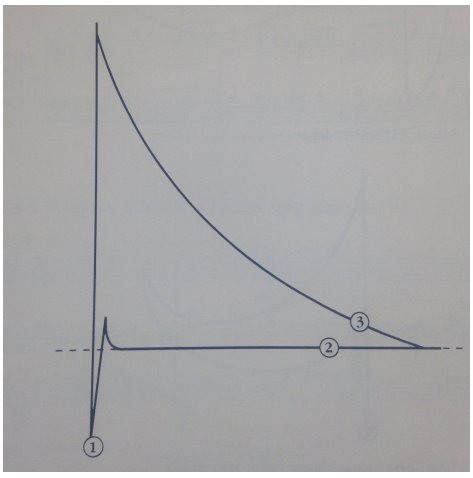
Dans (42), Max Rudolf nous explique la théorie de la conduite d'orchestre, les différents mouvements à effectuer en fonction de la musique et nous donne un exemple de squelette des différents mouvements existants.Nous présentons ici les squelettes des 4 mouvements les plus répandus dans le morceau. Les numéros que nous pouvons apercevoir sur les différentes figures sont les temps où la main doit se trouver.



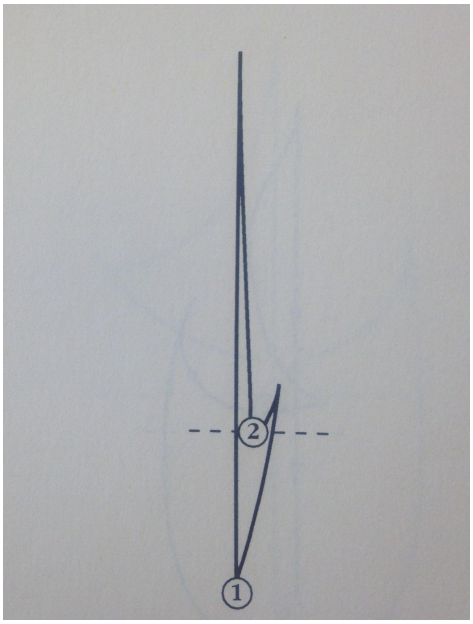
(a) 4 temps neutre



(b) 5 temps alterné 3+2



(c) 3 temps neutre legato



(d) 2 temps neutre legato

FIGURE 2.2 – Structure de certains mouvements du chef d'orchestre, respectivement figure 1.3, 13.1d, 5.1, et 2.2d de (42)

En plus de connaître les différents mouvements, nous avons relevé différentes variables qui ont trait à la musique et au temps grâce à la partition. Ces variables figurent dans le tableau 2.3

TABLE 2.3 – Tableau récapitulatif des variables extraites de la partition

Nom	Description
temps_mesure_ms	Temps dans l'enregistrement du début de la mesure en millisecondes
temps_min	Temps dans l'enregistrement du début de la mesure en minutes
temps_mesure_s	Temps dans l'enregistrement du début de la mesure en secondes
duree_mesure_ms	Durée totale de la mesure en millisecondes
Page	Numéro de la page de la partition
id_mesure	Numéro de la mesure

TABLE 2.3 – Tableau récapitulatif des variables extraites de la partition

nb_temps	Nombre de temps dans la mesure (notation numérique au début de chaque mesure, lorsqu'il n'y a qu'un chiffre correspond à ce dernier, lorsqu'il y en a 2, correspond au chiffre du haut)
temps_dénom	Temps de référence, plus le chiffre est petit, plus la musique est lente. 2= blanche ; 4=noire ; 8=croche
Move_name	Nom du mouvement effectué par le chef d'orchestre durant la mesure
move_id	Identifiant donné à chaque type mouvement ; 1=5 temps alterné 3+2 ; 2=2 temps neutre ; 3=3 temps neutre ; 4=4 temps neutre ; 5=autre
move_id_bi	Identifiant binaire donné à chaque mouvement du chef d'orchestre 1=4 temps neutre ; 0=autre
tempo	Valeur du tempo
time	Temps dans l'enregistrement de chaque relevé
temps_ref_st	Temps écoulé dans le morceau avec comme point de départ 0ms
temps_mesure_ms_st	Temps dans le morceau de début de la mesure en millisecondes avec comme point de départ 0ms
duree_mesure	Durée écoulée de la mesure en millisecondes.

2.4 Visualisation des données

2.4.1 Graphiques de comparaison

Nous présentons ici la position en X de la main droite et de la main gauche en fonction de la position en Z pour différents types de mouvements. Nous avons choisi de représenter les graphiques avec les directions X en fonction de Z car ils sont plus intuitifs et permettent de mieux faire la distinction entre les mouvements. Les figures ont été obtenues avec la bibliothèque ggplot2 du logiciel R.

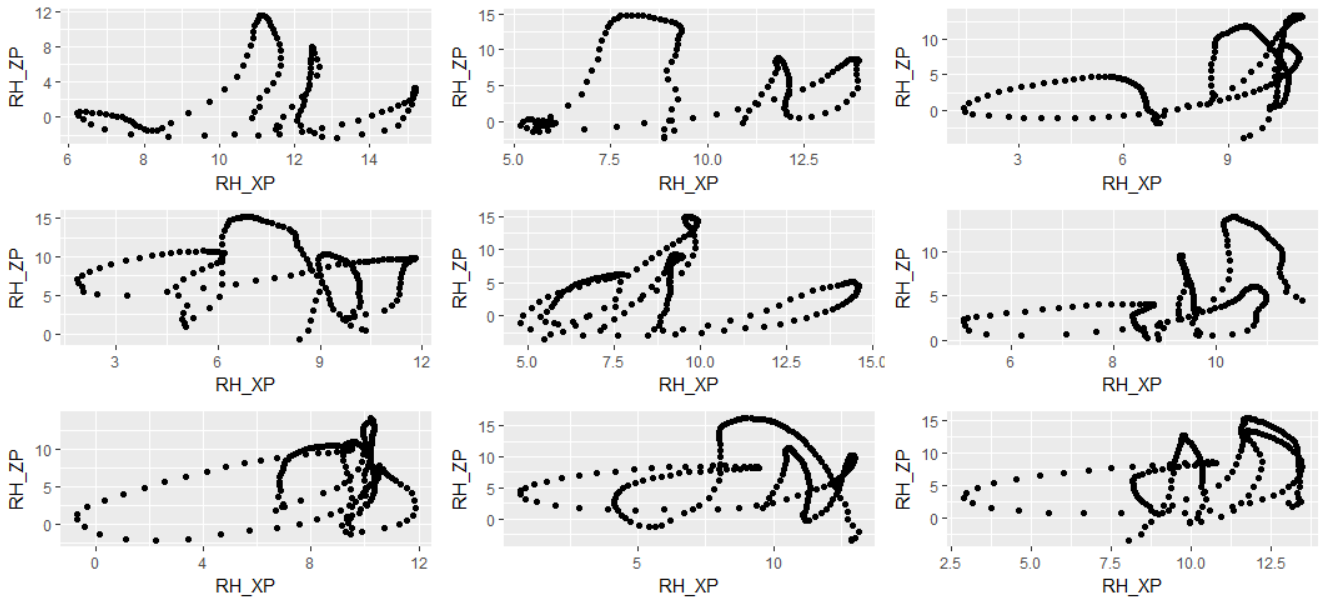


FIGURE 2.3 – Position de la main droite en X en fonction de Z pour 9 mouvements de 4 temps neutre

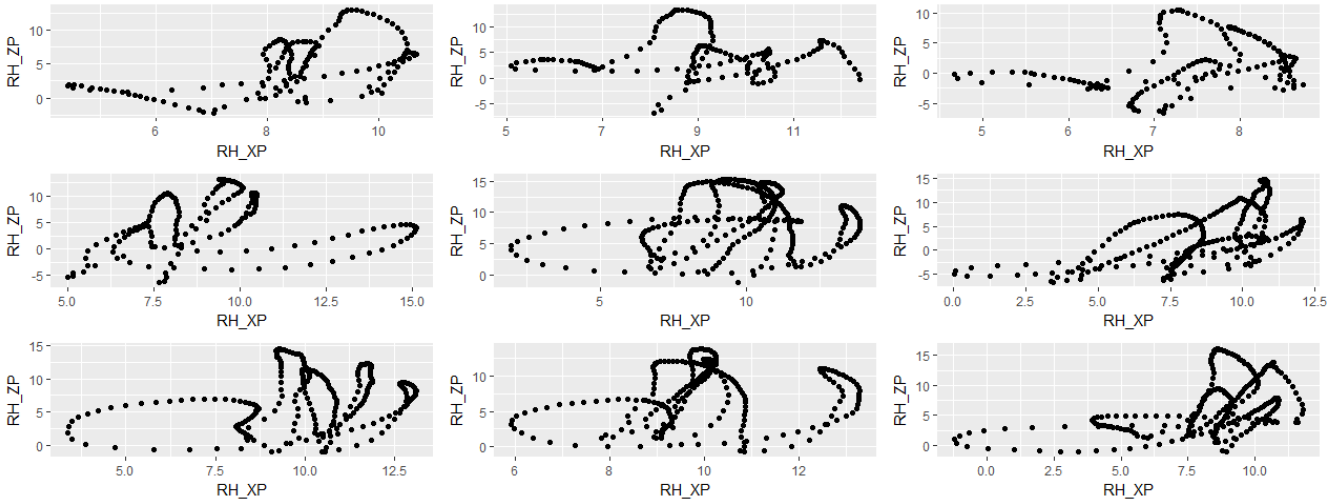


FIGURE 2.4 – Position de la main droite en X en fonction de Z pour 9 mouvements de 5 temps alterné 3+2

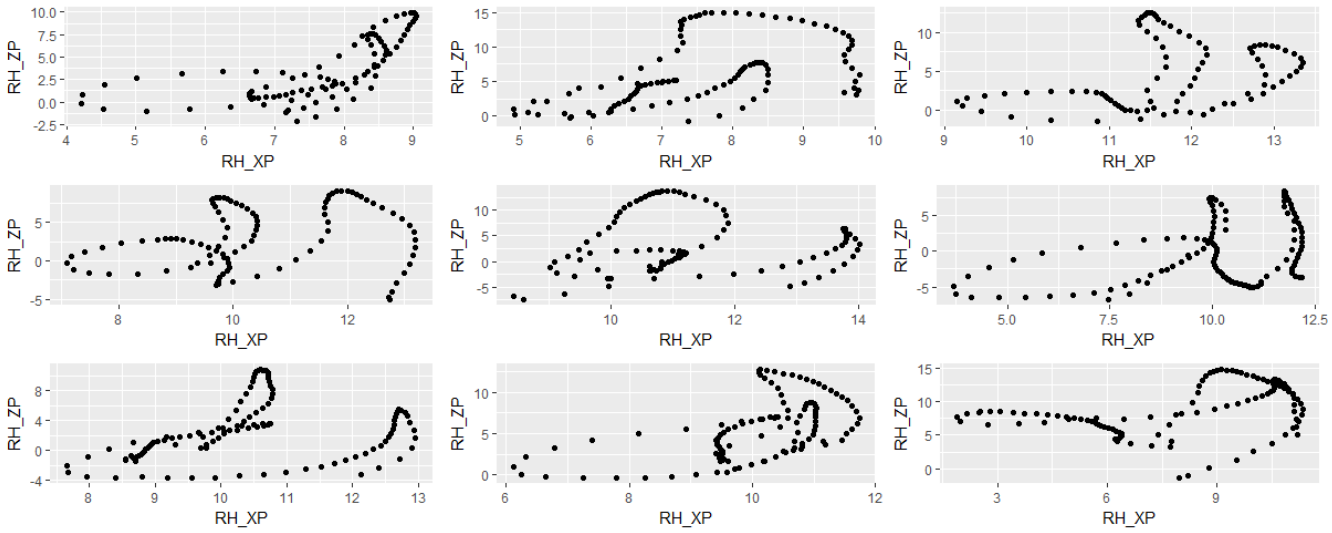


FIGURE 2.5 – Position de la main droite en X en fonction de Z pour 9 mouvements de 3 temps neutre

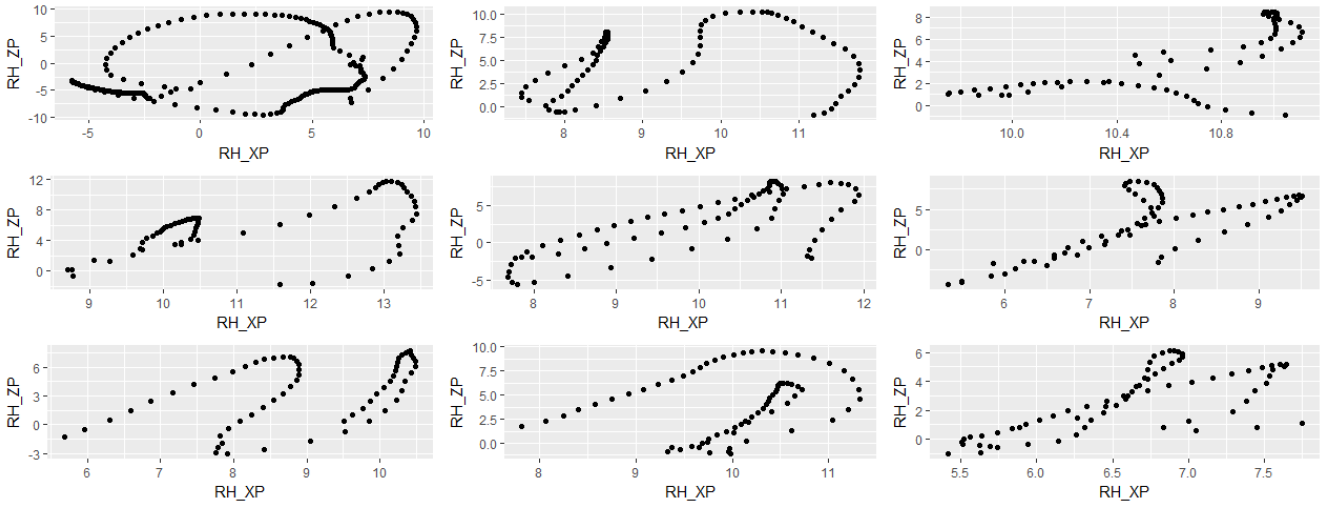


FIGURE 2.6 – Position de la main droite en X en fonction de Z pour 9 mouvements de 2 temps neutre

Les 4 figures 2.3, 2.4, 2.5 et 2.6 confirment visuellement qu'il existe des tendances entre les principaux mouvements effectués par la main droite. On peut voir qu'il existe une forte variation entre chaque mouvement. Cependant, le squelette du mouvement reste le même entre les 9 graphiques de chaque figure.

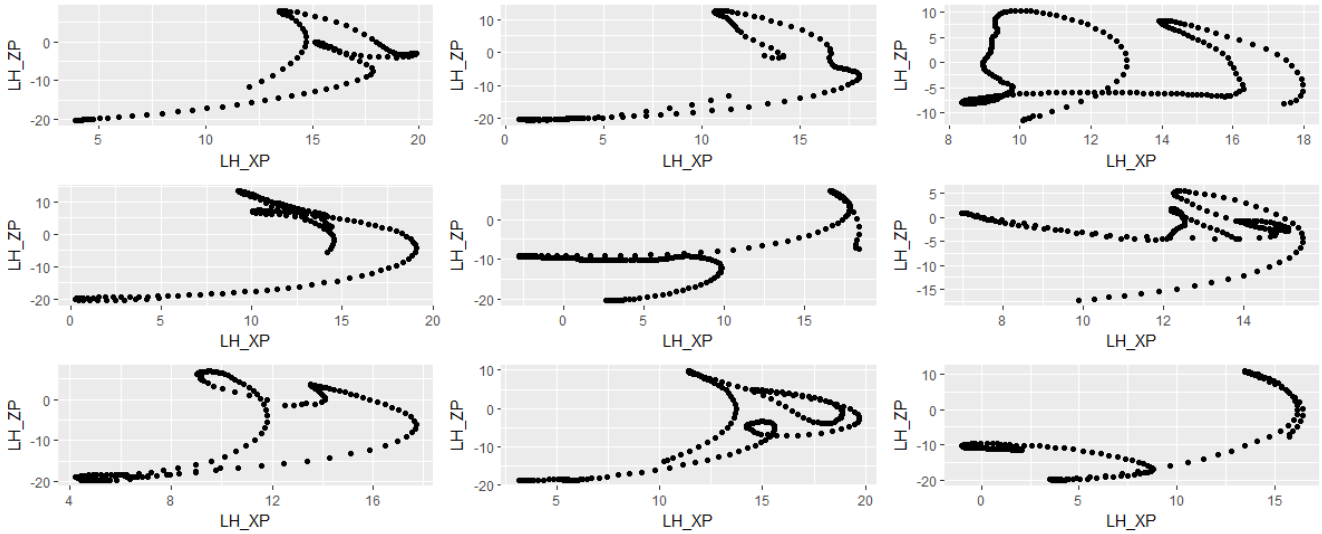


FIGURE 2.7 – Position de la main gauche en X en fonction de Z pour les 9 premiers mouvements de 4 temps neutre

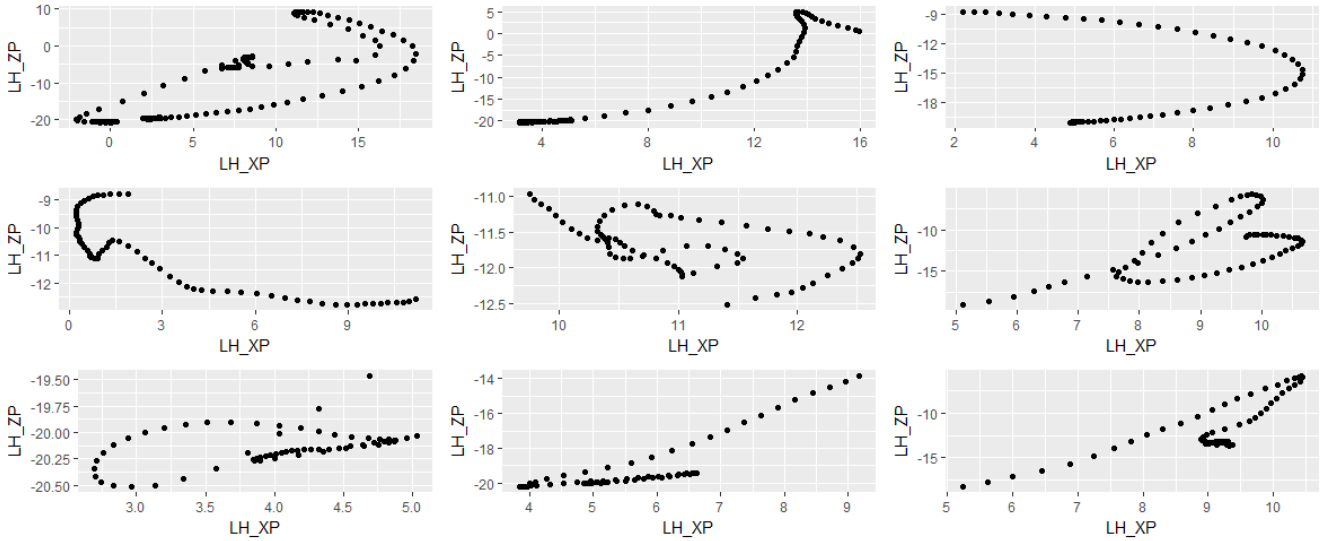


FIGURE 2.8 – Position de la main gauche en X en fonction de Z pour les 9 premiers mouvements de 2 temps neutre

Nous voyons graphiquement sur les figures 2.7 et 2.8, que les mouvements de la main gauche ne semblent pas répétitifs et semblent beaucoup moins prévisibles que les mouvements de la main droite.

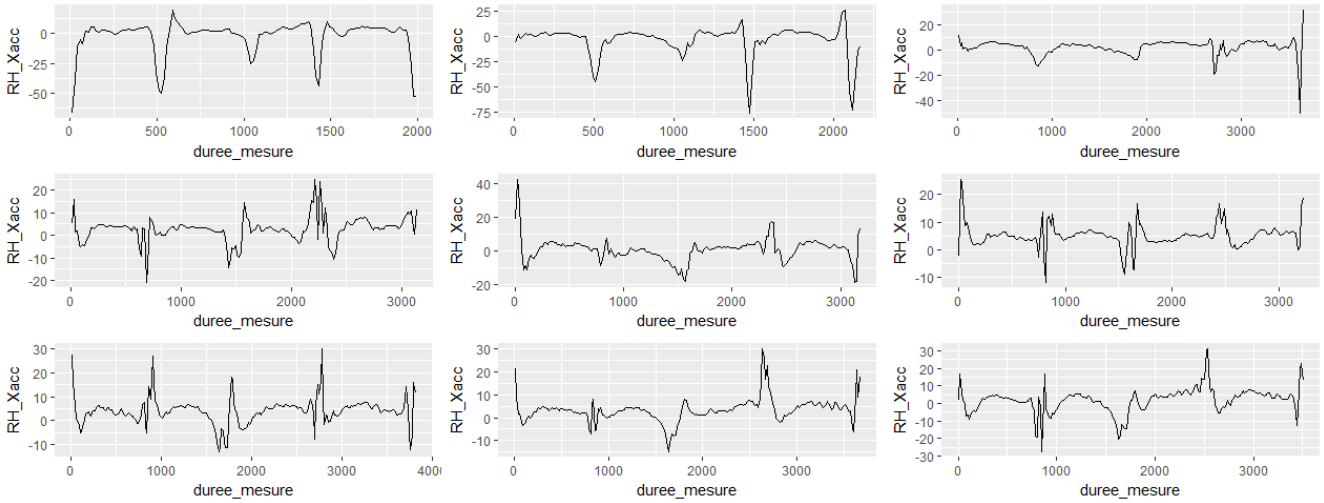


FIGURE 2.9 – Accélération de la main droite en X en fonction du temps pour les 9 premiers mouvements de 4 temps neutre

Nous voyons nettement sur la figure 2.9 que, pour chaque mouvement, il existe 4 pics d'accélérations et de ralentissements correspondant aux 4 temps du mouvement. Même si les graphiques ne sont pas à la même échelle, nous constatons une tendance entre les mouvements. Cela vient confirmer qu'il existe un patron pour le mouvement de la main droite. L'axe des X n'est pas égal sur les 9 graphiques. En fait, aucune des mesures n'est exactement égale à une autre en durée. La plus courte des mesures de 4 temps neutre dure 881,000 ms, c'est-à-dire moins d'une seconde, alors que la plus longue dure 8 945,33 ms, soit près de 9 secondes. Deux variables influencent principalement ce facteur : le tempo et la note de référence de la mesure. Le tempo, qui est indiqué en haut de la partition, est la pulsation naturelle ressentie par l'auditeur. Plus mathématiquement, c'est le nombre de notes de référence par minute contenu dans la musique. Plus la valeur du tempo est élevée, plus il y aura de notes de référence dans la musique et donc, généralement plus le rythme sera soutenu. Par exemple, si le tempo est à 60, cela signifie qu'il y a 60 pulsations par minute, donc une pulsation par seconde. Si le tempo est à 120, il y a une pulsation toutes les demi-secondes. Ainsi, le compositeur peut jouer avec ces paramètres ce qui a pour conséquence de faire durer plus ou moins la mesure. Plus la mesure sera longue, plus le mouvement du chef sera lent. Cela se traduit aussi au travers de la variation de l'accélération de mouvement mesurée. Lorsque la musique est plus lente, l'accélération et le ralentissement du mouvement du chef d'orchestre varient plus doucement car le chef a une plus grande durée pour réaliser l'ensemble de son mouvement.

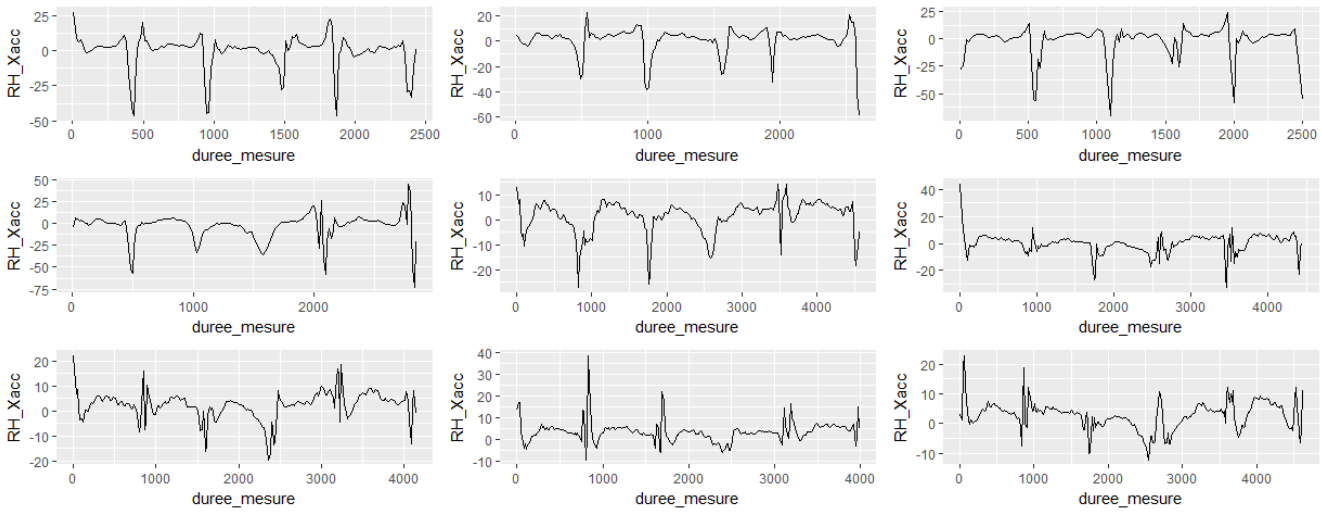


FIGURE 2.10 – Accélération de la main droite en X en fonction du temps pour les 9 premiers mouvements de 5 temps alterné 3+2

Ici nous voyons bien sur la figure 2.10 qu'il y a 5 pics sur les 9 graphiques. Cela correspond aux 5 temps du mouvement.

2.4.2 Corrélation des positions

Les figures 2.11, 2.12 et 2.13 nous montrent les différentes corrélations entre les capteurs pour les positions dans chaque direction X, Y et Z. Les tests de significativité sur les corrélations sont effectués avec un niveau de confiance de 99%. Les chiffres indiqués sur les graphiques correspondent au coefficient de corrélation. Lorsque le coefficient n'est pas significatif, il est marqué par une croix comme dans la figure 2.15. Les corrélations ont été obtenues avec la bibliothèque corrplot du logiciel R.

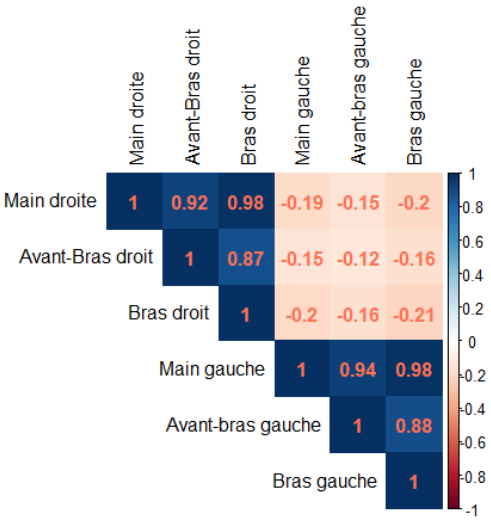


FIGURE 2.11 – Corrélation de Pearson de la position des 6 capteurs dans l'axe des X à travers l'ensemble du morceau

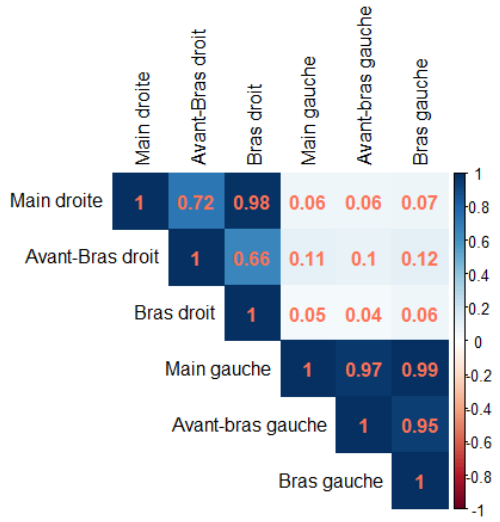


FIGURE 2.12 – Corrélation de Pearson de la position des 6 capteurs dans l'axe des Y à travers l'ensemble du morceau

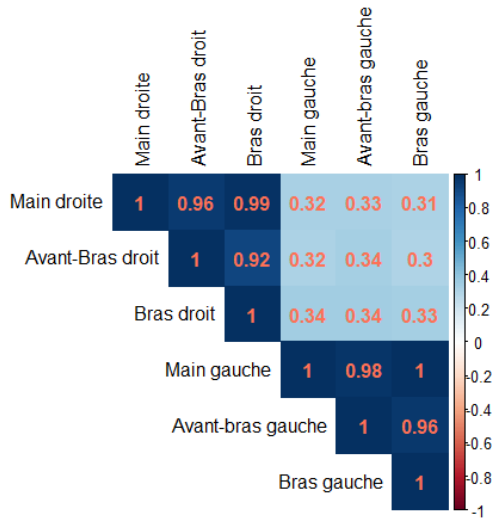


FIGURE 2.13 – Corrélation de Pearson de la position des 6 capteurs dans l'axe des Z à travers l'ensemble du morceau

Les figures 2.11, 2.12 et 2.13 montrent une tendance similaire. Les 3 capteurs du côté droit sont très corrélés entre eux. De même, les 3 capteurs du côté gauche sont très corrélés entre eux. Les corrélations sont toutes significatives à un niveau de confiance 99%. Les capteurs droits sont peu corrélés avec les capteurs gauches. Mais les corrélations sont significatives avec un niveau de confiance de 99%. Ces résultats confirment graphiquement le fait que les mains du chef d'orchestre ont deux rôles très différents et sont indépendantes l'une de l'autre. Dans les 3 cas, nous pouvons aussi remarquer que la main est très corrélée avec le bras (à droite comme à gauche). Pour l'axe X, la corrélation entre bras droit et main droite est de 98,39%. La corrélation entre bras gauche et main gauche est de 98,12%. Cependant la corrélation main/bras est plus forte que les corrélations avec l'avant-bras. En effet, la corrélation entre avant-bras et bras droit est de 87,15%. La corrélation entre avant-bras et main droite est de 92,05%. De même à gauche,

entre le bras et l'avant-bras, la corrélation est de 88,03%, entre l'avant-bras et la main, la corrélation est de 93,83%. Nous trouvons des résultats similaires pour l'axe Y et Z.

Les graphiques des corrélations des positions entre les capteurs pour chaque mouvement ne sont pas montrés ici car les résultats obtenus n'apportent pas d'information supplémentaire aux corrélations faites sur l'ensemble du morceau. Les résultats obtenus sont très similaires.

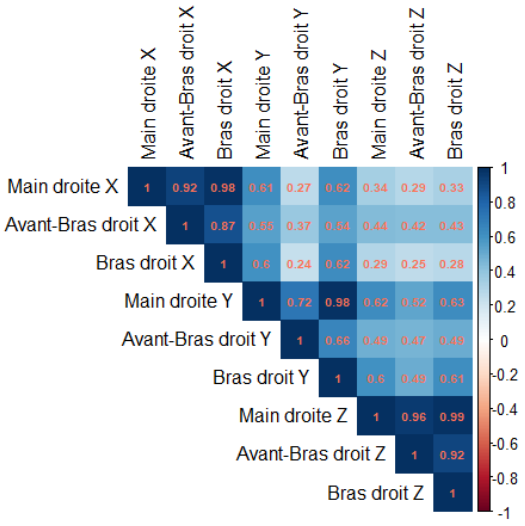


FIGURE 2.14 – Corrélation de Pearson de la position des 3 capteurs du côté droit dans les 3 directions (X, Y, Z) à travers l'ensemble du morceau

Nous voyons sur la figure 2.14 que les corrélations sont très élevées entre les capteurs d'une même direction. Par exemple en X, les corrélations s'étendent de 87,15% à 98,39%. Les carrés qui les représentent sont bleus foncés. Les corrélations sont plus faibles entre les capteurs de l'axe X/Y et Y/Z. Les couleurs sont plus proches du bleu océan Les corrélations entre les capteurs X/Z sont presque nulles. Elles sont comprises entre 25,14% et 43,70%. De plus, nous voyons que les corrélations entre les mains de 2 directions sont équivalentes aux corrélations main/bras de 2 directions différentes. Par exemple, la corrélation entre la main droite en X et Y est de 61,19%. Elle est équivalente à celle entre la main droite en X et le bras droit en Y qui est de 61,78% et à celle entre le bras droit en X et la main droite en Y qui est de 60,39%. Les corrélations avec l'avant-bras sont plus faibles, sauf pour le couple X/Z. Les corrélations sont toutes significatives à 99%. Ces résultats peuvent paraître étonnant au premier abord, ils seront approfondis dans la seconde partie de ce mémoire.

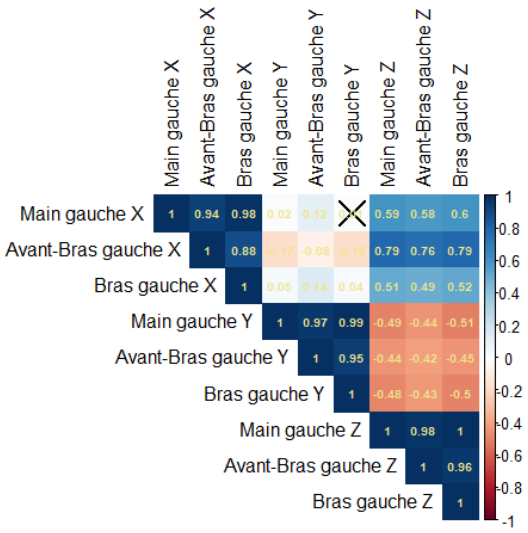


FIGURE 2.15 – Corrélation de Pearson de la position des 3 capteurs du côté gauche dans les 3 directions (X, Y, Z) à travers l'ensemble du morceau

Nous voyons que les corrélations du côté gauche, indiquées dans la figure 2.15, ont une tendance similaire à la tendance des corrélations du côté droit. Les corrélations entre les capteurs d'un même axe sont très élevées. Les corrélations entre Y/Z et X/Z sont plus faibles que pour le côté droit et les corrélations entre X/Y sont proches de zéro, alors que du côté droit les corrélations X/Z sont proches de zéro. Les corrélations restent toujours très significatives à 99%.

2.4.3 Corrélation des autres variables

Dans la section A.2 de l'annexe, nous trouvons tous les graphiques des corrélations correspondant à cette section. Pour l'accélération du côté droit, nous observons des corrélations assez faibles. Les corrélations les plus extrêmes se trouvent entre l'axe Y et Z. Les corrélations avec l'avant-bras sont plus élevées contrairement aux corrélations des variables de position examinées précédemment. A gauche, les corrélations sont extrêmement faibles. Les seules corrélations plus fortes sont celles réalisées entre variables du même axe et celle entre l'avant-bras gauche en X et le bras gauche en Y qui est presque égale à 1.

Pour la vitesse de rotation, dans l'axe des X, nous voyons que les corrélations sont plus fortes entre les capteurs qui sont situés les uns à côté des autres et du même côté. C'est-à-dire que les corrélations main/avant-bras et avant-bras/bras à droite et à gauche sont plus élevées que les autres. Pour l'axe Y, les capteurs sont corrélés entre eux pour un même côté, mais les corrélations gauche/droite sont faibles. En Z, les corrélations sont toutes élevées sauf pour celles avec la main droite. A droite, il y a une forte corrélation entre les variables prises en Y et en Z. Les autres sont peu corrélées. A gauche, comme dans l'axe des X, nous observons de la corrélation entre les capteurs qui sont juxtaposés.

2.5 Objectifs du mémoire

Ce mémoire répond à deux objectifs de recherche.

2.5.1 Inférence et données fonctionnelles

Nous rechercherons, dans un premier temps, d'où provient la variabilité des mouvements du chef d'orchestre lorsqu'il dirige. Nous regarderons à l'aide de techniques d'analyse de données fonctionnelles,

quels sont les capteurs qui sont le plus responsable de la variabilité des mouvements et nous quantifierons cette variabilité.

2.5.2 Prédiction et réseaux de neurones récurrents

Dans un deuxième temps, nous chercherons à prédire à partir des données issues des capteurs le type de mouvements que le chef réalise. Pour cela, nous utiliserons une technique de prévision qui prend en compte la temporalité des données. Cette technique utilise un modèle, basé sur le modèle de réseau de neurones, appelé réseau de neurones récurrents.

Chapitre 3

Inférence et données fonctionnelles

Le contenu théorique de ce chapitre est très largement tiré des chapitres 3, 4, 5 et 8 du livre de Ramsay et Silverman, Functional Data Analysis,(2005) (40). Les autres sources qui sont citées dans ce chapitre viennent éclairer la compréhension des théories exposées dans les chapitres de ce livre.

3.1 Présentation de nos données fonctionnelles

L'enregistrement des différentes variables du jeu de données dure 11 minutes et 45 secondes. Il est représenté par 44 467 valeurs discrètes pour chaque variable. Ces valeurs sont relevées grâce aux 6 capteurs qui ont été posés sur le chef d'orchestre. Elles sont relevées à une fréquence de 64 valeurs par seconde avec l'éventualité d'une imprécision de l'appareil utilisé de quelques millisecondes. Ces valeurs reflètent les variations fluides des variables car il s'agit de données représentant des mouvements. Ces dernières pourraient être évaluées, en théorie, aussi souvent que souhaité et donc nous pouvons qualifier ces variables de fonctions. Ainsi, notre jeu de données est constitué de 90 observations fonctionnelles : 6 de la position du chef d'orchestre selon l'axe X, Y, Z puis 6 de l'accélération des capteurs selon l'axe X, Y, Z puis 6 de la vitesse de rotation des capteurs selon l'axe X, Y, Z puis 6 du nord magnétique des capteurs selon l'axe X, Y, Z puis 6 des angles d'Euler (yaw, pitch, roll) comme expliqué dans le tableau 1.1.

3.2 Etudes antérieures

Très peu d'études statistiques ont été menées à ce jour sur les mouvements des chefs d'orchestre. Les études (28) et (30) portent sur les différences d'interprétation plus que sur l'origine de la variabilité du mouvement. Cependant, il existe un grand nombre d'articles et d'études disponibles sur d'autres types de mouvements tels que les mouvements des humains ou des animaux. La démarche est peut-être le mouvement le plus analysé car il est très répétitif et assez facile à analyser. Dans (51), Troje essaie de classer les sujets de son expérience en deux groupes : homme et femme. Il observe la démarche des personnes à l'aide de capteurs de mouvements et crée une représentation informatique de ces gestes pour permettre d'effectuer des analyses statistiques sur ces données. Ces analyses révèlent que la partie dynamique du mouvement contient plus d'informations sur le genre que les signaux structurels à mouvement. Le cadre proposé peut être utilisé non seulement pour l'analyse du mouvement biologique, mais aussi pour synthétiser de nouveaux modèles de mouvement. Un modélisateur de mouvement simple est présenté qui peut être utilisé pour visualiser et exagérer les différences dans les modèles de marche masculins et féminins.

La revue de littérature effectuée dans le chapitre 13 de (20) nous donne une bonne indication sur les différents types d'articles que nous pouvons lire sur les analyses de mouvements grâce aux données fonctionnelles. Ramsay et Silverman (40) ont utilisé des données de démarche pour expliquer certaines méthodes d'analyse de données fonctionnelles (FDA) telles que la corrélation canonique et l'analyse des

composantes principales fonctionnelles, notée ACP fonctionnelle. Depuis, les techniques de FDA ont été utilisées en biomécanique avec différentes applications. Par exemple, Ryan et al. (43) ont utilisé la FDA pour étudier le saut vertical. Donoghue et al. (19) a utilisé l'analyse par composantes principales fonctionnelle pour comparer différentes situations expérimentales lors d'une course en cas de lésion chronique du tendon d'Achille chez le sujet. Plusieurs auteurs ont utilisé diverses approches de l'ACP comme Sadeghi et al. (45) et Sadeghi (44) pour examiner la symétrie des membres inférieurs dans plusieurs études impliquant des sujets souffrant d'arthrose et des sujets sains, ou Deluzio et al. (31) et Daffertshofer et al. (13) ont reconnu la solution potentielle qu'une ACP pourrait fournir en analysant la coordination et le mouvement humain.

Le but de notre analyse est d'expliquer la variabilité du mouvement du chef d'orchestre à l'aide de technique d'ACP fonctionnelle. Ce chapitre fournit un rappel des notions fondamentales en analyse fonctionnelle, et décrit la méthode d'une ACP fonctionnelle que nous allons appliquer sur nos données.

3.3 Lissage des données

3.3.1 Modèle et concept de base

Nous observons n mesures d'une variable Y, notée $y_1, ..., y_n$ aux temps $t_1, ..., t_n$. Nous voulons représenter nos observations à l'aide d'une fonction $x(t)$. Nous cherchons donc à définir le modèle suivant :

$$y_j = x(t_j) + \epsilon_j, \text{j=1, ..., n.} \tag{3.1}$$

Un système de fonctions de base est un ensemble de fonctions connues ϕ_k qui sont mathématiquement indépendantes les unes des autres et qui peuvent définir correctement n'importe quelle fonction en prenant une somme pondérée ou une combinaison linéaire d'un nombre K de ces fonctions. Le système de fonctions de base le plus commun est la collection de monômes qui sont utilisés pour construire des séries de puissance définies par :

$$\phi_k(t) = t^k, \text{ k=0,1,2...}$$

Le système de séries de Fourier est probablement le système le plus couramment utilisé pour les données périodiques. Il est défini par :

$$\phi_{2k}(t) = sin(k\omega t) \text{ et } \phi_{2k+1}(t) = cos(k\omega t), \text{ k=0,1,2...}$$

où le paramètre ω détermine la période.

Le système de fonctions spline est le système le plus commun pour les séries non-périodiques. Il n'y a pas de périodicité dans les données utilisées pour ce mémoire. C'est pourquoi nous avons fait le choix d'utiliser ce système, qui sera décrit en détail dans la section 3.3.2. La fonction $x : \mathbb{R} \longrightarrow \mathbb{R}$ de l'équation (3.1) ci-dessus peut s'exprimer à l'aide de ce système de fonctions de bases de la façon suivante :

$$x(t) = \sum_{k=1}^{K} c_k \phi_k(t), \tag{3.2}$$

avec K fonctions de base connues ϕ_k. En posant $\mathbf{c}$ le vecteur de longueur K des coefficients c_k tel que $c = (c_1, ..., c_k)$ et ϕ comme le vecteur fonctionnel dont les éléments sont les fonctions de base ϕ_k tel que $\phi = (\phi_1, ..., \phi_k)$, nous pouvons exprimer aussi l'équation précédente en notation matricielle :

$$x = \mathbf{c}'\phi = \phi'\mathbf{c}. \tag{3.3}$$

Une représentation exacte ou *interpolation* est obtenue lorsque K = n, dans notre cas n=44 467, c'est-à-dire lorsque nous choisissons les coefficients c_k pour produire exactement $x(t_j) = y_j$ pour chaque j.

Par conséquent, le degré auquel les données y_j sont *lissées* par opposition à *interpolées* est déterminé par le nombre K de fonctions de base. Ainsi, nous ne voyons pas un tel système en fonction d'un nombre fixe de paramètres K, mais nous voyons plutôt K comme un paramètre que nous choisissons selon les caractéristiques des données. Nous reviendrons sur le concept d'interpolation des données dans le chapitre 3 lorsque nous effectuerons une mise en forme des données pour notre modèle de prédiction. Dans les sections suivantes, nous examinerons la structure d'une fonction spline puis décrirons le système B-spline, qui est le système couramment utilisé pour construire ce genre de fonction.

3.3.2 Définition d'une fonction spline

Tout d'abord, pour définir une fonction spline il faut diviser l'intervalle sur lequel la fonction est définie en L sous-intervalles séparés par des valeurs τ_l, l=1,...,L-1, qui sont appelés points de rupture. Sur chaque intervalle défini par 2 points de rupture, une fonction spline est un polynôme d'ordre m spécifié. L'ordre d'un polynôme est le nombre de constantes nécessaires pour le définir, et correspond à une unité supplémentaire par rapport à son degré c'est-à-dire sa plus haute puissance. Les polynômes adjacents se rejoignent au point de rupture qui les sépare en splines distincts, d'ordre supérieur à 1, pour que les valeurs des fonctions soient contraintes d'être égales au point de leur jonction. Le moyen le plus facile de gagner en flexibilité et en précision dans une spline est d'augmenter le nombre de points de rupture. Il est possible de vouloir plus de points de rupture sur les régions où la fonction présente le plus de variations complexes, et moins où la fonction est seulement légèrement non linéaire. Une considération secondaire est que nous ne voulons pas d'intervalles qui ne contiennent pas de données, cependant cela semble logique puisque nous ne pouvons pas espérer capturer les fonctionnalités d'une fonction sans données. La figure 3.1 illustre le concept de points de rupture. La ligne continue définit une fonction spline d'un ordre donné (2, 3 ou 4) qui correspond à la fonction sinus affichée en pointillés. Les lignes pointillées verticales sont les points de rupture intérieurs définissant les jonctions entre les splines.

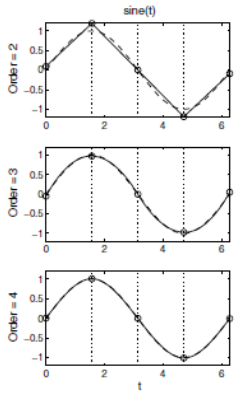


FIGURE 3.1 – Illustration des points de rupture d'une fonction spline et de l'ordre des polynômes, figure 3.4 de (40)

Dans notre cas, les points de rupture sont à égale distance. En effet, les données présentent des variations importantes sur l'ensemble du jeu de données. De plus, les données sont réparties à égale distance dans le temps. Il n'y a donc pas de risque d'avoir d'intervalles sans données. Il n'y a donc aucune raison de répartir irrégulièrement les points de rupture. Ainsi, une fonction spline est déterminée par deux choses : l'ordre des segments polynomiaux, et le nombre de points de rupture τ. Or comme vu précédemment τ = L-1. Donc le nombre de paramètres nécessaire pour définir une fonction spline est l'ordre des polynômes, m, plus le nombre de points de rupture intérieurs, L-1 :

Nombre de paramètres *fonction B-spline* = m + L - 1.

Nous donnons maintenant les éléments qui permettent de construire une telle fonction.

3.3.3 Construction d'une fonction spline

Nous avons, tout d'abord, besoin d'un système de fonctions de base $\phi_k(t)$, et celles-ci auront les propriétés essentielles suivantes :

- chaque fonction de base $\phi_k(t)$ est une fonction spline définie par un ordre m et une séquence de points de rupture τ.
- comme un multiple d'une fonction spline est une fonction spline, et que les sommes et les différences de splines sont également des splines, toute combinaison linéaire de ces fonctions de base reste une fonction spline.
- toute fonction spline définie par m et τ peut être exprimée comme une combinaison linéaire de ces fonctions de base.

La notation $B_k(t, \tau)$ est souvent utilisée pour indiquer la valeur de la fonction B-spline de base au temps t définie par la séquence de points de rupture τ. Ici k représente l'index du point de rupture le plus élevé qui se situe exactement ou immédiatement à gauche de la valeur t. Cette notation nous donne m + L - 1 fonctions de base, comme requis dans le cas habituel où tous les points de rupture intérieurs sont discrets. Si l'on se réfère à l'équation (3.2), alors K= m + L - 1. Selon cette notation, une fonction spline S(t) avec des points de rupture intérieurs discrets est définie comme :

$$S(t) = \sum_{k=1}^{m+L-1} c_k B_k(t, \tau).$$

Une caractéristique qui peut paraître étonnante des splines est que l'augmentation de la valeur du nombre de fonctions de base K n'améliore pas toujours certains aspects du lissage. Cela s'explique par le fait que, lorsque l'ordre m d'une spline est fixe, l'espace de la fonction définie par K B-splines n'est pas nécessairement contenu dans celui défini par K + 1 B-splines. Les effets compliqués dus à l'espacement des points de rupture par rapport à l'échantillonnage des points peuvent mener un système B-spline de dimension inférieure à produire de meilleurs résultats qu'un système de plus grande dimension. Donc la question se pose du choix du nombre de fonctions de base. Comment choisir le bon K ? Car pour un K élevé, le lissage sera très ajusté aux données mais conservera du bruit ou de la variation que l'on cherche à éliminer. Si le K est trop faible, certains aspects de la fonction que l'on cherche à estimer seront probablement occultés.

Avant de poursuivre notre développement, nous rappelons rapidement l'origine de l'erreur quadratique moyenne. Cette erreur sera une mesure essentielle dans la construction du lissage des données. Ainsi, comprendre comment cette mesure est composée nous permettra de mieux comprendre comment l'influencer lors de la construction de nos fonctions lissées.

3.3.4 Rappel sur l'erreur quadratique moyenne (ou MSE)

L'erreur quadratique moyenne de l'estimation de $y(t)$ comme dans (3.1) ou MSE se définit par :

$$\text{MSE}[x(t)] = \frac{1}{n} \sum_{j=1}^{n} (y_j(t) - x_j(t))^2. \tag{3.4}$$

Notre but est de minimiser cette mesure lors du lissage des données. Cependant, dans la plupart des cas, nous ne pouvons pas minimiser cela car nous n'avons aucun moyen de savoir ce qu'est $y(t)$ sans utiliser les données. Cependant, nous pouvons réécrire le MSE en fonction du biais et de la variance comme ceci :

$$\text{MSE}[x(t)] = \text{Biais}^2[x(t)] + \text{Var}[x(t)].$$

Le biais de l'estimation de $y(t)$ se définit par :

$$\text{Biais}[x(t)] = y(t) - E[x(t)],$$

et donc logiquement plus le nombre de fonctions de base K sera grand, plus le biais sera petit. En effet, lorsque K = n, le lissage des données est équivalent à de l'interpolation. La courbe estimée $x(t)$ et la courbe d'origine $y(t)$ sont identiques et donc le biais est nul. Plus le nombre de fonctions de base est petit, plus $y(t)$ et $x(t)$ seront différents et donc plus le biais sera grand. Le biais nous pousse donc à favoriser un nombre élevé K de fonctions de base.

L'un des buts principaux du lissage est de réduire l'influence du bruit sur l'estimation x et donc de diminuer au maximum la valeur de l'écart-type de l'estimation, $\sqrt{Var(x)}$, qui mesure ce bruit. La variance de l'estimation est définie par :

$$\text{Var}[x(t)] = E[(x(t) - E[x(t)])^2].$$

Lorsque K = n, c'est-à-dire lorsque la courbe estimée $x(t)$ et la courbe d'origine y(t) sont identiques, les écarts à la moyenne seront sûrement très élevés et la variance de $x(t)$ sera certainement très élevée. Nous voulons donc réduire la variance pour réduire le bruit, ce qui nous amène à chercher des valeurs plus petites de K. Ainsi, contrairement au biais, la variance nous amène à chercher un faible nombre K de fonctions de base. Nous voyons donc que le biais et la variance fonctionnent dans les directions opposées. Un petit nombre K permettrait d'avoir des valeurs de variance raisonnable mais conduirait le biais à avoir des valeurs très hautes et donc inacceptables. Il faut donc chercher un juste-milieu entre les deux. Le rapport entre $y(t)$ et l'écart-type de l'estimation $x(t)$, plus communément appelé rapport signal / bruit (RSB), peut être noté $y(t)/\sqrt{Var(x(t))}$. Ce rapport permet de mesurer le déséquilibre entre la fonction d'origine (signal) $y(t)$ et le bruit lié à l'estimation $x(t)$. Notre but est donc d'avoir un RSB équilibré, c'est-à-dire d'avoir un niveau de bruit raisonnable pour un signal donné $y(t)$. Tant qu'un équilibre raisonnable ne sera pas trouvé, nous tenterons d'influencer ce ratio donc de contrôler la variance plutôt que le biais, pour ainsi réguler le nombre de fonctions K en fonction de la variance et non du biais. Acceptant un peu de biais, nous pouvons obtenir une réduction importante de la variance dans l'équation (3.5). En pratique, c'est presque toujours le cas, et ceci est la raison fondamentale du lissage des données afin d'estimer des fonctions de base.

Dans la suite, nous utiliserons deux techniques différentes qui permettent de choisir K. Tout d'abord, nous expliquerons le choix de K en fonction du critère des moindres carrés puis ensuite, nous utiliserons une méthode plus sophistiquée : la régularisation.

3.3.5 Le critère des moindres carrés

Nous venons d'expliquer ce qu'est l'erreur quadratique moyenne. Nous utiliserons ensuite une mesure directement dérivée du MSE : la somme des carrés des erreurs, notée SSE. Le SSE se définit par :

$$\text{SSE}[x(t)] = \sum_{j=1}^{n} (y_j(t) - x_j(t))^2,$$

donc d'après (3.4), nous pouvons écrire :

$$\text{SSE}[x(t)] = n * MSE.$$

Un lissage linéaire simple est obtenu si nous déterminons les coefficients linéaires c_k de l'équation (3.2) qui minimisent le critère des moindres carrés tels que :

$$\text{SSE}(\mathbf{y}|\mathbf{c}) = \sum_{j=1}^{n}[y_j - \sum_{k}^{K} c_k\phi_k(t_j)]^2.$$

Si nous définissons K matrices Φ contenant les valeurs $\phi_k(t_j)$, alors le SSE peut être exprimé plus simplement en termes matriciels tel que :

$$\text{SSE}(\mathbf{y}|\mathbf{c}) = (\mathbf{y} - \mathbf{\Phi c})'(\mathbf{y} - \mathbf{\Phi c}). \tag{3.5}$$

La solution de l'équation (3.5) en termes matriciels, notée $\hat{\mathbf{c}}$, et qui minimise le critère des moindres carrés est donnée par :

$$\hat{\mathbf{c}} = (\mathbf{\Phi'\Phi})^{-\mathbf{1}}\mathbf{\Phi'y},$$

où la matrice Φ, de taille n par K contient les valeurs des K fonctions de base aux n points d'échantillonnage, et où $\mathbf{y}$ est le vecteur de données discrètes à lisser. Le vecteur d'ajustement aux données s'exprime ainsi :

$$\hat{\mathbf{y}} = \mathbf{\Phi}(\mathbf{\Phi'\Phi})^{-\mathbf{1}}\mathbf{\Phi'y} = \mathbf{S}_\phi\mathbf{y},$$

où $\mathbf{S}_\phi$ est le projecteur :

$$\mathbf{S}_\phi = \mathbf{\Phi}(\mathbf{\Phi'\Phi})^{-\mathbf{1}}\mathbf{\Phi'},$$

correspondant au système de base ϕ.

3.3.6 La régularisation ou pénalité de lissage

La pénalité de lissage, également appelée approche de régularisation, est une technique d'estimation générale différente des moindres carrés. Elle conserve les avantages des techniques utilisant simplement les fonctions de base et la combinaison linéaire pour lisser les données, mais contourne certaines de leurs limites. Cette technique sera la technique utilisée au cours de ce travail.

Les méthodes de pénalité de lissage sont basées sur l'optimisation des variations de la courbe estimée à l'aide d'un critère de pénalité. La méthode de lissage spline estime une courbe x à partir des observations $y_j = x(t_j) + \epsilon_j$ avec pour objectif de base de minimiser le MSE. Nous avons vu que le MSE de l'estimation peut souvent être considérablement réduit en sacrifiant le biais. Dans ce cas, nous obtiendrons une estimation des données plus éloignée des données réelles, ce qui permettra de réduire la variance, c'est-à-dire des variations de la courbe trop fortes. Notre but est donc que la courbe estimée ne soit pas trop éloignée de la courbe réelle sans avoir une trop grande variance c'est-à-dire de trop grandes variations entre les données estimées. Nous voulons que l'estimation de la courbe varie en douceur d'une valeur à l'autre. Pour cela, nous supposons une certaine régularité dans la fonction x que nous essayons d'estimer. Grâce à cette régularité, l'estimation devient plus stable, la courbe est moins variable aux dépens d'une certaine augmentation du biais. La pénalité de lissage permet d'imposer cette régularité et rend explicite ce que nous sacrifions en biais pour atteindre une amélioration du MSE.

3.3.7 Quantifier la pénalité de lissage

Une façon commune de quantifier la fonction de pénalité d'une fonction est d'utiliser la dérivée seconde de la fonction $x(t)$. La dérivée seconde indique la variation de la pente :

- si elle est positive sur un intervalle, la pente augmente, la courbure est vers le haut, la fonction est dite « convexe » sur cet intervalle ;
- si elle est négative sur un intervalle, la pente diminue, la courbure est vers le bas, la fonction est dite « concave » sur cet intervalle ;
- si elle est nulle, la courbe est localement rectiligne ;
- si la dérivée seconde s'annule et change de signe, nous avons un point d'inflexion, la courbure de la courbe s'inverse.

Ainsi, le carré de la dérivée seconde, noté $[D^2x(t)]^2$, d'une fonction au temps t est souvent appelé la courbure en t de la fonction. Par conséquent, une mesure logique de la fonction de pénalité est la dérivée seconde intégrée au carré :

$$\text{PEN}_2(x) = \int [D^2x(s)]^2 ds.$$

Des fonctions très variables peuvent donner des valeurs élevées de $\text{PEN}_2(x)$ parce que leurs secondes dérivées sont grandes sur au moins une partie de l'intervalle d'intérêt. Nous pouvons donc généraliser la fonction de pénalité en permettant une dérivée $D^m x$ d'ordre arbitraire afin de travailler avec la pénalité :

$$\text{PEN}_m(x) = \int [D^m x(s)]^2 ds.$$

Nous modifions maintenant le SSE obtenu en (3.5) afin d'inclure la pénalité $\text{PEN}_2(x)$ dans la définition de *x(s)*. Soit $x(\mathbf{t})$ le vecteur résultant de la fonction x évaluée au vecteur $\mathbf{t}$ composé des valeurs de cet argument. Nous définissons un compromis entre le lissage et l'ajustement des données en définissant le critère *pénalisé* des moindres carrés par :

$$\begin{aligned}\text{PENSSE}_\lambda(x|\mathbf{y}) &= SSE + \lambda \times \text{PEN}_2(x), \\ &= (\mathbf{y} - x(\mathbf{t}))'(\mathbf{y} - x(\mathbf{t})) + \lambda \times \text{PEN}_2(x).\end{aligned}$$

L'estimation de la fonction est obtenue en trouvant la fonction x qui minimise $\text{PENSSE}_\lambda(x)$ sur l'espace des fonctions x pour lesquelles $\text{PEN}_2(x)$ est définie. Le paramètre λ est un *paramètre de lissage* qui mesure concrètement l'arbitrage entre l'ajustement aux données, tel que mesuré par le critère des moindres carrés dans le premier terme, et la variabilité de la fonction x, telle que quantifiée par $\text{PEN}_2(x)$ dans le second terme. Puisque la dérivée seconde mesure la variabilité d'une fonction, plus une fonction sera variable et non-linéaire, plus le carré de la dérivée sera élevé et donc plus $\text{PEN}_2(x)$ sera élevé. Donc lorsque λ augmente, le terme $\lambda \times \text{PEN}_2(x)$ augmente fortement pour les fonctions qui ne sont pas linéaires. Par conséquent le poids du lissage de x dans $\text{PENSSE}_\lambda(x)$ augmente au dépend de l'ajustement des données représenté par le SSE de (3.5). Pour cette raison, lorsque $\lambda \to \infty$ la courbe ajustée x tend vers la régression linéaire standard des données observées, cas où la dérivée seconde est nulle et donc $\text{PEN}_2(x) = 0$. D'autre part, pour un λ faible, la courbe a tendance à devenir de plus en plus variable car la pénalité de lissage est faible, et lorsque $\lambda \to 0$ la courbe x s'approche d'une interpolation des données, vérifiant l'équation $x(t_j) = y_j$ pour tous les j.

3.3.8 Comment estimer les coefficients de lissage ?

Nous utilisons dès à présent certaines des équations vues précédemment pour comprendre comment la pénalisation modifie le processus de lissage. Rappelons que, sans pénalité de lissage, $x(t)$ est défini par :

$$x(t) = \sum_k^K c_k \phi_k(t) = \mathbf{c}'\boldsymbol{\phi}(t) = \boldsymbol{\phi}'(t)\mathbf{c}, \tag{3.6}$$

où $\mathbf{c}$ est le vecteur de taille K des coefficients et ϕ est le vecteur de taille K de fonctions de base. La solution de (3.6), notée $\hat{\mathbf{c}}$, et qui minimise le critère des moindres carrés (3.5), est donnée par :

$$\hat{\mathbf{c}} = (\boldsymbol{\Phi}'\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}'\mathbf{y},$$

où la matrice $\mathbf{\Phi}$, de taille n, par K contient les valeurs des K fonctions de base aux n points d'échantillonnage, et où $\mathbf{y}$ est le vecteur de données discrètes à lisser. Le vecteur d'ajustement des données s'exprime ainsi :

$$\hat{\mathbf{y}} = \mathbf{\Phi}(\mathbf{\Phi'\Phi})^{-1}\mathbf{\Phi'y} = \mathbf{S}_\phi \mathbf{y},$$

où $\mathbf{S}_\phi$ est le projecteur :

$$\mathbf{S}_\phi = \mathbf{\Phi}(\mathbf{\Phi'\Phi})^{-1}\mathbf{\Phi'}, \tag{3.7}$$

correspondant au système de base ϕ. Nous pouvons ré-exprimer la pénalité de lissage $\mathrm{PEN_m(x)}$ en termes matriciels :

$$\begin{aligned}
\mathrm{PEN}_m(x) &= \int [D^m x(s)]^2 ds, \\
&= \int [D^m \mathbf{c}'\boldsymbol{\phi}(s)]^2 ds, \\
&= \int \mathbf{c}' D^m \boldsymbol{\phi}(s) D^m \boldsymbol{\phi}'(s) \mathbf{c} ds, \\
&= \mathbf{c}' [\int D^m \boldsymbol{\phi}(s) D^m \boldsymbol{\phi}'(s) ds] \mathbf{c}, \\
&= \mathbf{c}' \mathbf{R} \mathbf{c},
\end{aligned}$$

où nous posons :

$$\mathbf{R} = \int D^m \boldsymbol{\phi}(s) D^m \boldsymbol{\phi}'(s) ds.$$

En ajoutant la somme des carrés des erreurs $\mathrm{SSE(y|c)}$ et en multipliant $\mathrm{PEN_m(x)}$ par un paramètre de lissage λ, nous obtenons :

$$\mathrm{PENSSE}_m(\mathbf{y}|\mathbf{c}) = (\mathbf{y} - \mathbf{\Phi c})'(\mathbf{y} - \mathbf{\Phi c}) + \lambda \mathbf{c'Rc}. \tag{3.8}$$

Afin de trouver le $\mathbf{c}$ qui minimise le SSE pénalisé, nous prenons la dérivée par rapport au vecteur $\mathbf{c}$, et nous obtenons :

$$2\mathbf{\Phi\Phi'c} - 2\mathbf{\Phi'y} + \lambda \mathbf{Rc} = 0.$$

A partir de cela, nous obtenons l'expression pour le vecteur de coefficient estimé :

$$\hat{\mathbf{c}} = (\mathbf{\Phi'\Phi} + \lambda \mathbf{R})^{-1}\mathbf{\Phi'y}.$$

Finalement, l'expression pour le vecteur d'ajustement des données $\hat{\mathbf{y}}$ est :

$$\hat{\mathbf{y}} = \mathbf{\Phi}(\mathbf{\Phi'\Phi} + \lambda \mathbf{R})^{-1}\mathbf{\Phi'}\boldsymbol{y} = \mathbf{S}_{\phi,\lambda}\mathbf{y},$$

où la matrice "chapeau" symétrique d'ordre n est :

$$\mathbf{S}_{\phi,\lambda} = \mathbf{\Phi}(\mathbf{\Phi'\Phi} + \lambda \mathbf{R})^{-1}\mathbf{\Phi'}. \tag{3.9}$$

La comparaison de cette expression avec (3.7) nous montre que le seul changement observable est l'ajout de $\lambda\mathbf{R}$ à la matrice $\mathbf{\Phi'\Phi}$ avant son inversion, et que les deux expressions deviennent identiques lorsque $\lambda = 0$. Le projecteur qui représente le cas le plus général (3.9) peut être appelé *sous-projecteur* car, contrairement au projecteur, la sous-projection n'est pas idempotente, c'est-à-dire qu'elle n'a pas le même effet lorsque l'on réalise une ou plusieurs fois l'opération :

$$\mathbf{S}_{\phi,\lambda}\ \mathbf{S}_{\phi,\lambda} \neq \mathbf{S}_{\phi,\lambda}.$$

De plus, $\mathrm{S}_{\phi,\lambda}$ permet aussi de calculer le degré de liberté de la fonction spline lissée :

$$df(\lambda) = trace\mathbf{S}_{\phi,\lambda}. \tag{3.10}$$

3.3.9 Comment choisir le bon paramètre de lissage

Nous expliquons maintenant étape par étape comment trouver le paramètre de lissage optimal.

Trouver le λ maximal

Il existe une règle d'or pour définir la valeur λ maximale pour un nombre fixé K de fonctions de base (40) . La taille de $\lambda\mathbf{R}$ ne doit pas dépasser 10^{10} multiplié par la taille de $\mathbf{\Phi'\Phi}$. Ainsi, nous avons calculé pour différentes valeurs K de fonctions de base la valeur maximale que λ pourrait prendre. Nous avons, dans un premier temps, calculé les normes de $\|\mathbf{\Phi'\Phi}\|$ et de $\|\mathbf{R}\|$:

$$\|\mathbf{R}\| = \sqrt{\sum_k \sum_l r_{kl}^2}.$$

Puis, nous effectuons le calcul suivant :

$$\lambda \leq \frac{10^{10} \times \|\mathbf{\Phi'\Phi}\|}{\|\mathbf{R}\|}.$$

Nous avons donc un calcul qui nous permet de trouver le λ maximal pour un nombre donné de fonctions de base. Cependant, nous sommes intéressés ici à savoir quel est le λ qui minimise le critère (3.8). Pour cela, nous verrons une technique appelée validation croisée généralisée.

Trouver le λ avec la validation croisée généralisée

L'idée de base de la validation croisée classique est de séparer le jeu de données en k échantillons. Parmi ces k échantillons, nous en sélectionnons un que nous appelons *échantillon de validation*, qui servira à évaluer l'erreur quadratique du modèle estimé. Les *k-1* autres échantillons forment l'*échantillon d'apprentissage* qui permet d'entraîner le modèle. Si nous avons séparé le jeu de données en k échantillons de base, alors cette manipulation est répétée k fois. Chaque échantillon deviendra à tour de rôle un échantillon de validation. La moyenne des k erreurs quadratiques moyennes est calculée à la fin pour évaluer l'erreur de prédiction du modèle. Cette technique peut être poussée à l'extrême afin de permettre le choix d'un paramètre de lissage. Au lieu de laisser un échantillon composé de plusieurs observations, nous ne laissons qu'une seule observation pour former l'échantillon de validation, et donc il reste *n-1* points de données pour estimer le modèle. L'erreur du modèle est estimée sur une seule observation à la fois. Un total de n lissages est donc réalisé et la moyenne des erreurs quadratiques est calculée à la fin. Cette opération est réalisée pour plusieurs valeurs du paramètre de lissage λ, et nous choisissons le λ qui donne l'erreur minimum. Une des hypothèses de la validation croisée est que les observations sont relativement indépendantes les unes des autres.

Cependant, cette méthode pose deux problèmes. D'abord, elle nécessite beaucoup de calculs, ce qui est difficilement faisable lorsque nous avons des échantillons avec des milliers d'observations, comme dans notre cas. Le deuxième problème est que minimiser l'erreur de la validation croisée peut conduire à un lissage trop faible des données parce que cette méthode a tendance à s'ajuster à des variations très bruyantes des données ou qui ont des amplitudes très élevées que nous souhaiterions ignorer.

Pour faire face à ce problème, il existe une procédure, développée par Craven et Wahba (1979), appelée validation croisée généralisée, notée GCV pour *Generalized Cross-Validation*. Elle a été développée à l'origine comme une version plus simple de la procédure de validation croisée pour éviter de lisser n fois. Elle a également été jugée plus fiable que la validation croisée car elle a moins tendance à suivre les variations bruyantes de la courbe d'origine. Nous cherchons le λ qui va minimiser le critère de la GCV, défini comme suit :

$$GCV_\lambda = \frac{n^{-1}\text{PENSSE}_m}{[n^{-1}trace(\mathbf{I} - \mathbf{S}_{\phi,\lambda})]^2}.$$

Nous pouvons réécrire l'équation du GCV ainsi :

$$\begin{aligned} GCV_\lambda &= \frac{n^{-1}\text{PENSSE}_m}{[n^{-1}trace(\mathbf{I}-\mathbf{S}_{\phi,\lambda})]^2}, \\ &= \frac{n^{-1}}{[n^{-1}trace(\mathbf{I}-\mathbf{S}_{\phi,\lambda})]} \times \frac{\text{PENSSE}_m}{[n^{-1}trace(\mathbf{I}-\mathbf{S}_{\phi,\lambda})]}. \end{aligned}$$

Or, trace $(\mathbf{I} - \mathbf{S}_{\phi,\lambda})$ = trace $(\mathbf{I})$ − trace$(\mathbf{S}_{\phi,\lambda})$, et donc :

$$GCV_\lambda = \frac{1}{[trace(\mathbf{I}) - trace(\mathbf{S}_{\phi,\lambda})]} \times \frac{\text{PENSSE}_m}{[n^{-1}(trace(\mathbf{I}) - trace(\mathbf{S}_{\phi,\lambda}))]}.$$

De plus, nous notons λ_i les valeurs propres de la matrice identité $\mathbf{I}$, donc :

$$Trace(I) = \sum_{i=1}^{n} \lambda_i = n,$$

car toutes les valeurs propres de la matrice identité sont égales à 1. En utilisant l'équation (3.10) qui définit $df(\lambda)$ mesurant le degré de liberté d'une fonction spline lissée, nous obtenons finalement :

$$\begin{aligned} GCV_\lambda &= \frac{1}{[n - df(\lambda)]} \times \frac{\text{PENSSE}_m}{[n^{-1}(n - df(\lambda))]}, \\ &= \frac{n}{n - df(\lambda)} \times \frac{\text{PENSSE}_m}{n - df(\lambda)}. \end{aligned}$$

Nous venons de définir le critère du GCV qui nous permet de trouver le λ optimal. Nous allons regarder dans la section suivante les résultats de l'application aux données de capteurs présentées dans ce mémoire.

3.4 Résultats et interprétation

Nous analysons maintenant les données que nous avons présentées dans le premier chapitre. Nous effectuons un lissage de ces données pour ensuite réaliser des analyses plus sophistiquées sur les données telle que l'analyse par composantes principales fonctionnelles que nous verrons par la suite. Nous avons tout d'abord lissé les données variable par variable. Les variables de position, d'accélération, de vitesse de rotation et d'angle ont été lissées pour les 6 capteurs dans les 3 directions X, Y et Z (sauf pour les angles où il n'y a pas de direction). Seulement certains des résultats sont montrés dans ce mémoire, les autres sont présentés en annexes. Pour construire le lissage des données, nous cherchons d'abord le nombre de fonctions de base K et la pénalité de lissage λ optimum. Nous procédons par étapes en essayant, dans un premier temps, de se donner un ordre d'idée pour les valeurs maximum de K et λ. Ensuite, nous serons capable de restreindre nos recherches pour déterminer un K et un λ optimum pour le lissage de nos données. Nous avons réalisé ses analyses avec la bibliothèque FDA du logiciel de programmation R. Les visualisations sont faites avec la bibliothèque ggplot2.

3.4.1 Recherche d'un λ maximum

Dans le tableau 3.1, nous pouvons voir les valeurs maximales du paramètre de lissage que l'on pourrait avoir pour nos données en fonction d'un nombre de fonctions de base fixé. Par exemple, si nous choisissons de prendre 100 fonctions de base alors la pénalité de lissage λ ne devra pas excéder 1,46e+18.

TABLE 3.1 – Valeur maximale du paramètre de lissage en fonction du nombre de fonctions de base

Nombre de fonctions de bases K	$\|R\|$	$\|\Phi'\Phi\|$	λ_{max}
100	6,48e-07	107,31	1,65e18
200	6,19e-06	154,93	2,50e17
300	2,35e-05	192,67	8,18e16
400	6,12e-05	224,73	3,66e16
500	1,29e-04	252,11	1,94e16
600	2,38e-04	277,15	1,16e16
700	4,02e-04	303,61	7,54e15
800	6,32e-04	326,85	5,16e15
900	9,44e-04	346,32	3,66e15
1 000	1,35e-03	362,48	2,67e15
1 100	1,87e-03	382,31	2,04e15
1 200	2,52e-03	400,59	1,58e15
1 300	3,32e-03	418,85	1,26e15
1 400	4,28e-03	434,10	1,01e15
1 500	5,42e-03	449,19	8,27e14
1 600	6,77e-03	461,50	6,80e14
1 700	8,35e-03	475,15	5,68e14
1 800	1,01e-02	489,13	4,80e14
1 900	1,22e-02	502,20	4,09e14
2 000	1,46e-02	516,52	3,53e14
2 100	1,73e-02	529,84	3,06e14
2 200	2,03e-02	541,82	2,66e14
2 300	2,37e-02	553,29	2,33e14
2 400	2,74e-02	568,30	2,06e14
2 500	3,16e-02	580,48	1,83e14
2 600	3,62e-02	591,30	1,63e14
2 700	4,12e-02	602,40	1,45e14
2 800	4,68e-02	615,59	1,31e14
2 900	5,28e-02	625,92	1,18e14
3 000	5,94e-02	637,63	1,07e14
3 100	6,66e-02	646,10	9,69e13
3 200	7,43e-02	653,75	8,78e13
3 300	8,27e-02	663,03	8,00e13

Nous avons donc trouvé le λ maximal pour un nombre donné de fonctions de base. Nous cherchons maintenant un K maximum pour guider nos calculs.

3.4.2 Recherche d'un K maximum

Les figures 3.2 et 3.3 présentent respectivement l'évolution du GCV pour la variable RH_XP, position de la main droite en X, et LA_Zacc, accélération du bras gauche en Z, en fonction du nombre de fonctions de base utilisées et de la pénalisation λ. Une seule courbe apparaît sur les deux figures, alors qu'il y a trois valeurs de pénalisation. Cela s'explique par la proximité des valeurs avec pénalisation et sans pénalisation. Les valeurs sont tellement proches que les courbes sont superposées. Nous avons choisi de représenter les variables RH_XP, la position en X de la main droite, et LA_Zacc, l'accélération en Z du bras gauche, pour avoir une représentation de deux capteurs différents, un à droite et un à gauche, de deux directions

différentes et de deux mesures différentes (la position et l'accélération). Nous avons pris un capteur de la main et un du bras pour avoir les deux extrémités représentées.

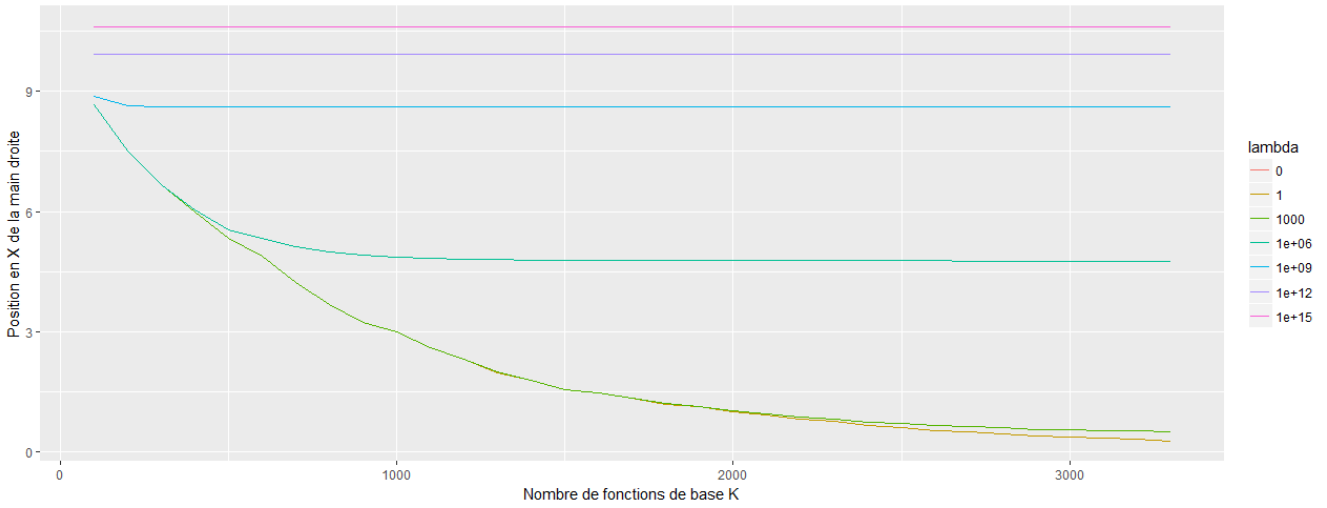


FIGURE 3.2 – GCV de la position en X du capteur de la main droite

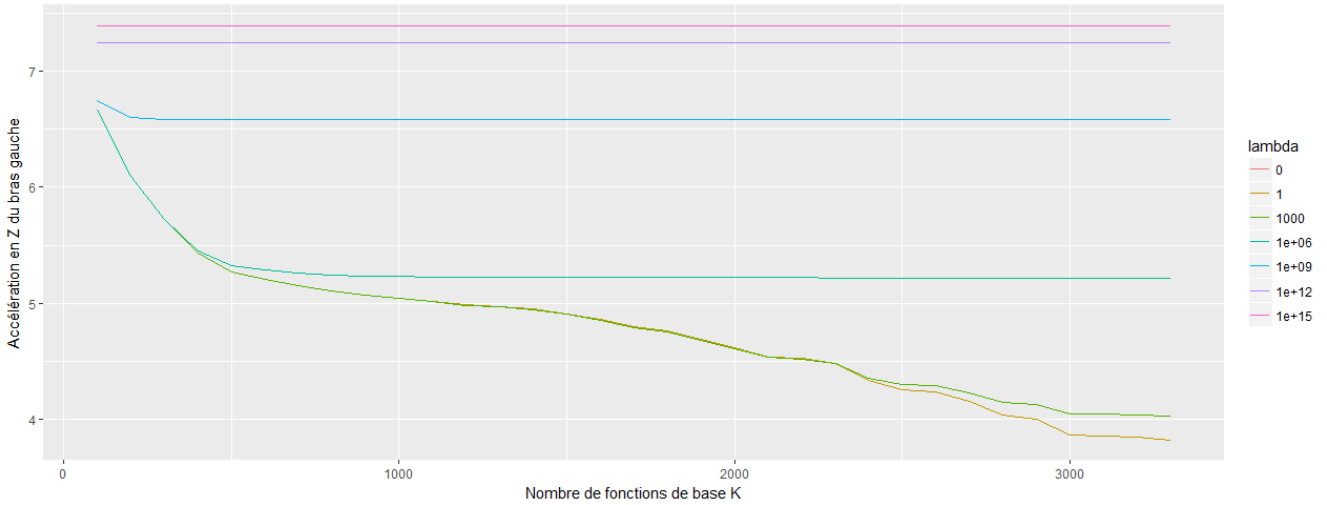


FIGURE 3.3 – GCV de l'accélération en Z du capteur du bras gauche

Nous voyons clairement que plus le λ est élevé, moins le GCV est élevé. Les pénalisations à 10^{12} et 10^{15} ont des valeurs de GCV élevées et presque stagnantes. Nous ne distinguons pas sur les graphiques les courbes qui correspondent à une pénalisation de 0 et 1. Elles sont trop similaires. De plus, le GCV décroît pour les deux variables avec l'augmentation du nombre de fonctions de base. Nous voyons sur les deux graphiques que lorsque le nombre de fonctions de base augmente, la valeur du GCV diminue plus lentement. Le GCV décroît d'environ 1,5 point lorsque l'on passe de 100 à 500 fonctions pour les 4 courbes ayant une pénalisation inférieure ou égale à 10^6. Il faut attendre K=3 300 pour que le GCV diminue à nouveau de 1,5 point avec une pénalisation de 0 ou 1. De plus, la lenteur des calculs augmente considérablement avec le nombre de fonctions de base. Un K trop grand ne semble donc pas approprié au vu de la lenteur des calculs et de l'amélioration marginale du GCV. Ainsi, nous nous sommes restreints à un K maximal de 1 000 bases dans notre recherche des paramètres optimaux de lissage K et λ.

Nous obtenons des résultats similaires avec les autres variables du jeu de données ce qui confirme notre choix de 1 000 pour le K maximal.

3.4.3 Recherche d'un λ optimal

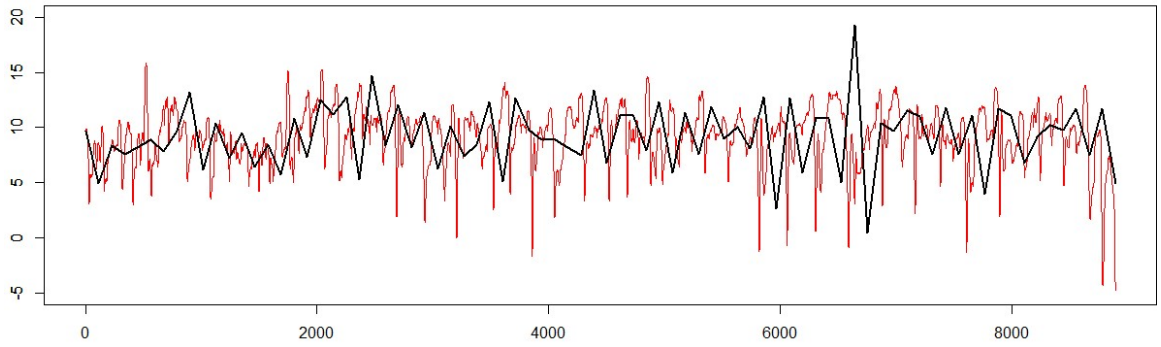
Le tableau 3.2 présente les différents GCV pour les paramètres testés. D'autres tableaux présentant des GCV sont en annexes avec plus de paramètres. Nous avons vu avec les figures 3.2 et 3.3 que les lambdas supérieurs à 10^6 n'ont pas d'intérêt. Nous n'avons donc pas fait de recherche avec ces valeurs. Comme les valeurs de λ entre 0 et 10^3 fournissent des résultats de GCV très intéressants et similaires sur les figures 3.2 et 3.3, nous avons choisi d'explorer des valeurs intermédiaires. Les pénalisations qui minimisent le GCV sont $\lambda = 10^{-3}$ ou $\lambda = 10^3$. Les différences de valeurs de GCV sont minimes. Nous avons décidé d'utiliser $\lambda = 10^{-3}$ comme valeur optimale. En effet, lorsque le GCV est minimisé avec $\lambda = 10^{-3}$, l'écart entre le GCV minimum et les autres valeurs est plus grand. Le minimum se distingue donc mieux avec $\lambda = 10^{-3}$ qu'avec $\lambda = 10^3$.

TABLE 3.2 – GCV des données lissées pour la variable position des capteurs du côté droit

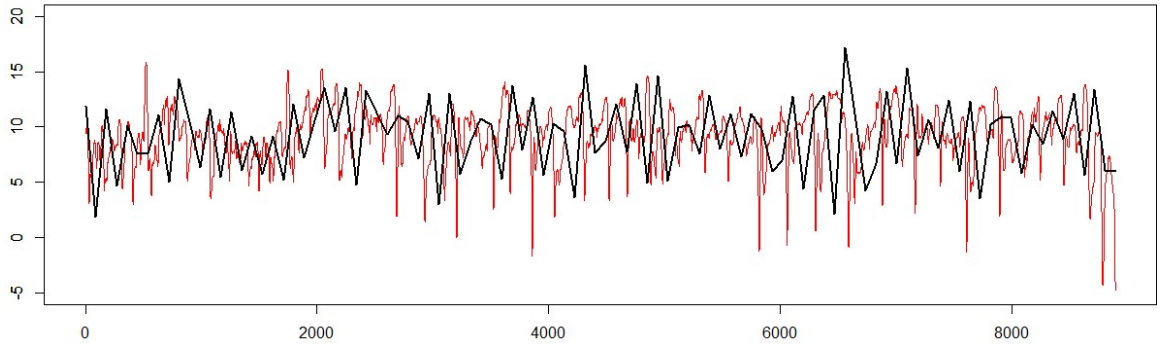
Bases	λ	Variables								
		Avant-bras en X	Avant-bras en Y	Avant-bras en Z	Main en X	Main en Y	Main en Z	Bras en X	Bras en Y	Bras en Z
400	1e+06	2,26	0,80	6,48	6,04	6,74	21,41	8,85	11,68	29,23
400	1e+03	2,22	**0,79**	**6,43**	5,97	6,71	21,25	8,77	11,64	**28,98**
400	1	2,22	0,79	6,43	5,97	6,71	21,25	8,77	11,64	28,98
400	1e-03	**2,14**	0,79	6,43	**5,88**	**6,70**	**21,24**	**8,68**	**11,59**	29,02
400	1e-06	**2,14**	0,79	6,43	5,88	6,70	**21,24**	**8,68**	11,59	29,02
400	0	**2,14**	0,79	6,43	5,88	6,70	**21,24**	**8,68**	11,59	29,02
500	1e+06	2,00	0,76	6,25	5,55	6,55	20,67	8,18	11,35	28,17
500	1e+03	**1,89**	**0,73**	**6,12**	**5,33**	**6,46**	**20,28**	**7,90**	**11,21**	**27,56**
500	1	1,89	0,73	6,12	5,33	6,46	20,28	7,90	11,21	27,56
500	1e-03	1,90	0,74	6,14	5,40	6,51	20,35	8,01	11,30	27,70
500	1e-06	1,90	0,74	6,14	5,40	6,51	20,35	8,01	11,30	27,70
500	0	1,90	0,74	6,14	5,40	6,51	20,35	8,01	11,30	27,70
600	1e+06	1,90	0,73	6,08	5,33	6,46	20,11	7,85	11,20	27,31
600	1e+03	1,68	0,68	5,82	4,87	6,27	19,21	7,18	10,89	25,88
600	1	1,68	0,68	5,82	4,87	6,27	19,21	7,18	10,89	25,88
600	1e-03	**1,64**	**0,67**	**5,78**	**4,79**	**6,19**	**19,07**	**7,06**	**10,79**	**25,69**
600	1e-06	1,64	0,67	5,78	4,79	6,19	19,07	7,06	10,79	25,69
600	0	1,64	0,67	5,78	4,79	6,19	19,07	7,06	10,79	25,69
700	1e+06	1,79	0,72	5,99	5,12	6,30	19,80	7,55	10,94	26,85
700	1e+03	1,35	0,64	5,54	4,24	5,76	18,20	6,28	10,01	24,41
700	1	1,35	0,64	5,54	4,24	5,76	18,20	6,28	10,01	24,41
700	1e-03	**1,34**	**0,63**	**5,50**	**4,11**	**5,69**	**18,05**	**6,08**	**9,95**	**24,20**
700	1e-06	1,34	0,63	5,50	4,11	5,69	18,05	6,08	9,95	24,20
700	0	1,34	0,63	5,50	4,11	5,69	18,05	6,08	9,95	24,20
800	1e+06	1,74	0,70	5,93	4,99	6,18	19,59	7,37	10,74	26,55
800	1e+03	**1,14**	**0,57**	**5,31**	**3,67**	**5,15**	**17,35**	**5,47**	**9,00**	**23,27**
800	1	1,14	0,57	5,31	3,67	5,15	17,35	5,47	9,00	23,27
800	1e-03	1,14	0,57	5,31	3,67	5,15	17,35	5,47	9,00	23,27
800	1e-06	1,14	0,57	5,31	3,67	5,15	17,35	5,47	9,00	23,27
800	0	1,14	0,57	5,31	3,67	5,15	17,35	5,47	9,00	23,27
900	1e+06	1,71	0,69	5,90	4,91	6,10	19,46	7,24	10,59	26,36
900	1e+03	**1,00**	**0,53**	**5,12**	**3,22**	**4,60**	**16,76**	**4,78**	**8,03**	**22,47**
900	1	1,00	0,53	5,12	3,22	4,60	16,76	4,78	8,03	22,48
900	1e-03	1,00	0,53	5,12	3,22	4,60	16,76	4,78	8,03	22,48
900	1e-06	1,00	0,53	5,12	3,22	4,60	16,76	4,78	8,03	22,48
900	0	1,00	0,53	5,12	3,22	4,60	16,76	4,78	8,03	22,48
1 000	1e+06	1,69	0,69	5,88	4,87	6,06	19,39	7,19	10,53	26,28
1 000	1e+03	**0,90**	**0,50**	**4,97**	**2,98**	**4,31**	**16,26**	**4,43**	**7,50**	**21,89**
1 000	1	0,90	0,50	4,97	2,98	4,318	16,26	4,43	7,50	21,89
1 000	1e-03	0,90	0,50	4,97	2,98	4,31	16,26	4,43	7,50	21,89
1 000	1e-06	0,90	0,50	4,97	2,98	4,31	16,26	4,43	7,50	21,89
1 000	0	0,90	0,50	4,97	2,98	4,31	16,26	4,43	7,50	21,89

3.4.4 Recherche d'un K optimal

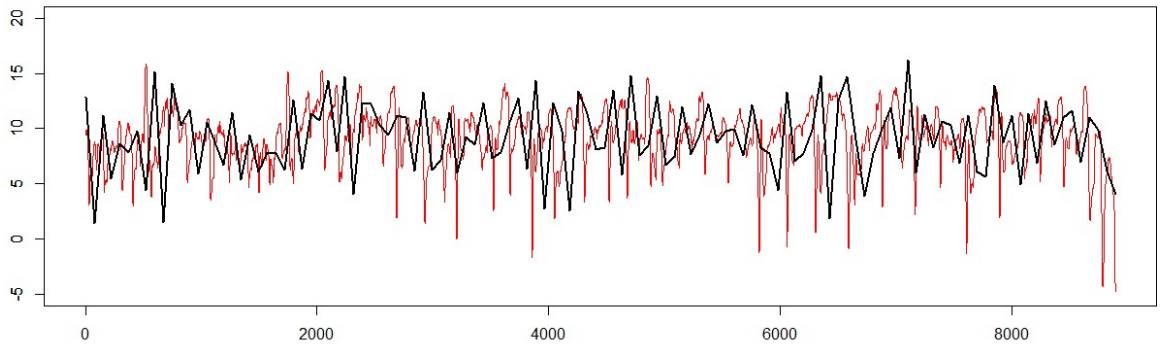
Les figures 3.4a, 3.4b, 3.4c, 3.4d, 3.4e, 3.4f et 3.4g présentées comparant les données originales et les données lissées confirment que plus le nombre de bases augmentent, plus la courbe lissée se rapproche de la courbe initiale. Je n'expose ici que sept graphiques qui sont le résultat de la variable de position de la main droite en X avec un nombre de bases différentes avec une pénalisation de 0,001. De plus, sur les graphiques seulement les premiers 20% des données sont représentées pour avoir une meilleure vision des données. Représenter l'ensemble ne permet pas de distinguer les variations des variables et la qualité du lissage. Dans l'annexe B.2, nous avons placé les mêmes graphiques pour la variable de la vitesse de rotation en Z de la main gauche. Nous observons des résultats similaires.



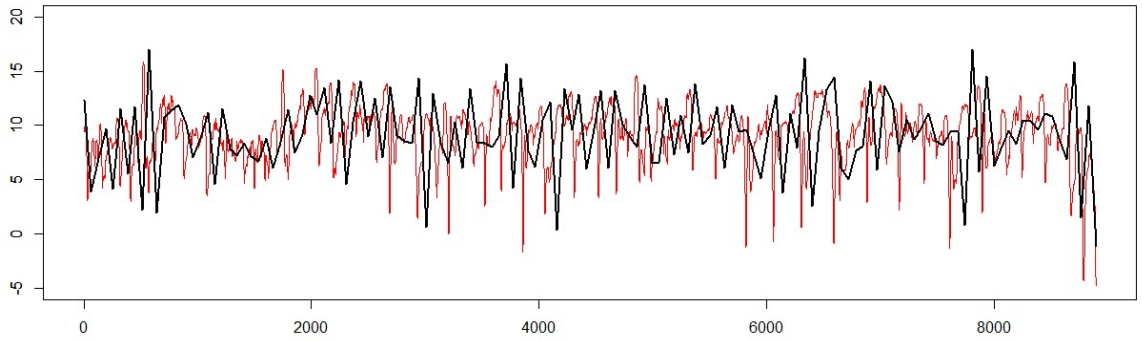
(a) Données originales et lissées de RH_XP K= 400 et lambda = 0.001



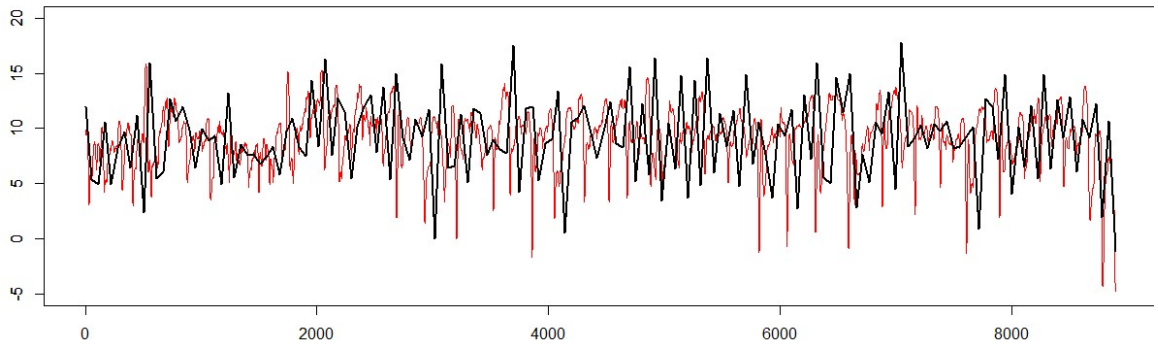
(b) Données originales et lissées de RH_XP K= 500 et lambda = 0.001



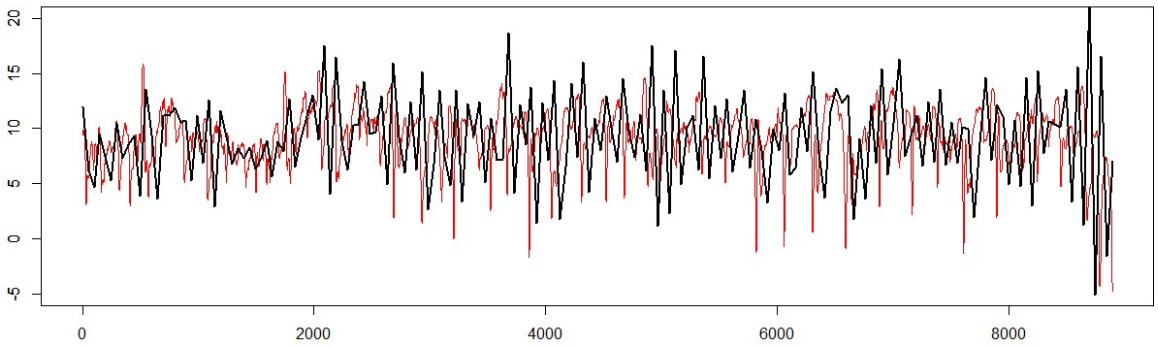
(c) Données originales et lissées de RH_XP K= 600 et lambda = 0.001



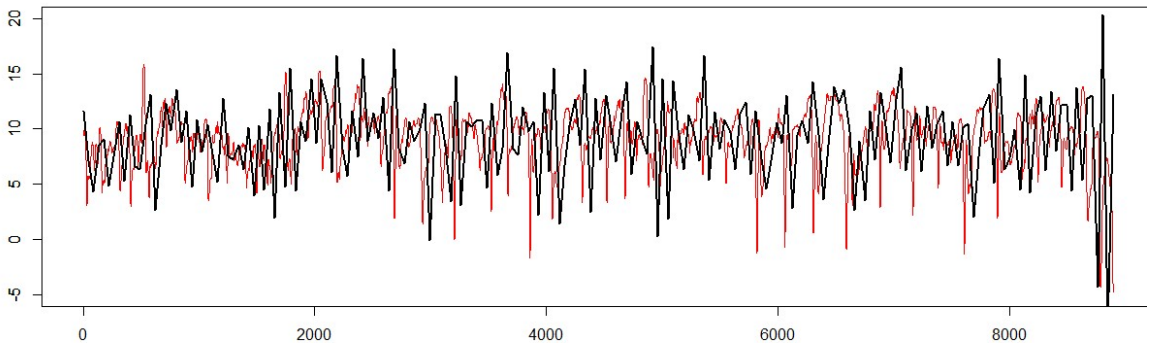
(d) Données originales et lissées de RH_XP K= 700 et lambda = 0.001



(e) Données originales et lissées de RH_XP K= 800 et lambda = 0.001



(f) Données originales et lissées de RH_XP K= 900 et lambda = 0.001



(g) Données originales et lissées de RH_XP K= 1000 et lambda = 0.001

FIGURE 3.4 – Comparaison entre les données originales et lissées de la position de la main droite en X avec différentes valeurs de lissage pour les premiers 20% des données

Nous notons qu'avec un nombre trop faible de bases, nous perdons de la précision. Avec un nombre trop élevé de bases, nous nous rapprochons trop des données originales et récupérons le bruit des données que nous cherchons à éliminer. Graphiquement, il semblerait qu'un nombre de bases en dessous de 500 soit trop faible car nous ne retrouvons pas vraiment la forme des données. Un nombre comme 900 semble trop élevé car le lissage est trop proche des données. Nous pouvons considérer un nombre de 600, 700 ou 800 fonctions de base. Cependant, graphiquement 700 semble idéal. Nous allons donc poursuivre ce travail avec les données lissées grâce à un système de **700 fonctions de base** et **une pénalisation** $\lambda = 10^{-3}$.

3.5 Analyse fonctionnelle par composantes principales

3.5.1 L'analyse par composantes principales pour des données multivariées

Le problème

L'analyse par composantes principales est un concept clé des statistiques multivariées. Le but est de trouver un ensemble de combinaisons linéaires des différentes variables qui capturent la plus grande part de variabilité présente dans les données. Nous définissons dans la suite plus en détail ce concept. Soit X une matrice de données composée de p variables $X_1, ..., X_p$ et de n valeurs telle que :

$$X = nX_p. \tag{3.11}$$

Il est possible de définir une combinaison linéaire f_1 de variables X, aussi appelée composante f_1, par un ensemble de poids w_1 telle que :

$$f_1 = Xw_1.$$

Donc si l'on veut définir p composantes, cela revient à définir une matrice de composantes f telle que :

$$f = Xw,$$

où f est une matrice $n \times p$ notée $[f_1, ..., f_p]$' dans laquelle chaque colonne est une combinaison linéaire des variables X et où w est une matrice $p \times p$ représentant p poids pour p composantes. La première composante f_1 est définie telle que sa variance est maximisée. Nous cherchons donc le vecteur $w_1 = (w_{11}, ..., w_{1p})$' tel que :

$$w_1 = \underset{\|w\|=1}{\operatorname{argmax}} Var(f_1).$$

Avant de continuer le développement de la méthode d'analyse en composantes principales, nous introduisons une nouvelle notion qui nous servira au cours de notre démonstration : les vecteurs propres et valeurs propres.

Vecteurs propres et valeurs propres

Ces explications sont largement tirées de la page web (47). Un vecteur propre est un vecteur dont la direction ne change pas lorsqu'une transformation linéaire lui est appliquée. Les vecteurs propres d'une transformation linéaire V définissent de facon unique cette matrice carrée V. Le vecteur propre w d'une matrice carrée V est le vecteur qui répond à l'équation suivante :

$$Vw = \lambda w, \tag{3.12}$$

où λ est une valeur scalaire appelée *valeur propre*. Ainsi, la transformation linéaire V sur le vecteur w est complétement définie par λ. Nous pouvons réécrire l'équation (3.12) de la façon suivante :

$$\begin{aligned} Vw - \lambda w &= 0, \\ w(V - \lambda I) &= 0, \end{aligned} \tag{3.13}$$

où I est la matrice identité de la même dimension que V. En supposant que w n'est pas le vecteur nul, l'équation (3.13) est définie seulement si la matrice $(V - \lambda I)$ n'est pas inversible. Or $(V - \lambda I)$ est une matrice carrée, donc si une matrice carrée n'est pas inversible, cela signifie que son déterminant est égal à 0. Ainsi, pour trouver les vecteurs propres de V, nous devons résoudre l'équation suivante :

$$Det(V - \lambda I) = 0.$$

Cette équation nous permet de trouver les valeurs propres. Une matrice carrée de taille $p \times p$ aura toujours p valeurs propres qui correspondent chacune à un vecteur propre. Lorsque nous avons trouvé les valeurs propres λ, nous pouvons utiliser l'équation (3.12) pour trouver les vecteurs propres.

La solution

Revenons maintenant à notre vecteur de poids w_1 :

$$\begin{aligned} w_1 &= \underset{\|w\|=1}{\operatorname{argmax}} Var(f_1), \\ &= \underset{\|w\|=1}{\operatorname{argmax}} f_1' f_1, \\ &= \underset{\|w\|=1}{\operatorname{argmax}} (Xw_1)' Xw_1, \qquad (3.14) \\ &= \underset{\|w\|=1}{\operatorname{argmax}} w_1' X' X w_1. \qquad (3.15) \end{aligned}$$

Nous posons $V = X'X$ la matrice de variance/covariance de X. L'équation (3.15) revient donc à maximiser le quotient de Rayleigh tel que :

$$w_1 = \operatorname{argmax} \frac{w_1' V w_1}{w'w},$$

avec w'w= 1. Or, maximiser le quotient de Rayleigh est équivalent à résoudre :

$$Vw_1 = \lambda w_1,$$

c'est-à-dire l'équation de vecteurs propres (3.12). Parmi les différents vecteurs propres qui répondent à (3.12), w_1 est le vecteur propre correspondant à la plus grande valeur propre λ de la matrice V. Nous procédons par itérations pour trouver le k$^{\text{ème}}$ vecteur de poids w_k qui correspond à la k$^{\text{ème}}$ composante f_k. Lors des itérations, nous utilisons la variance résiduelle de la composante f_k, après avoir soustrait la

variabilité des $(k-1)$ premières composantes, pour calculer le vecteur de poids w_k. Mathématiquement, cela signifie qu'au lieu d'utiliser le vecteur de données X de (3.11), nous utilisons :

$$\hat{X}_k = X - \sum_{s=1}^{k-1} X w_s w_s^T.$$

De plus, l'équation (3.14) devient :

$$w_k = \underset{\|w\|=1}{\operatorname{argmax}} (\hat{X}_k w_k)'(\hat{X}_k w_k).$$

Nous posons la matrice de variance/covariance $V = \hat{X}_k' \hat{X}_k$, et nous résolvons l'équation :

$$V w_k = \lambda w_k,$$

où w_k est le vecteur propre correspondant à la plus grande valeur propre λ de la matrice V.

3.5.2 Analyse par composantes principales pour des données fonctionnelles

L'idée dans l'analyse par composantes principales pour les données fonctionnelles est de trouver des fonctions propres $w_1(s), ..., w_k(s)$ et des composantes associées $f_1, ..., f_k$ telles que :

$$f_k = \int X(s) w_k(s) ds,$$

où $X(s)$ peut s'exprimer tel que :

$$X(s) = \sum_{k=1}^{\infty} f_k w_k(s).$$

Avec l'hypothèse qu'un petit nombre de composantes est suffisant, nous pouvons écrire :

$$X(s) \simeq \sum_{k=1}^{n} f_k w_k(s).$$

Nous posons $\lambda_1, \lambda_2, ..., \lambda_p$ p valeurs propres associées à p fonctions propres $w_1, w_2, ..., w_p$. La première fonction propre $w_1(s)$ capture le mode dominant de variation de X :

$$\begin{aligned} w_1(s) &= \underset{\|w\|=1}{\operatorname{argmax}} Var[\int X(s) w_1(s) ds], \\ &= \underset{\|w\|=1}{\operatorname{argmax}} Var[f_1], \end{aligned}$$

où :

$$\|w\| = (\int w(s)^2 ds)^{1/2}.$$

Or d'après le théorème Karhunen-Loève, nous pouvons écrire :

$$w_1(s) = \underset{\|w\|=1}{\operatorname{argmax}} \lambda_1.$$

La k$^{\text{ème}}$ fonction propre w_k est le mode de variation dominant par rapport à $w_1, ..., w_{k-1}$. Ainsi :

$$\begin{aligned} w_k &= \underset{\|w\|=1, <w, w_j>=0, j \in [1,..,k-1]}{\operatorname{argmax}} Var[\int X(s) w_k(s) ds], \\ &= \underset{\|w\|=1, <w, w_j>=0, j \in [1,..,k-1]}{\operatorname{argmax}} Var[f_k], \end{aligned}$$

et d'après le théorème Karhunen-Loève,

$$w_k = \underset{\|w\|=1, <w,w_j>=0, j\in[1,..,k-1]}{\operatorname{argmax}} \lambda_k,$$

où :

$$< w, w_j >= \int w(t)w_j(t)dt.$$

Nous appliquons, dans la prochaine section, la méthode d'analyse en composantes principales fonctionnelles aux données de capteurs présentées dans ce mémoire.

3.6 Résultats des ACP

Nous avons voulu comprendre d'où provenait la variabilité des données de mouvements, quels capteurs et quelles variables expliquaient le plus cette variabilité. Pour cela, nous avons fait des ACP fonctionnelles sur les données en regroupant différentes variables. Nous exposons ici les résultats des ACP effectuées avec les données issues des variables de positions car ce sont ces variables qui nous intéressent le plus. Nous avons cependant fait d'autres ACP fonctionnelles avec les autres variables comme l'accélération et la vitesse de rotation. Pour réaliser les ACP fonctionnelles présentées ci-dessous, nous avons utilisé les variables de position des 6 capteurs dans les 3 directions X, Y et Z. Cela représente donc 18 variables. Cependant, nous n'avons pas utilisé les 18 variables en même temps pour effectuer les ACP. Nous nous sommes concentrés sur les variables de positions dans une direction précise ou les variables d'un bras plutôt qu'un autre. Nous avons d'abord réalisé des ACP fonctionnelles en utilisant les données de position des 6 capteurs dans une direction précise. Les ACP et les visualisations ont été réalisées avec la bibliothèque funclustering du logiciel R.

En ce qui concerne la direction X, 98,4% de la variabilité est expliquée avec 2 facteurs. Le premier facteur explique 66,3% de la variabilité. Le deuxième facteur explique 32,1% de la variabilité.

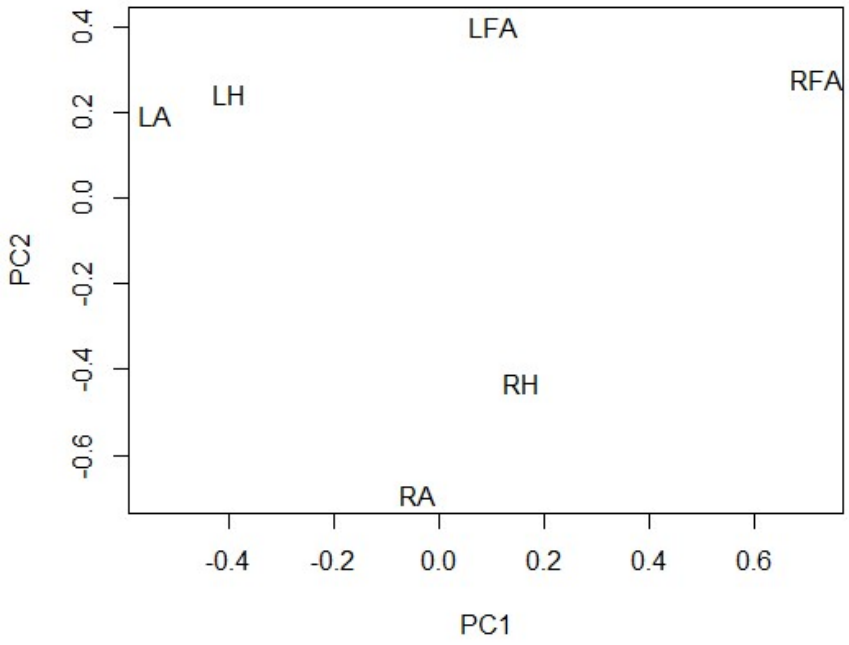


FIGURE 3.5 – ACP des 6 capteurs des positions en X

Nous voyons sur la figure 3.5 que les 2 facteurs discriminent les capteurs gauches et droits en fonction de leur position (avant-bras, main, bras). En ce qui concerne la direction Y : 99,9% de la variabilité est expliquée avec 2 facteurs. Le premier facteur explique 98,8%

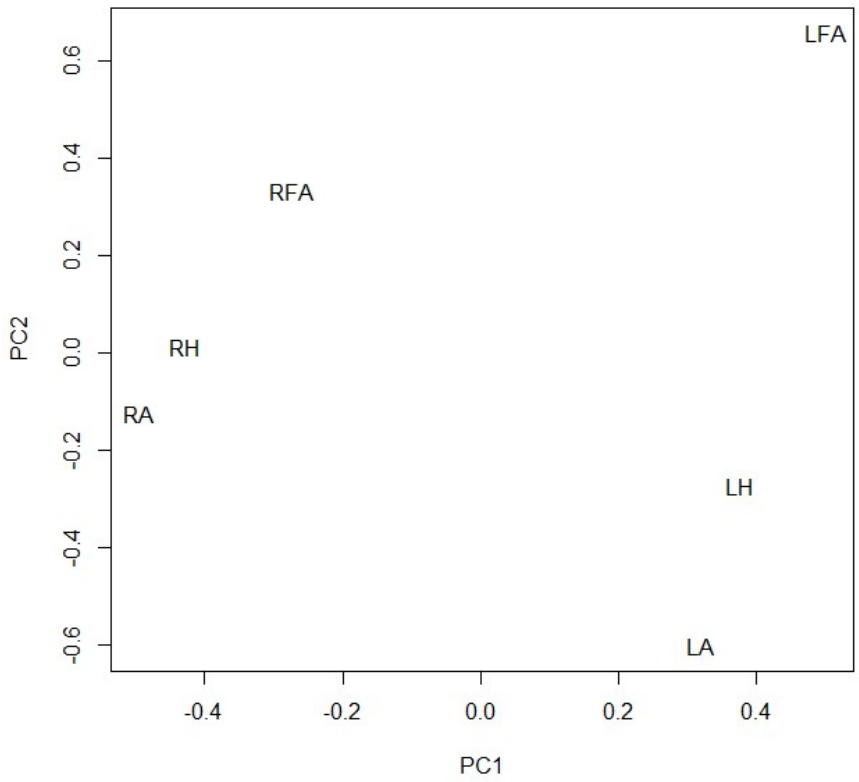


FIGURE 3.6 – ACP des 6 capteurs des positions en Y

Nous voyons sur la figure 3.6 que le facteur 1 discrimine les capteurs gauches et droits alors que le facteur 2 discrimine les capteurs en fonction de leur position (avant-bras, main, bras). En ce qui concerne la direction Z : 99,6% de la variabilité est expliquée avec 2 facteurs. Le premier facteur explique 94% de la variabilité.

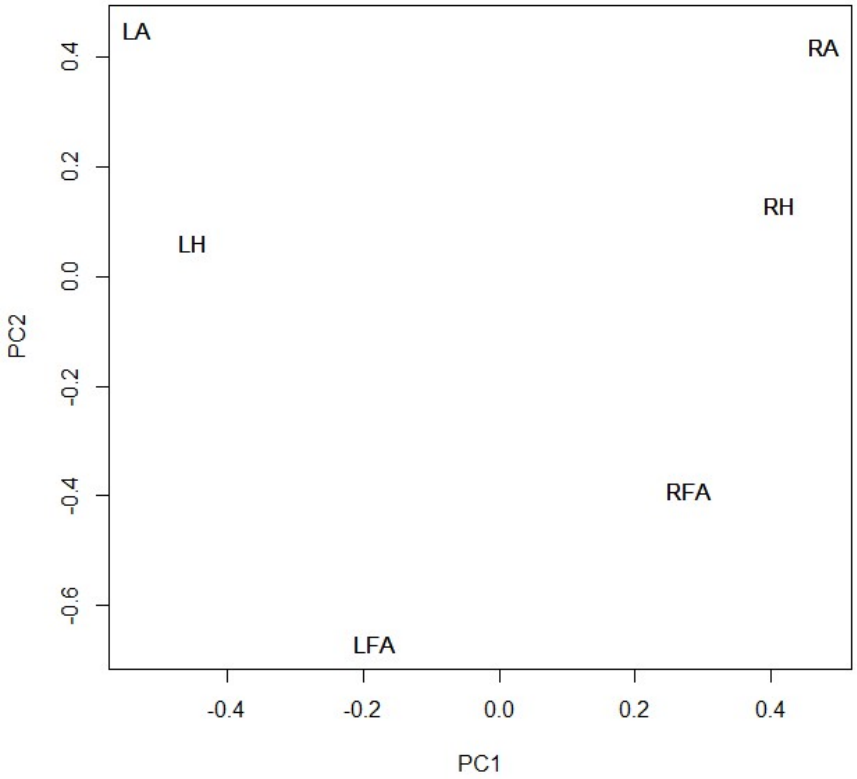


FIGURE 3.7 – ACP des 6 capteurs des positions en Z

Nous voyons sur la figure 3.7 que le facteur 1 discrimine les capteurs gauches et droits alors que le facteur 2 discrimine les capteurs en fonction de leur position (bras, main, avant-bras).

Nous avons ensuite réalisé deux autres ACP fonctionnelles : une pour les 9 variables qui concernent le bras droit et une avec les 9 variables qui concernent le bras gauche. En ce qui concerne le côté droit : 99,8% de la variabilité est expliquée avec 2 facteurs. Le premier facteur explique 98,7% de la variabilité.

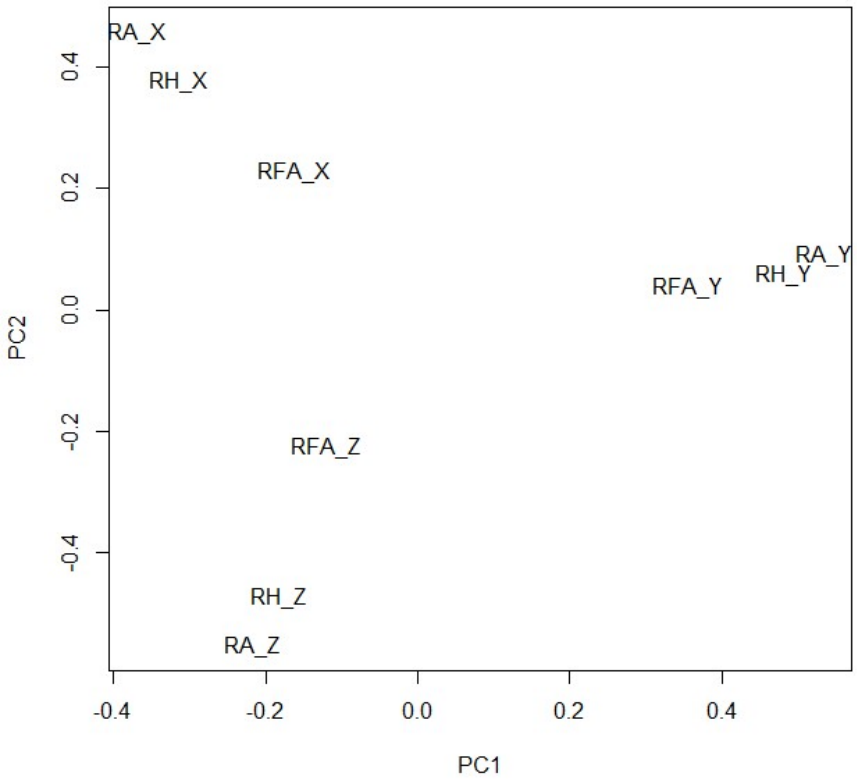


FIGURE 3.8 – ACP des 9 capteurs des positions à droite

Nous voyons sur la figure 3.8 que les capteurs sont regroupés en fonction de leur direction. De plus, le facteur 1 discrimine les directions X , Z avec Y. Le facteur 2 discrimine les directions X et Z. En ce qui concerne le côté gauche : 98,3% de la variabilité est expliquée avec 2 facteurs. Le premier facteur explique 75,7% de la variabilité. Le deuxième facteur explique 22,6% de la variabilité.

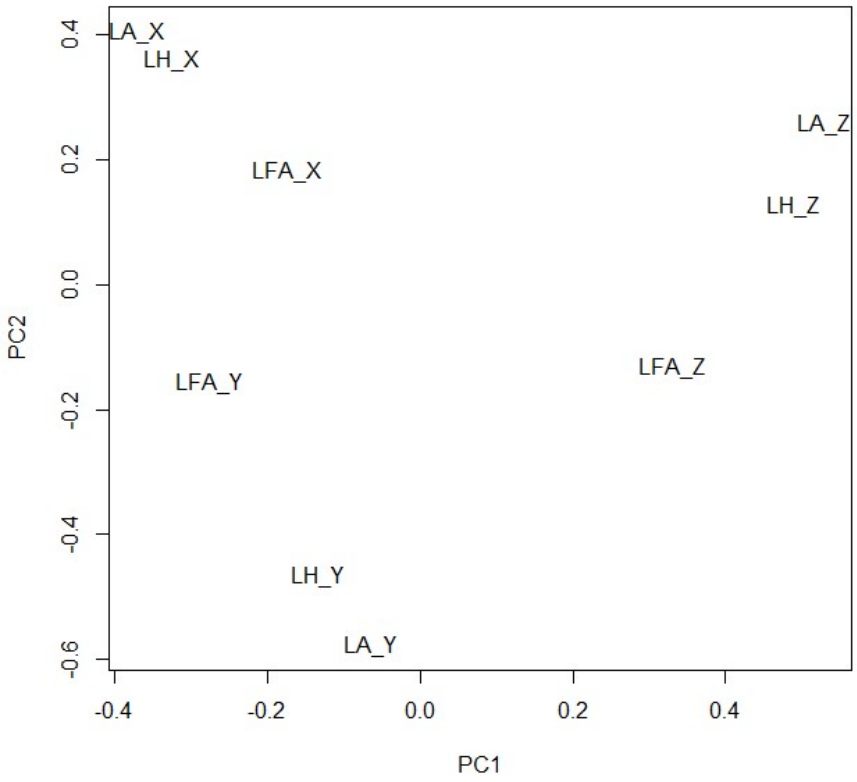


FIGURE 3.9 – ACP des 9 capteurs des positions à gauche

Nous voyons que les capteurs sont discriminés en fonction de leur direction sur la figure 3.9.

Ces analyses nous montrent que les capteurs sont généralement discriminés par le côté gauche ou droite. Ils sont ensuite discriminés par leur direction X, Y ou Z.

Nous venons d'analyser la variabilité des mouvements grâce à des ACP fonctionnelles. Les résultats obtenus correspondent aux résultats graphiques trouvés dans le chapitre précédent. Les mouvements droite/gauche sont très distincts. Le côté des capteurs explique bien la variabilité des mouvements. Cela corrobore les corrélations du chapitre 2 où nous avions vu que les capteurs droite/gauche sont peu corrélés. Nous changeons de technique statistique dans le prochain chapitre pour passer de l'inférence à la prédiction avec des réseaux de neurones.

Chapitre 4

Prédiction et réseaux de neurones

4.1 Objectifs de la prédiction

Dans ce chapitre, nous ne cherchons plus à expliquer la variabilité des mouvements mais à prédire les types de mouvements. En effet, dans la première partie, nous avons expliqué que les mouvements des chefs d'orchestre étaient des mouvements très précis et choisis. Nous construirons ici un algorithme capable de reconnaître le type de mouvements du chef en fonction des données de position des capteurs que nous lui fournissons.

4.2 Réseaux de neurones perceptrons multicouches

La chaîne Youtube 3Blue1Brown avec les vidéos (1), (2) et (3) nous a particulièrement aidés à construire cette partie. (32) et (33) expliquent également de façon claire et structurée comment calculer l'algorithme de rétropropagation.

4.2.1 Structure d'un réseau de neurones perceptrons multicouches

Nous rappellerons tout d'abord ce qu'est un réseau de neurones classique avant de rentrer dans le détail de réseaux plus complexes. Nous prendrons le cas du réseau de neurones à perceptrons multicouches qui est le réseau de neurones le plus basique. Tout en expliquant l'algorithme simple de l'apprentissage du réseau de neurones avec la rétropropagation, nous expliquerons l'apprentissage par mini-lots qui simplifie et facilite l'apprentissage du réseau. Nous baserons notre explication sur le schéma de la figure 4.1 :

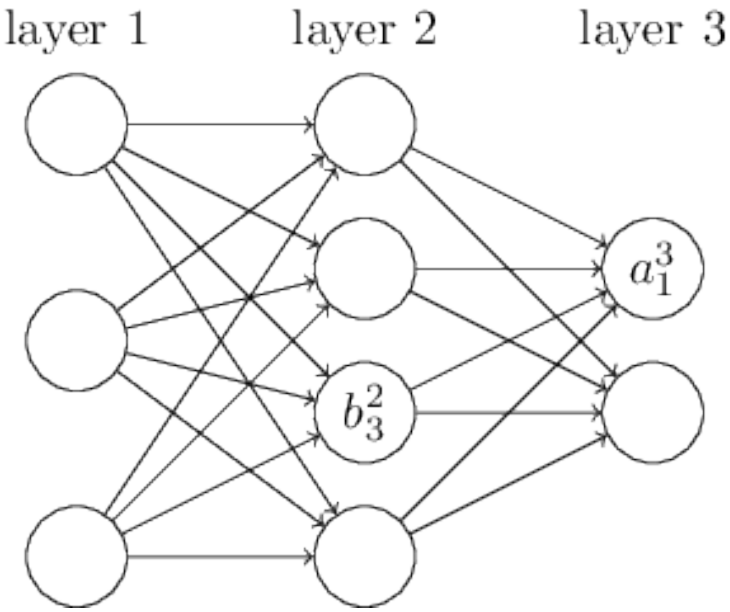


FIGURE 4.1 – Schéma d'un réseau de neurones avec 3 couches : 1 couche d'entrée, 1 couche cachée et 1 couche de sortie du chapitre 2 de (37)

Un réseau de neurones est composé de différents éléments. Tout d'abord, nous avons p variables explicatives, $x_1, ..., x_p$ à partir desquelles nous voulons prédire une variable cible Y (5), (37). Chaque variable explicative joue un rôle différent dans la prédiction et n'aura donc pas la même importance. Le but du réseau est donc de déterminer les paramètres qui définissent l'importance de chaque variable dans la prédiction. Pour passer des variables explicatives à la prédiction et trouver ces paramètres, le réseau est composé de plusieurs couches. Sur la figure 4.1, le réseau est composé de trois couches (layer sur la figure). La première couche est la couche de d'entrée qui contient toutes les variables explicatives. Dans la figure 4.1, la première couche contient les trois variables du modèle chacune représentée par un cercle. La dernière couche est la couche de sortie qui contient autant d'éléments que de valeurs possibles de Y. Sur la figure 4.1, Y est binaire, donc il n'y a que deux cercles. Au milieu, il y a les couches intermédaires, appelées couches cachées. Elles contiennent un nombre d'éléments variable. Ici il y a une couche cachée composée de quatre éléments. Les couches sont notées $a^{(1)}, ..., a^{(L)}$ où L est le nombre de couches. Dans notre exemple, L=3. La couche $a^{(1)}$ correspond à la couche d'entrée, elle contient les variables explicatives. Ainsi dans notre exemple :

$$a^{(1)} = (a_1^{(1)}, a_2^{(1)}, a_3^{(1)}) = (x_1, x_2, x_3).$$

Chaque élément des couches $a^{(1)}, ..., a^{(L)}$ s'appelle un neurone. Il n'y a pas nécessairement le même nombre de neurones dans chaque couche. Les neurones des couches successives sont reliés entre eux. Chaque neurone d'une couche l est une fonction d'une combinaison linéaire de tous les neurones de la couche $l-1$ plus un biais telle que :

$$a_j^{(l)} = f[\sum_{i \in a^{(l-1)}} w_{ji}^{(l)} a_i^{(l-1)} + b_j^{(l)}],$$

où :

- $a_j^{(l)}$ représente le j^{eme} neurone de la couche l,
- $w_{ji}^{(l)}$ représente le poids du neurone i de la couche $l-1$ dans le neurone j de la couche l,
- $b_j^{(l)}$ représente le biais du neurone j de la couche l.

La fonction f que nous utilisons dépend des caractéristiques de la variable Y que nous voulons prédire. Elle s'appelle fonction d'activation. Dans notre exemple, nous utiliserons la fonction tangente hyperbolique, *tanh* comme fonction d'activation car c'est une des plus utilisées. Les poids w et les biais b sont appelés les paramètres du modèle. Il est à noter que la fonction d'activation change pour calculer la dernière couche. Nous avons alors :

$$a_j^{(L)} = g[\sum_{i \in a^{(L-1)}} w_{ji}^{(L)} a_i^{(L-1)} + b_j^{(L)}],$$

où L représente la couche de sortie. Elle dépend directement du type de variable que nous cherchons à prédire. Ici nous prédisons une variable binaire. La fonction g sera donc la fonction sigmoïde. Illustrons nos propos avec les données de la figure 4.1. Nous avons les variables explicatives :

$$\begin{aligned} a_1^{(1)} &= x_1, \\ a_2^{(1)} &= x_2, \\ a_3^{(1)} &= x_3. \end{aligned}$$

Nous pouvons donc calculer les neurones de la couche cachée $a^{(2)}$:

$$\begin{aligned} a_1^{(2)} &= f[w_{11}^{(2)} a_1^{(1)} + w_{12}^{(2)} a_2^{(1)} + w_{13}^{(2)} a_3^{(1)} + b_1^{(2)}], \\ a_2^{(2)} &= f[w_{21}^{(2)} a_1^{(1)} + w_{22}^{(2)} a_2^{(1)} + w_{23}^{(2)} a_3^{(1)} + b_2^{(2)}], \\ a_3^{(2)} &= f[w_{31}^{(2)} a_1^{(1)} + w_{32}^{(2)} a_2^{(1)} + w_{33}^{(2)} a_3^{(1)} + b_3^{(2)}], \\ a_4^{(2)} &= f[w_{41}^{(2)} a_1^{(1)} + w_{42}^{(2)} a_2^{(1)} + w_{43}^{(2)} a_3^{(1)} + b_4^{(2)}], \end{aligned} \tag{4.1}$$

avec :

$$f(a) = \frac{exp(2a) - 1}{exp(2a) + 1}.$$

Nous pouvons ensuite calculer les neurones de la couche de sortie $a^{(3)}$:

$$\begin{aligned} a_1^{(3)} &= g[w_{11}^{(3)} a_1^{(2)} + w_{12}^{(3)} a_2^{(2)} + w_{13}^{(3)} a_3^{(2)} + w_{14}^{(3)} a_4^{(2)} + b_1^{(3)}], \\ a_2^{(3)} &= g[w_{21}^{(3)} a_1^{(2)} + w_{22}^{(3)} a_2^{(2)} + w_{23}^{(3)} a_3^{(2)} + w_{24}^{(3)} a_4^{(2)} + b_2^{(3)}], \end{aligned} \tag{4.2}$$

avec :

$$\sigma(a) = \frac{1}{1 + e^{-a}}.$$

4.2.2 Apprentissage du réseau de neurones

Dans la phase d'apprentissage, l'objectif du réseau est donc de trouver les paramètres b et w qui maximisent le taux de bonne prédiction c'est-à-dire qui permettent de minimiser l'erreur de prédiction. Pour mesurer cette erreur, nous utilisons une fonction de coût. L'algorithme le plus répandu pour minimiser la fonction de coût est le calcul du gradient avec la rétro-propagation (*backpropagation*). Nous expliquerons étape par étape comment minimiser cette fonction et comment trouver les paramètres optimum du réseau pour obtenir les meilleures prédictions possibles.

Etape 0 : division du jeu de données en mini-lots

Nous prenons comme hypothèse que nous avons 1 000 observations d'apprentissage. Nous divisons notre jeu de 1 000 observations en 10 lots (par exemple) de 100 observations. Nous notons M_k le lot k du jeu de données où $k = 1, .., 10$. La répartition des observations dans chaque lot se fait au hasard et sans remise. Entre l'étape 1 et l'étape 3, nous utiliserons seulement le lot 1, M_1.

Etape 1 : initialisation et 1ère prédiction - Propagation vers l'avant - Forward propagation FP

La première étape consiste à initialiser tous les paramètres $w_{ji}^{(l)}$ et $b_j^{(l)}$ avec des valeurs aléatoires et calculer l'erreur de prédiction ce qui permet d'en juger la qualité pour chaque donnée d'apprentissage. Par exemple, nous définissons aléatoirement une valeur commune pour les poids et les biais de 0,25 comme décrit dans (21). Ainsi, d'après (4.1), nous pouvons calculer les valeurs des neurones de la couche cachée :

$$\begin{aligned} a_1^{(2)} &= f(0,25x_1 + 0,25x_2 + 0,25x_3 + 0,25), \\ a_2^{(2)} &= f(0,25x_1 + 0,25x_2 + 0,25x_3 + 0,25), \\ a_3^{(2)} &= f(0,25x_1 + 0,25x_2 + 0,25x_3 + 0,25), \\ a_4^{(2)} &= f(0,25x_1 + 0,25x_2 + 0,25x_3 + 0,25). \end{aligned}$$

D'après (4.2), nous pouvons calculer les valeurs des neurones de la couche de sortie :

$$\begin{aligned} a_1^{(3)} &= g(0,25a_1^{(2)} + 0,25a_2^{(2)} + 0,25a_3^{(2)} + 0,25a_4^{(2)} + 0,25), \\ a_2^{(3)} &= g(0,25a_1^{(2)} + 0,25a_2^{(2)} + 0,25a_3^{(2)} + 0,25a_4^{(2)} + 0,25), \end{aligned}$$

où $a_1^{(3)}$ et $a_2^{(3)}$ représentent les probabilités conditionnelles P(Y=0 | $a^{(3)}$) et P(Y=1|$a^{(3)}$) respectivement (15). Ensuite nous calculons l'erreur de prédiction. La fonction utilisée pour calculer cette erreur est appelée fonction de perte, notée $\boldsymbol{L_m}$, où m représente la m$^{\text{ème}}$ donnée d'apprentissage. Nous définissons Y, le vecteur cible, tel que Y=$(y_1, ..., y_m)$. Ainsi :

$$L_m = (y_m - a_m^{(3)})^2$$

Nous effectuons ces opérations pour toutes les données d'apprentissage m. Nous calculons ensuite la fonction de coût (16). Dans notre exemple, la fonction de coût est le SSE (5) :

$$SSE = \sum_{m=1}^{100} L_m = \sum_{m=1}^{100} (y_m - a_m^{(3)})^2 \tag{4.3}$$

La différence $(y - a^{(3)})^2$ sera grande lorsque la prédiction sera éloignée de la vérité. La différence sera petite lorsque la prédiction sera proche de la vérité. Ainsi, le SSE est bien minimisé lorsque les prédictions sont de plus en plus exactes (16).

Etape 2 : Calcul du gradient

Nous cherchons maintenant à minimiser la fonction de coût, ici le SSE, c'est-à-dire trouver les paramètres $w_{ji}^{(l)}$ et $b_j^{(l)}$ qui la minimisent. Il existe un outil, appelé *gradient*, qui permet de mesurer le changement de valeur de la fonction de coût lorsque ces paramètres sont modifiés (9). Le gradient mesure le changement de valeur du SSE lorsque les paramètres du réseau sont modifiés par rapport à leur valeur initialisée aléatoirement lors de l'étape 1. Dans le cas d'un réseau de neurones, il n'est mathématiquement pas juste de représenter la fonction du SSE en deux ou trois dimensions. Il y a autant de dimensions que de paramètres, c'est-à-dire 26 dimensions pour notre exemple (20 poids et 6 biais). Mais si nous simplifions et imaginons le SSE, en trois dimensions, représenté par un espace avec des collines et des vallées, alors nous pouvons mieux comprendre le rôle du gradient. Avec notre valeur actuelle du SSE trouvée à l'étape 1, nous nous situons sur un point de cet espace. Le gradient donne la direction de la montée la plus raide ou encore dans quelle direction aller pour augmenter le SSE le plus rapidement. Ainsi, la valeur opposée du gradient donne la direction de la pente la plus raide c'est-à-dire la direction qui permet de diminuer le SSE le plus

rapidement. Cela nous permet de déplacer les valeurs des paramètres pour faire diminuer la valeur du SSE. Nous quantifierons, pour chaque paramètre du réseau, l'effet d'un changement de valeur sur le SSE. Dans un premier temps, nous calculerons l'effet du changement de valeurs des poids puis nous quantifierons l'effet des changements de valeurs des biais sur le SSE. Pour quantifier l'effet du changement de valeur des poids, prenons l'exemple du poids $w_{11}^{(3)}$. Cet effet est représenté par la dérivée partielle du SSE en $w_{11}^{(3)}$ ou encore le gradient du SSE en $w_{11}^{(3)}$ exprimé par $\frac{\partial SSE}{\partial w_{11}^{(3)}}$. Le calcul de $\frac{\partial SSE}{\partial w_{11}^{(3)}}$ peut se faire de façon analytique car il s'agit au final simplement d'une fonction composée de combinaisons linéaires des variables originales X. La dérivée d'une telle fonction peut se calculer facilement pour chaque observation du jeu de données et la moyenne sur les 100 observations peut alors être calculée. Nous réalisons ces opérations pour les 26 paramètres $w_{ji}^{(l)}$ et $b_j^{(l)}$ du réseau, ce qui nous permet d'obtenir le vecteur du gradient $\nabla(SSE)$:

$$\nabla(SSE) = \mathbf{grad}(SSE) = \begin{bmatrix} \frac{\partial SSE}{\partial w_{11}^{(2)}} \\ \vdots \\ \frac{\partial SSE}{\partial w_{24}^{(3)}} \\ \frac{\partial SSE}{\partial b_1^{(2)}} \\ \vdots \\ \frac{\partial SSE}{\partial b_2^{(3)}} \end{bmatrix} \tag{4.4}$$

Le signe de $\nabla(SSE)$ nous indique dans quel sens doit évoluer le paramètre du réseau alors que la valeur absolue du nombre nous indique l'intensité du changement. Les dérivées partielles nous donnent le gradient de la fonction de coût. Cependant, nous sommes réellement intéressés par la valeur opposée au gradient $-\nabla(SSE)$.

Etape 3 : ajuster les poids

Lors de l'étape 2, nous avons vu comment obtenir le gradient $\nabla(SSE)$ de la fonction de coût. Maintenant nous allons ajuster les valeurs des paramètres avec les valeurs du gradient. Lors de cette étape, un hyper-paramètre fait son entrée : le taux d'apprentissage η. Multiplier les paramètres par η amène donc un changement dans le SSE de :

$$-\eta\nabla(SSE).$$

Si nous posons η=0,01 par exemple, alors nous trouvons les nouveaux paramètres grâce à l'opération suivante d'après (4.4) :

$$\begin{bmatrix} N(w_{11}^{(2)}) \\ \vdots \\ N(w_{24}^{(3)}) \\ N(b_1^{(2)}) \\ \vdots \\ N(b_2^{(3)}) \end{bmatrix} = \begin{bmatrix} (w_{11}^{(2)}) \\ \vdots \\ (w_{24}^{(3)}) \\ (b_1^{(2)}) \\ \vdots \\ (b_2^{(3)}) \end{bmatrix} - 0,01 \begin{bmatrix} \frac{\partial SSE}{\partial w_{11}^{(2)}} \\ \vdots \\ \frac{\partial SSE}{\partial w_{24}^{(3)}} \\ \frac{\partial SSE}{\partial b_1^{(2)}} \\ \vdots \\ \frac{\partial SSE}{\partial b_2^{(3)}} \end{bmatrix} = \begin{bmatrix} 0,25 \\ 0,25 \\ \vdots \\ 0,25 \\ 0,25 \end{bmatrix} - 0.01 \begin{bmatrix} \frac{\partial SSE}{\partial w_{11}^{(2)}} \\ \vdots \\ \frac{\partial SSE}{\partial w_{24}^{(3)}} \\ \frac{\partial SSE}{\partial b_1^{(2)}} \\ \vdots \\ \frac{\partial SSE}{\partial b_2^{(3)}} \end{bmatrix}$$

Nous notons N_1 le vecteur des nouveaux paramètres réalisé après la 1ère rétropropagation (9).

Etape 4 : nouvelle prédiction et recalcul de la fonction de coût pour un nouveau lot

Dans l'étape 3, nous avons calculé un nouveau vecteur de paramètres N_1, à partir des paramètres initiaux et des données du lot 1. Nous effectuons maintenant de nouvelles prédictions avec ces paramètres et de nouvelles données : les données du lot 2. La procédure est la même qu'à l'étape 1 d'initialisation sauf qu'au lieu d'être égaux à 0,25, les poids et les biais sont contenus dans le vecteur N_1 de l'étape 3. Après avoir calculé les nouvelles prédictions $a_1^{(3)}$ et $a_2^{(3)}$ à partir des données du lot 2, nous pouvons calculer la nouvelle valeur de la fonction de coût comme dans l'équation (4.3) de l'étape 1.

Etape 5 : recommencer les étapes 2, 3 et 4

Refaire les étapes 2, 3 et 4 en boucle jusqu'à ce que tous les lots soient passés une fois, c'est-à-dire jusqu'à ce que le modèle ait vu toutes les observations une fois. Puisqu'il y a 10 lots, nous réalisons 10 rétropropagations. L'étape 5 sera terminée lorsque le vecteur N_{10} sera calculé.

Etape 6 : recommencer les étapes 1 à 5 pour les e époques

A l'étape 5, le réseau a vu les 1 000 observations une fois. Cependant, cela ne suffit pas pour trouver les paramètres qui minimisent la fonction de coût. Nous souhaitons ici que le modèle voie l'ensemble des données un nombre e de fois. Après que le modèle ait vu la totalité des données une fois, nous disons qu'une *époque* est terminée (10). Dans notre exemple, nous considérons 5 époques. Lorsque l'époque 1 est terminée, les 1000 données sont remélangées et 10 nouveaux lots sont recréés au hasard et sans remise. Nous pouvons alors repartir à l'étape 1 de prédiction avec comme paramètres de départ, les paramètres du vecteur N_{10} calculés avec les données du dernier lot M_{10}. Le processus s'arrête lorsque le modèle a vu les données e fois, 5 fois dans notre exemple. Les paramètres du vecteur N_{10} de l'époque 5 seront les paramètres retenus pour le calcul des prédictions.

4.3 Réseaux de neurones récurrents RNN

Nous venons d'expliquer le fonctionnement d'un réseau de neurones perceptrons multicouches. Nous verrons maintenant le fonctionnement d'un réseau de neurones plus complexes qui permet de gérer des données temporelles : le réseau de neurones récurrents, noté RNN (35).

4.3.1 Architecture d'un RNN

Dans ce qui suit, nous nous aiderons de la figure 4.2 pour appuyer nos explications :

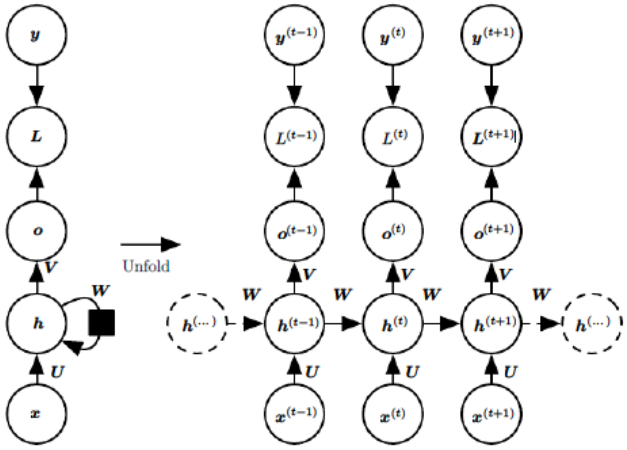


FIGURE 4.2 – Architecture d'un réseau de neurones récurrents, figure 10.3 de (22)

Supposons que nous avons une séquence de données $x = (x^{(1)}, ..., x^{(\tau_x)})$ où l'indice correspond à un pas de temps *t*, aussi appelé *timestep*, où t = 1, ..., τ_x. Par exemple, si nous cherchons à prédire Y à partir de p variables X alors $x^{(t)} = (x_1^{(t)}, ..., x_p^{(t)})$ et $x^{(t)}$ représente donc le vecteur de données au temps t $\in \mathbb{R}^p$. Nous avons également une séquence de données $y = (y^{(1)}, ..., y^{(\tau_y)})$. Similairement au réseau de neurones expliqué dans la section 4.2, le vecteur de données au temps *t*, $x^{(t)}$, est relié à une seule couche cachée $h^{(t)}$ $\in \mathbb{R}^m$ composée de m neurones par une fonction d'activation *f*. De plus, la couche cachée $h^{(t)}$, grâce à une autre fonction d'activation g, permet de faire une prédiction $o^{(t)} \in \mathbb{R}^C$ sur C classes de la variable Y et ainsi, connaissant la vraie valeur de la variable cible $y^{(t)}$, le modèle peut calculer la valeur de la fonction de coût $L^{(t)}$. Cependant, dans un RNN contrairement à un réseau de neurones classique, la couche cachée, ou encore appelée état caché, au pas de temps t dépend de l'activation de l'état caché du pas de temps *t-1*. Ainsi la couche $h^{(t)}$ prend en compte la couche $h^{(t-1)}$ qui a elle-même pris en compte $h^{(t-2)}$... Cela permet au modèle d'avoir une mémoire temporelle d'assez courte durée (12). Il existe différentes structures de RNN dépendemment de la longueur des séquences x et y correspondantes. Ainsi les différentes structures sont :

— MANY to MANY où $\tau_x > 1$, $\tau_y > 1$ et $\tau_x = \tau_y$,
— MANY to MANY où $\tau_x > 1$, $\tau_y > 1$ et $\tau_x \neq \tau_y$,
— MANY to ONE où $\tau_x > 1$, $\tau_y = 1$,
— ONE to MANY où $\tau_x = 1$, $\tau_y > 1$.

La structure présentée dans la figure 4.2 est une structure MANY-TO-MANY avec séquences égales car pour chaque $x^{(t)}$ de la séquence de données il y a un $y^{(t)}$ associé. La longueur des séquences x et y est égale. Une des applications de ce réseau est la prédiction du prix d'une action d'un jour à l'autre. Dans ce cas, la valeur de $y^{(t-1)}$ sert également comme valeur pour $x^{(t)}$. Pour chaque $x^{(t)}$, le modèle donne une prédiction $o^{(t)}$ du prix de l'action le lendemain (46). Nous pouvons retrouver le schéma MANY to MANY avec des longueurs de séquences différentes dans un contexte de traduction de texte d'une langue vers une autre. Le nombre de mots d'une phrase dans une langue, c'est-à-dire la longueur d'une séquence x, ne sera pas nécessairement le même dans une autre langue, c'est-à-dire la longueur de la séquence attendue y (14), (29). Le schéma MANY to ONE sera le schéma utilisé dans ce travail. Pour chaque séquence de données x, il y a un unique y. Ce schéma est utilisé dans des travaux de classification. Dans ce mémoire, nous avons de multiples séquences et à chaque séquence est associée deux types de mouvements comme nous pouvons le voir sur la figure 4.3. Le but est que le modèle réussisse à classer chaque séquence dans l'une ou autre des catégories correctement (14), (29).

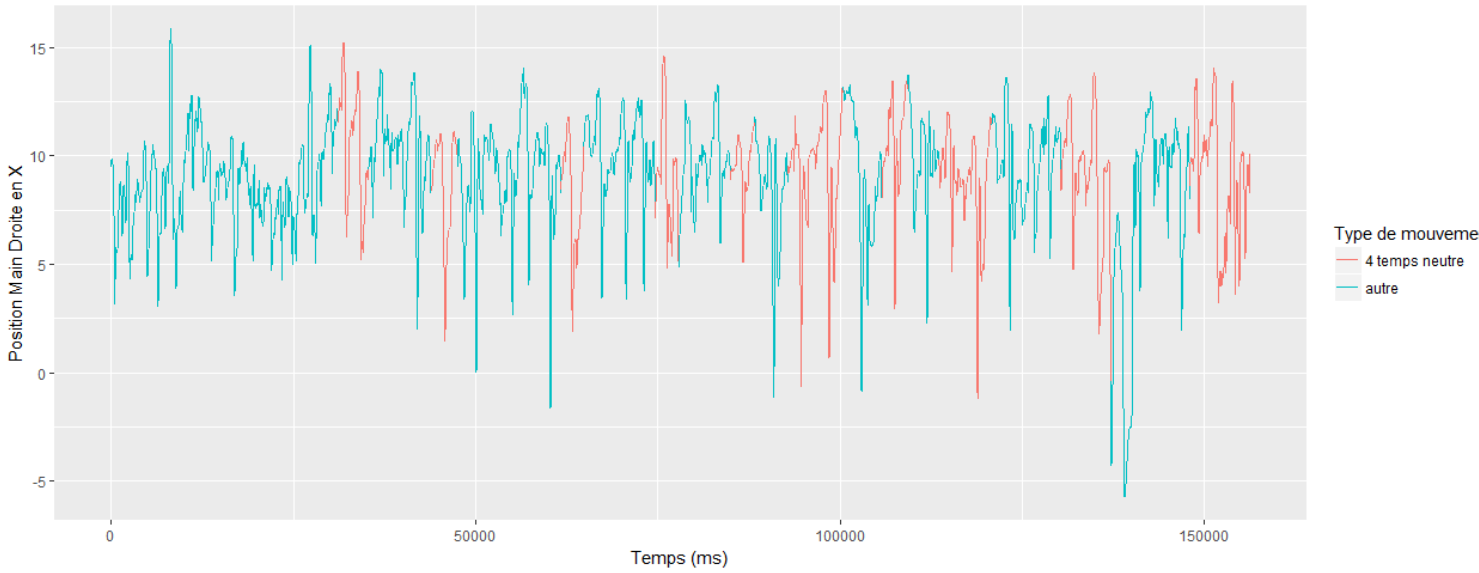


FIGURE 4.3 – Position en X de la main droite en fonction du temps et des différents mouvements

Nous pouvons retrouver le dernier schéma ONE to MANY dans des applications telles que la création de musique. En effet, le modèle part avec seulement une note et génère toute une mélodie, c'est-à-dire qu'il est nourri avec un simple x et est capable de générer toute une séquence o (14), (29).

Nous allons expliquer maintenant le fonctionnement de l'algorithme MANY to MANY avec des séquences égales. L'algorithme MANY to ONE a un fonctionnement très similaire. L'algorithme ONE to MANY et MANY to MANY avec des séquences de longueurs inégales ont des fonctionnements un peu différents que nous n'expliquerons pas dans ce mémoire.

4.3.2 Etape 1 : initialisation et 1ère prédiction : propagation vers l'avant - Forward propagation FP

Nous présentons maintenant les équations de *forward propagation*, noté FP, pour le réseau représenté dans la figure 4.2.

Initialisation

La FP commence par l'initialisation d'un état $h^{(0)}$ et de trois matrices U, V et W. La matrice $\mathbf{U} \in \mathbb{R}^p \times \mathbb{R}^m$ correspond à la liaison entre la couche d'entrée et la couche cachée. Elle contient les poids w associés à chaque variable $x^{(t)}$ dans le calcul de chaque neurone de $h^{(t)}$. La matrice $\mathbf{V} \in \mathbb{R}^m \times \mathbb{R}^C$ correspond à la liaison entre la couche cachée et la couche de sortie. Elle contient les poids w associés à chaque neurone de $h^{(t)}$ dans le calcul de la prédiction $o^{(t)}$. La matrice $\mathbf{W} \in \mathbb{R}^m \times \mathbb{R}^m$ correspond à la liaison entre les couches cachées des deux pas de temps *t-1* et *t*. Elle contient les poids w associés à chaque neurone de $h^{(t-1)}$ dans le calcul de chaque neurone de $h^{(t)}$. C'est ce vecteur qui permet au modèle d'avoir une mémoire temporelle. De plus, comme avec le réseau de neurones classique, à chaque pas de temps t, chaque neurone est calculé avec un biais. Nous définissons alors un vecteur de biais $\mathbf{b} \in \mathbb{R}^m$ qui représente les biais des neurones de la couche cachée $h^{(t)}$ et $\mathbf{c}$ le vecteur $\in \mathbb{R}^C$ qui représente les biais des neurones de la couche de sortie $o^{(t)}$. Ces trois matrices $\mathbf{U}$, $\mathbf{V}$, $\mathbf{W}$ et ces deux vecteurs $\mathbf{b}$, $\mathbf{c}$ sont constants pour tout t (22). Nous avons donc une relation entre les couches $h^{(t-1)}$ et $h^{(t)}$ que nous calculons dans un premier temps :

$$\mathbf{h}^{(t)} = f(\mathbf{b} + \mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{U}\mathbf{x}^{(t)}),$$

où la fonction f est une fonction d'activation. La fonction tangente hyperbolique, $tanh$, ou la fonction unité de rectification linéaire, $ReLu$, sont beaucoup utilisées comme fonctions d'activation. Nous voyons bien ici que la fonction linéaire ne dépend plus uniquement des variables d'entrée x mais elle dépend aussi de la valeur de $h^{(t-1)}$. Ensuite nous calculons les prédictions $\mathbf{o}^{(t)}$ telles que :

$$\mathbf{o}^{(t)} = g(\mathbf{c} + \mathbf{V}\mathbf{h}^{(t)}).$$

La fonction g est la fonction d'activation des neurones de la couche de sortie. Cette fonction dépend du type de données que nous cherchons à prédire. Dans des cas de classification où nous recherchons à prédire plusieurs classes, la fonction softmax est utilisée. Dans notre cas, nous avons seulement deux types de mouvements donc deux classes. Nous utiliserons donc la fonction sigmoïde définie dans la section précédente. Ces deux fonctions ne prédisent pas directement le type de classes mais donnent les probabilités que Y soient dans chacune des K classes.

Calcul de la fonction de coût

A chaque temps t, nous pouvons maintenant calculer la fonction de perte $\boldsymbol{L}$. Lorsque nous sommes en présence de problème de classification, il est courant d'utiliser comme fonction de perte l'entropie croisée. Pour développer nos modèles, nous utiliserons l'entropie croisée pour valeurs binaires de Y qui se définit comme suit :

$$L^{(t)} = y^{(t)} log(o^{(t)}) + (1 - y^{(t)}) log(1 - o^{(t)}).$$

La fonction de coût L_{total} pour une séquence de valeurs $\boldsymbol{x}$ produisant une séquence de prédiction $c = o^{(1)}, ..., o^{(\tau)}$ sera la somme des pertes pour l'ensemble des pas de temps de 1 à τ. La fonction de coût s'écrit donc sous forme d'équation :

$$L_{total} = \sum_{t=1}^{\tau} L^{(t)} \tag{4.5}$$

4.3.3 Etape 2 : rétropropagation dans le temps - Backpropagation Through time BPTT

Le principe de la BPTT est le même que la BP vue précédemment. Le but est toujours de minimiser la fonction de coût et donc de trouver les paramètres U, V, W et b, c qui la minimisent. Cela revient à chercher le gradient de la fonction de coût en fonction de ces différents paramètres (23). Comme le principe est le même, nous n'allons pas réexpliquer la BP. Nous expliquerons uniquement en quoi la BPTT diffère de la BP. Rappelons que les paramètres du modèle (U, V, W, b, c) sont constants entre les différents pas de temps d'une même séquence. Donc pour une séquence, il n'y a que trois matrices U, V, W et deux vecteurs uniques b et c. Pour calculer l'erreur totale (4.5) lors de la FP, nous avons fait la somme des erreurs obtenues. Ainsi pour calculer le gradient de L_{total} en fonction d'un paramètre, nous allons calculer la somme des gradients obtenus à chaque pas de temps t (25). Tout d'abord, nous calculons le gradient de L_{total} en fonction de V :

$$\frac{\partial L_{total}}{\partial V} = \sum_{t=1}^{\tau} \frac{\partial L^{(t)}}{\partial o^{(t)}} \frac{\partial o^{(t)}}{\partial V}.$$

De même, nous pouvons calculer le gradient de L_{total} en fonction de la matrice de biais c :

$$\frac{\partial L_{total}}{\partial c} = \sum_{t=1}^{\tau} \frac{\partial L^{(t)}}{\partial o^{(t)}} \frac{\partial o^{(t)}}{\partial c}.$$

De même nous avons :

$$\frac{\partial L^{(t)}}{\partial W} = \frac{\partial L^{(t)}}{\partial o^{(t)}} \frac{\partial o^{(t)}}{\partial h^{(t)}} \frac{\partial h^{(t)}}{\partial W}.$$

Cependant, $h^{(t)}$ n'est pas une constante. D'après (4.5), $h^{(t)}$ dépend aussi de $h^{(t-1)}$ et de $h^{(t-2)}$ etc... Donc, pour avoir $\frac{\partial L^{(t)}}{\partial W}$ il faut prendre en compte tous les pas de temps précédents. Ainsi :

$$\frac{\partial L^{(t)}}{\partial W} = \sum_{k=1}^{t} \frac{\partial L^{(t)}}{\partial o^{(t)}} \frac{\partial o^{(t)}}{\partial h^{(t)}} \frac{\partial h^{(t)}}{\partial h^{(k)}} \frac{\partial h^{(k)}}{\partial W}.$$

Finalement :

$$\frac{\partial L_{total}}{\partial W} = \sum_{t=1}^{\tau} \frac{\partial L^{(t)}}{\partial W}.$$

Avec le même principe nous obtenons le gradient de la fonction de coût L_{total} en fonction de U en t.

4.3.4 Etape 3 : répétition

Ensuite, les étapes sont les mêmes qu'avec la BP classique. Les gradients nous permettent d'ajuster les valeurs des paramètres. Ensuite, nous pouvons effectuer une nouvelle FP pour calculer de nouvelles prédictions. Nous recommençons ce cercle jusqu'à ce que toutes les époques soient complétées. Les paramètres calculés grâce à la dernière époque sont les paramètres choisis.

Nous venons de voir que pour calculer l'impact d'un changement de valeur de certains paramètres sur la fonction de coût, nous devons prendre en compte l'impact des changements de valeurs à chaque pas de temps de la séquence. Cependant, ce calcul est souvent difficile à réaliser. Nous expliquerons les problèmes liés à cette structure et les solutions pour y remédier.

4.3.5 Problème de disparition du gradient

Certaines des fonctions d'activation non-linéaires très communes ont des gradients compris entre 0 et 1. Par exemple, le gradient de la fonction sigmoïde est compris entre]0 ;0,25], celui de la tangente hyperbolique est compris entre]0 ;1]. Donc pour la fonction sigmoïde, par exemple, le plus grand changement de valeur qui peut être représenté est 0,25. Ainsi, un grand changement de valeur d'un paramètre ne pourra pas se refléter par un grand changement de valeur des couches cachées définies par la fonction d'activation. Ce problème est renforcé par la longueur des séquences. En effet, plus les séquences s'allongent, plus le nombre de pas de temps à prendre en compte dans le calcul du gradient est élevé et donc la multiplication des termes entraîne le gradient à se rapprocher de zéro (17), (24), (6).

4.3.6 Problème d'explosion du gradient

Le problème de l'explosion du gradient est le problème opposé de la disparition du gradient. En fonction de l'initialisation des poids et des biais, et de la fonction d'activation, il se peut que l'on parvienne à avoir des changements de valeurs supérieurs à 1. Dans ces cas, les multiplications successives vont faire croître la valeur du gradient très rapidement (24), (6).

4.3.7 Les solutions

Il existe des solutions partielles à ces problèmes détaillées dans (24), (6), (8).

La fonction *ReLu*

Une solution courante et simple est d'utiliser la fonction d'activation *ReLu*. La dérivée de cette fonction est égale à 1 lorsque x $>$ 0 et égale 0 lorsque x $<$ 0. Cela permet d'avoir un réseau plus stable et donc d'éviter la disparition et l'explosition du gradient.

L'initialisation des poids et des biais

Avant de débuter l'entraînement du modèle, nous avons vu qu'il faut passer par une phase d'initialisation des poids et des biais. Cette phase d'initialisation est très importante. Si elle est bien réussie, elle permet de réduire les problèmes de disparition ou d'explosion du gradient. Il est donc intéressant de fixer les paramètres du modèle en fonction de la fonction d'activation utilisée et non simplement en fonction d'une distribution normale standardisée. Il existe une initialisation pour chaque type de fonction non-linéaire utilisée. Par exemple, lorsque la fonction *ReLu* est utilisée, au lieu d'établir les poids w avec une distribution normale standardisée, nous établissons w à partir d'une distribution normale où la variance suit :

$$Var(w^{[L]}) = \sqrt{2/n^{[L-1]}},$$

où $n^{[L-1]}$ correspond au nombre de valeurs d'entrée dans la couche $L-1$.

Bornement de la norme du gradient aussi appelé *gradient clipping*

Cette technique permet de réduire le problème de l'explosion du gradient. Nous fixons un seuil maximum pour le gradient. Si ce dernier le dépasse, alors le gradient est normalisé pour revenir à une valeur en dessous du seuil (38).

La BPTT tronquée

Il est fréquent d'utiliser la *BTPP tronquée*. Le principe est assez simple : au lieu de calculer la BPTT sur l'ensemble de la séquence, nous divisons la séquence en plusieurs petites séquences. Cela raccourcit et allège les calculs. Par exemple, lorsque nous avons une séquence où t=1 000, nous pouvons diviser cette séquence en 10 pour obtenir 10 nouvelles séquences où t=100. Même si cette solution est facile à implémenter, elle casse les séquences et donc ne prend pas en compte les dépendances à long terme qui peuvent exister entre les données (8), (50).

Utiliser d'autres formes de RNN

Une autre solution est d'utiliser un réseau LSTM (Long Short Term Memory) (25), (26), ou un réseau GRU (Gated Recurrent Unit), plutôt qu'un RNN classique. Ces réseaux sont des sous-catégories de RNN qui permettent de garder des informations pour de plus longues durées. Nous détaillerons la structure du LSTM dans la section suivante. C'est aujourd'hui le modèle le plus populaire, il est plus complexe que le GRU mais permet de garder plus d'informations.

4.4 LSTM

4.4.1 Architecture d'un LSTM

Nous présentons maintenant la structure du modèle LSTM. Nous nous appuierons sur les explications de (52), (26), et (22) qui expliquent tous les trois le fonctionnement d'un LSTM ainsi que la figure 4.4.

Pour commencer, la figure 4.4 représente une cellule LSTM au temps t. Une cellule LSTM représente le traitement d'un pas de temps t. En dessous de la cellule, la légende indique les couleurs et les notations qui seront utilisées dans cette section. Les notations sont les mêmes que précédemment. Cependant, nous constatons qu'il y a une nouvelle variable : $C^{(t-1)}$. Elle correspond à la mémoire de la cellule au temps $t-1$.

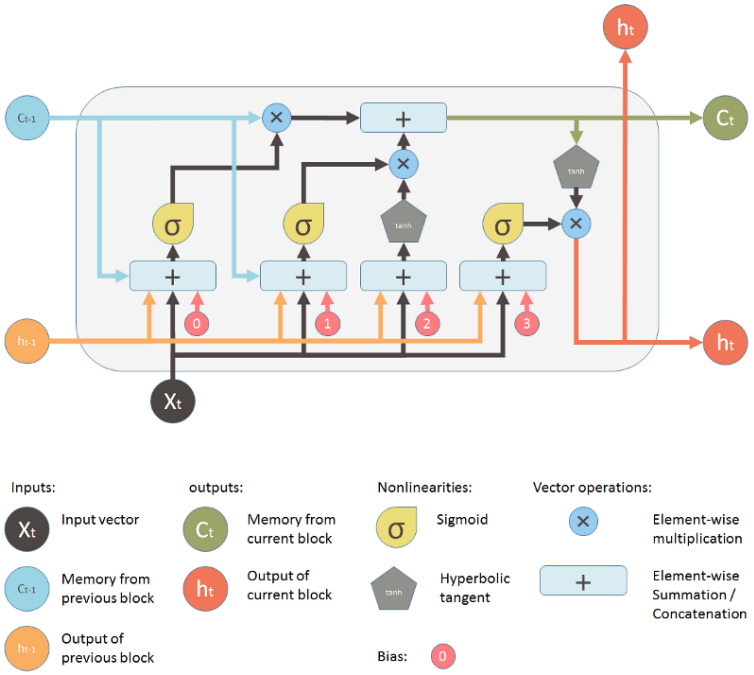


FIGURE 4.4 – Une cellule d'un réseau de neurones de type LSTM, (52)

Une cellule LSTM a trois paramètres d'entrée : $X^t \in \mathbb{R}^p$, $h^{(t-1)} \in \mathbb{R}^m$ et $C^{(t-1)} \in \mathbb{R}^m$ et deux paramètres de sortie : $h^{(t)}$ et $C^{(t)}$ tous deux $\in \mathbb{R}^m$. Pour générer un résultat, la cellule va donc prendre en compte le vecteur de données, l'état de la cellule précédente et la mémoire de la cellule précédente. Il est possible de mettre les cellules les unes à la suite des autres pour schématiser une séquence au cours du temps comme sur la figure 4.5 où trois cellules LSTM sont représentées.

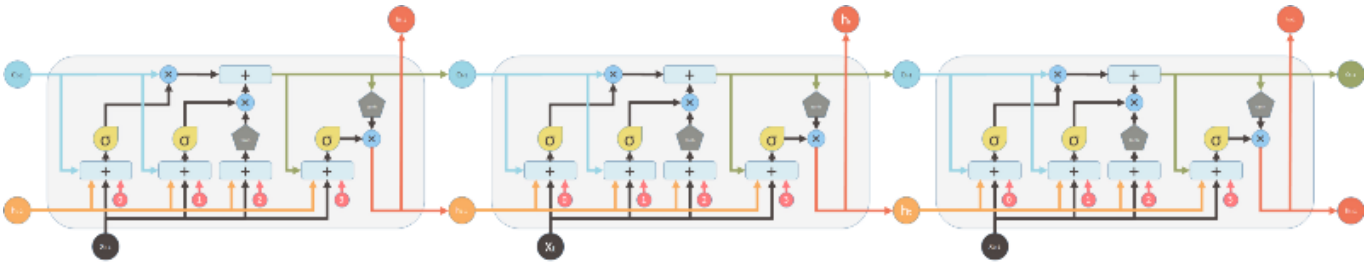


FIGURE 4.5 – Réseau de neurones de type LSTM avec 3 cellules, (52)

Décomposons maintenant le schéma du LSTM pour mieux le comprendre. Tout d'abord regardons comment calculer la mémoire de la cellule. Le trait qui se trouve en haut de la figure 4.6 pourrait être appelé le "tuyau de la mémoire". La mémoire de l'ancienne cellule $C^{(t-1)}$ entre dans ce tuyau pour être modifiée et devenir la mémoire de la cellule $C^{(t)}$.

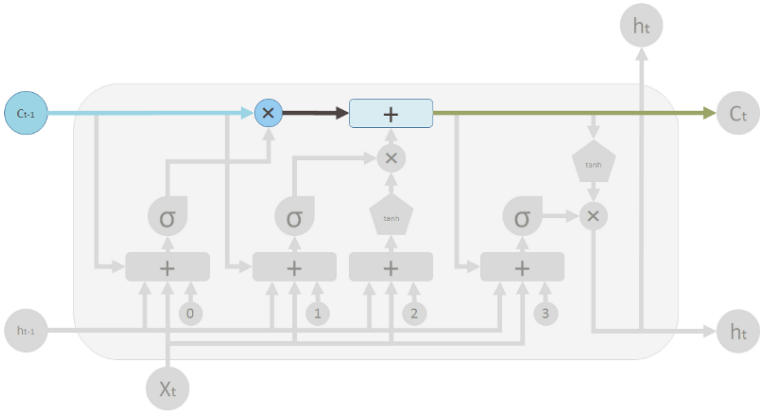


FIGURE 4.6 – Mémoire d'une cellule LSTM, (52)

La première opération que $C^{(t-1)}$ traverse est symbolisée par une croix $\times$. Elle correspond au produit matriciel de la porte d'oubli avec $C^{(t-1)}$. Donc, si nous multiplions $C^{(t-1)}$ par un vecteur proche de 0, cela signifie que nous voulons oublier la plupart de la mémoire de $C^{(t-1)}$. Si la porte d'oubli est égale à 1 alors nous gardons en mémoire $C^{(t-1)}$. Ensuite, le flux de mémoire traverse l'opérateur $+$. Cet opérateur correspond à la somme de ce que nous avons conservé de l'ancienne mémoire et ce que nous ajoutons de la nouvelle mémoire. L'opérateur $\times$ sous l'opérateur $+$ contrôle combien nous ajouterons de la nouvelle mémoire à l'ancienne mémoire. Le résultat de cette opération est la valeur de la nouvelle mémoire $C^{(t)}$. Intéressons-nous maintenant aux différentes portes qui constituent le réseau.

La première correspond à la porte d'oubli présentée sur la figure 4.7. Nous la notons $\Gamma_f \in \mathbb{R}^m$ (f pour *forget* gate).

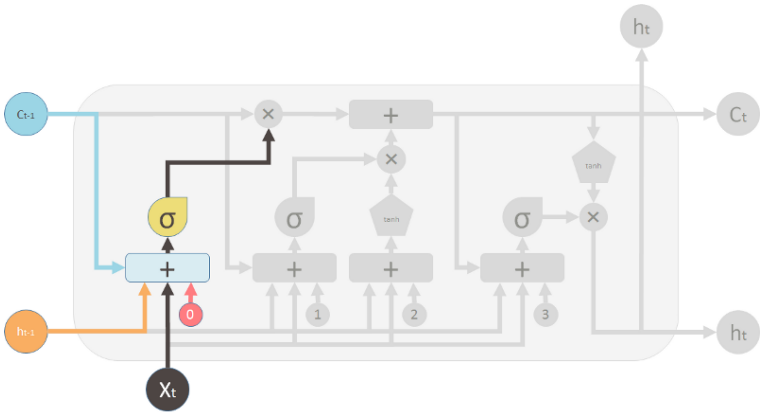


FIGURE 4.7 – Porte d'oubli d'une cellule LSTM, (52)

Elle est contrôlée par un simple réseau de neurones à une couche. Les entrées du réseau de neurones sont $h^{(t-1)}$ la sortie de la cellule précédente, $X^{(t)}$ le vecteur d'entrée pour la cellule actuelle et enfin un vecteur de biais $b_0 \in \mathbb{R}^m$. Ce réseau de neurones a une fonction sigmoïde comme activation, et son vecteur de sortie est la porte d'oubli Γ_f, qui sera appliquée à l'ancienne mémoire $C^{(t-1)}$ par un produit matriciel. Les deux vecteurs $h^{(t-1)}$ et $X^{(t)}$ ont chacun une matrice de poids W_f appartenant respectivement à $\mathbb{R}^m \times \mathbb{R}^m$ et $\mathbb{R}^p \times \mathbb{R}^m$. Nous avons simplifié la notation dans l'équation. Ainsi, nous calculons la porte d'oubli comme suit :

$$\Gamma_f = \sigma(W_f[h^{(t-1)}, X^{(t)}] + b_0).$$

Maintenant, observons la deuxième porte appelée porte de mise à jour, $\Gamma_u \in \mathbb{R}^m$ (u pour *update* gate), car elle met à jour la nouvelle mémoire présentée sur la figure 4.8. Elle est, comme précédemment, calculée grâce à un réseau neuronal simple à couche unique qui prend les mêmes entrées que la porte d'oubli. Cette porte contrôle combien la nouvelle mémoire devrait influencer l'ancienne mémoire. Le vecteur de biais $b_1 \in \mathbb{R}^m$. Comme précédemment, les deux vecteurs $h^{(t-1)}$ et $X^{(t)}$ ont chacun une matrice de poids W_u appartenant respectivement à $\mathbb{R}^m \times \mathbb{R}^m$ et $\mathbb{R}^p \times \mathbb{R}^m$. Nous avons simplifié la notation dans l'équation :

$$\Gamma_u = \sigma(W_u[h^{(t-1)}, X^{(t)}] + b_1),$$

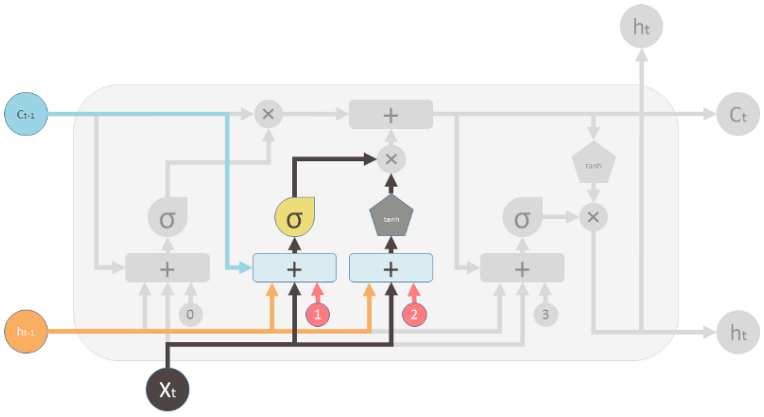


FIGURE 4.8 – Porte de mise à jour d'une cellulle LSTM, (52)

La nouvelle mémoire, $\tilde{C}$, présentée sur la figure 4.9, est calculée par un autre réseau de neurones. C'est aussi un réseau à une couche, mais qui utilise la fonction *tanh* comme fonction d'activation. $\tilde{C}$ forme ensuite un produit matricielle avec Γ_u. $\tilde{C} \in \mathbb{R}^m$. Ce résultat vient dans un deuxième temps s'ajouter à l'ancienne mémoire pour former la nouvelle mémoire $C^{(t)}$. Le vecteur de biais $b_2 \in \mathbb{R}^m$. Aussi, les deux vecteurs $h^{(t-1)}$ et $X^{(t)}$ ont chacun une matrice de poids W_c appartenant respectivement à $\mathbb{R}^m \times \mathbb{R}^m$ et $\mathbb{R}^p \times \mathbb{R}^m$. Nous avons simplifié la notation dans l'équation :

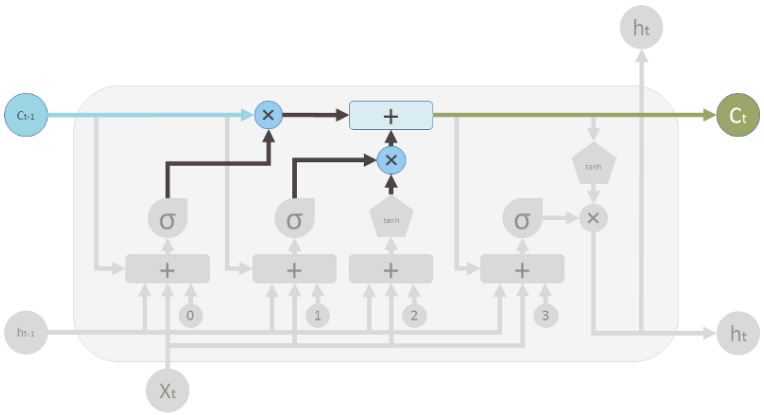


FIGURE 4.9 – Nouvelle mémoire d'une cellule LSTM, (52)

$$\tilde{C}^{(t)} = tanh(W_c[h^{(t-1)}, X^{(t)}] + b_2),$$

$$C^{(t)} = \Gamma_f * C^{(t-1)} + \Gamma_u * \tilde{C}^{(t)}.$$

La dernière étape, représentée sur la figure 4.10, consiste au calcul de l'état de la cellule $h^{(t)}$. Elle est réalisée grâce à une troisième porte. La porte de sortie $\Gamma_o \in \mathbb{R}^m$ (o pour *output* gate). Cette porte est encore calculée par un réseau de neurones à une couche qui prend les mêmes entrées que la porte d'oubli et la porte de mise à jour. Elle contrôle la quantité de nouvelle mémoire qui doit sortir vers la cellule suivante. Le vecteur de biais $b_3 \in \mathbb{R}^m$. Comme précédemment, les deux vecteurs $h^{(t-1)}$ et $X^{(t)}$ ont chacun une matrice de poids W_o appartenant respectivement à $\mathbb{R}^m \times \mathbb{R}^m$ et $\mathbb{R}^p \times \mathbb{R}^m$. Nous avons simplifié la notation dans l'équation :

$$\Gamma_o = \sigma(W_o[h^{(t-1)}, X^{(t)}] + b_3).$$

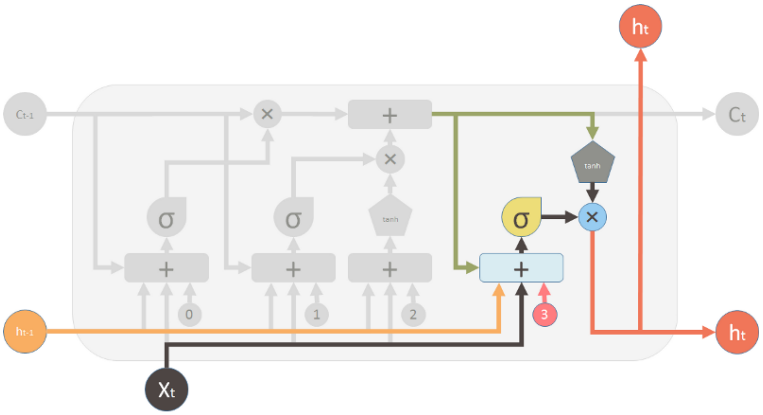


FIGURE 4.10 – Nouvel état d'une cellule LSTM, (52)

Le nouvel état $h^{(t)}$ est finalement calculé par le produit matriciel de Γ_o et la nouvelle mémoire $C^{(t)}$ comme suit :

$$h^{(t)} = \Gamma_o * tanh(C^{(t)}).$$

Pour réaliser la prédiction $o^{(t)}$ que nous souhaitons, il suffit d'appliquer la fonction d'activation g que nous avons choisie en fonction de nos données : softmax, sigmoïde, linéaire etc.

Pour résumé, voici les différentes équations qui permettent de calculer un LSTM :

$$\Gamma_f = \sigma(W_f[h^{(t-1)}, X^{(t)}] + b_0),$$

$$\Gamma_u = \sigma(W_u[h^{(t-1)}, X^{(t)}] + b_1),$$

$$\tilde{C}^{(t)} = tanh(W_c[h^{(t-1)}, X^{(t)}] + b_2),$$

$$C^{(t)} = \Gamma_f * C^{(t-1)} + \Gamma_u * \tilde{C}^{(t)},$$

$$\Gamma_o = \sigma(W_o[h^{(t-1)}, X^{(t)}] + b_3),$$

$$h^{(t)} = \Gamma_o * tanh(C^{(t)}).$$

4.4.2 Rétropropagation et en quoi le LSTM permet d'éviter le problème de la disparition du gradient ?

Pour illustrer notre propos, nous nous baserons sur un modèle qui contient trois cellules LSTM comme sur la figure 4.5. Le principe de la rétropropagation reste le même que dans les sections précédentes. Nous cherchons à optimiser les paramètres W_f, W_u, W_c, W_o et b_0, b_1, b_2, b_3 et donc calculer le gradient de la fonction de coût en fonction de ces paramètres. Regardons dans un premier temps comment calculer le gradient du coût L de la dernière cellule du modèle en fonction du paramètre W_f. Nous pouvons décomposer $\frac{\partial L^{(3)}}{\partial W_f}$:

$$\frac{\partial L^{(3)}}{\partial W_f} = \frac{\partial L^{(3)}}{\partial o^{(3)}} \frac{\partial o^{(3)}}{\partial C^{(3)}} \frac{\partial C^{(3)}}{\partial C^{(2)}} \frac{\partial C^{(2)}}{\partial C^{(1)}} \frac{\partial C^{(1)}}{\partial W_f}.$$

A l'intérieur de cette équation, nous allons maintenant examiner $\frac{\partial C^{(3)}}{\partial C^{(2)}}$ et $\frac{\partial C^{(2)}}{\partial C^{(1)}}$. Nous allons un instant repasser à des notations générales et ainsi analyser $\frac{\partial C^{(t)}}{\partial C^{(t-1)}}$:

$$\frac{\partial C^{(t)}}{\partial C^{(t-1)}} = \frac{\partial}{\partial C^{(t-1)}} (\Gamma_f * C^{(t-1)} + \Gamma_u * \tilde{C}^{(t)}).$$

Or avec la loi sur les dérivées de fonctions :

$$\frac{\partial}{\partial C^{(t-1)}} (\Gamma_f * C^{(t-1)} + \Gamma_u * \tilde{C}^{(t)}) = \Gamma_f \frac{\partial C^{(t-1)}}{\partial C^{(t-1)}} + C^{(t-1)} \frac{\partial \Gamma_f}{\partial C^{(t-1)}} + \Gamma_u \frac{\partial \tilde{C}^{(t)}}{\partial C^{(t-1)}} + \tilde{C}^{(t)} \frac{\partial \Gamma_u}{\partial C^{(t-1)}}.$$

Ou encore :

$$\frac{\partial C^{(t)}}{\partial C^{(t-1)}} = \Gamma_f + C^{(t-1)} \frac{\partial \Gamma_f}{\partial C^{(t-1)}} + \Gamma_u \frac{\partial \tilde{C}^{(t)}}{\partial C^{(t-1)}} + \tilde{C}^{(t)} \frac{\partial \Gamma_u}{\partial C^{(t-1)}}.$$

De plus, Γ_f et Γ_u sont des fonctions sigmoïdes et $\tilde{C}$ est une fonction tangente hyperbolique. Les dérivées sont inférieures à 1. Nous pouvons donc écrire :

$$\frac{\partial C^{(t)}}{\partial C^{(t-1)}} \simeq \Gamma_f,$$

et donc :

$$\frac{\partial L^{(3)}}{\partial W_f} = \frac{\partial L^{(3)}}{\partial o^{(3)}} \frac{\partial o^{(3)}}{\partial C^{(3)}} * \Gamma_f^3 * \Gamma_f^2 \frac{\partial C^{(1)}}{\partial W_f}.$$

Nous pouvons nous demander avec cette équation ce qu'il se passe si l'une des deux portes Γ_f^3 ou Γ_f^2 est proche de zéro ? Le gradient va disparaître et le problème de la disparition du gradient ne sera pas réglé. Cela n'est pas totalement vrai. Car si l'une des portes Γ_f^3 ou Γ_f^2 est proche de zéro, cela veut dire que nous ne souhaitons pas garder en mémoire les informations des cellules précédentes. Donc les cellules précédant cette porte n'ont plus aucune contribution à la prédiction. Par exemple, si Γ_f^3 est égale à 0, cela veut dire que le résultat de la cellule 3 ne sera pas influencé par la cellule 2 ni la cellule 1. Si la valeur des portes est proche de 1, cela signifie que nous souhaitons garder les informations des cellules précédentes. Il n'y a donc plus de problème de disparition du gradient.

Nous venons d'expliquer le fonctionnement d'un réseau de neurones LSTM. Nous utilisons maintenant ces différents modèles pour répondre à notre objectif de recherche : prédire le type de mouvements du chef d'orchestre en fonction de la position de ses bras.

4.5 Résultats des différents modèles

4.5.1 Les différents paramètres utilisés

Pour répondre à notre problématique, nous utilisons les variables de position en X, Y et Z obtenues grâce aux trois capteurs posés du côté droit du chef. En effet, nous cherchons à retrouver le type de mouvements que le chef effectue et que nous avons décrit plus tôt dans le chapitre 1. Ce sont les mouvements du bras droit. Nous n'avons donc pas besoin des capteurs gauches. Nous effectuons uniquement des modèles qui prédisent une sortie binaire : lorsque Y=1 le mouvement est un mouvement 4 temps, lorsque Y=0, le mouvement est un autre mouvement. Le jeu de données d'apprentissage utilise les 200 premières séquences tandis que le jeu de données de test utilise les 67 dernières séquences. La petite taille du jeu de données de départ nous empêchent d'avoir un nombre élevé de séquences et donc limite fortement la taille du jeu d'apprentissage et du jeu test. Il n'était pas possible d'obtenir plus de données dû à la complexité du processus de collecte. C'est également pourquoi nous avons seulement 2 échantillons : apprentissage et test. Nous ne pouvons pas diviser l'échantillon test en 2 échantillons de validation et de test avec aussi peu de données. Une des caractéristiques de notre jeu de données est que les différents mouvements, c'est-à-dire les séquences, sont de longueurs variables. Dans le chapitre 2, nous avons expliqué le principe de l'interpolation. Nous avons donc utilisé cette technique pour ramener toutes les séquences à une même longueur. Nous avons ajusté aux données les modèles RNN et LSTM décrits dans la section précédente. Pour chaque type de modèle, nous avons utilisé trois types de données différentes. Nous avons fait des modèles avec deux longueurs de séquences différentes :

- des données interpolées où chaque séquence compte 315 points (315=57+(516/2) où 57 est le minimum de points et 516 l'étendue),
- des données interpolées où chaque séquence compte 152 points (152 = médiane),
- des données non interpolées, c'est-à-dire que les séquences sont de longueurs différentes.

Les bibliothèques numpy et pandas ont été utilisées en Python pour manipuler les données. Les modèles ont été réalisés avec l'outil open source d'apprentissage automatique Tensorflow et la bibliothèque Keras. Pour les RNN et les LSTM, nous avons utilisé deux types de fonctions d'activation f : la fonction *tanh* et la fonction *ReLu*. Pour les modèles LSTM, nous avons laissé la valeur par défaut de Keras pour le calcul des trois portes qui est la fonction *hard sigmoïde*. La fonction *hard sigmoïde* est une fonction proche de la fonction sigmoïde mais plus rapide à calculer car elle ne requiert pas le calcul de l'exponentielle. Nous avons essayé des modèles avec et sans bornement des paramètres. Lorsque c'était le cas, nous avons établi un bornement des valeurs à 1. Nous avons essayé deux taux d'apprentissage différents : 0,1 et 0,001. Nous avons également fait varier le nombre d'époques en essayant 5, 10, 20, 50, 100, 200 et le nombre de lots avec 5, 15, 25, 35, 45, 50 comme valeurs. Nous avons aussi fait varier le nombre de neurones présents dans les couches cachées des modèles en essayant 5, 10, 20, 30, 50, 100, 200, 300, 1 000 neurones. Nous avons toujours gardé une seule couche cachée. Lorsque nous réalisons un modèle avec cinq neurones, pour les RNN dans la couche cachée, cela signifie qu'à chaque pas de temps il y a une seule couche cachée de cinq neurones. Dans le cas des LSTM, les quatre réseaux de neurones présents à l'intérieur d'une cellule d'un pas de temps t n'ont qu'une couche cachée de cinq neurones. La fonction de coût utilisée dans nos modèles est la fonction d'entropie croisée pour valeurs binaires définies par l'équation (4.5). L'algorithme du gradient utilisé ici est le RMSprop qui est l'algorithme d'optimisation préconisé dans les réseaux de neurones récurrents. Dans le tableau 4.5, nous présentons un récapitulatif des résultats des modèles que nous avons faits. Pour chaque type de modèles, nous avons affiché dans la dernière colonne, le meilleur taux de prédiction que nous avons pu obtenir en modifiant le nombre d'époques, de lots et de neurones. Nous n'avons pas testé toutes les valeurs des époques, des lots et des neurones évoqués précédemment pour chaque type de modèles. C'est pourquoi nous avons précisé le nombre de modèles testés pour chaque type. La semence utilisée pour reproduire les résultats des modèles est : 12345.

4.5.2 Taux naïf

Avant de comparer les différents modèles, nous présentons des scénarios de référence où nous utilisons des modèles simples. Sur le jeu de données d'apprentissage, il y a 83 séquences de mouvements 4 temps neutre (type 1) et 117 séquences de type 0 autres mouvements. Sur le jeu de données test, nous avons 55 séquences de mouvements de type 1, et 12 mouvements de type 0.

TABLE 4.1 – Résultats si le modèle prédit toujours 1 sur le jeu d'apprentissage (mouvement 4 temps neutre)

	prédit		
vérité		0	1
	0	0	117
	1	0	83

Le taux de bonne prédiction pour l'apprentissage est de 41.5%.

TABLE 4.2 – Résultats si le modèle prédit toujours 1 sur le jeu test (mouvement 4 temps neutre)

	prédit		
vérité		0	1
	0	0	12
	1	0	55

Le taux de bonne prédiction pour le test est de 82%. L'écart entre les taux d'apprentissage et de test est énorme. Le modèle a une très bonne prédiction pour l'échantillon test car il y a beaucoup de mouvements 4 temps neutre. Mais le taux d'apprentissage démontre que le modèle est trop simple.

TABLE 4.3 – Résultats si le modèle prédit toujours 0 sur le jeu d'apprentissage (mouvement 4 temps neutre)

	prédit		
vérité		0	1
	0	117	0
	1	83	0

Le taux de bonne prédiction pour l'apprentissage est de 58.5%.

TABLE 4.4 – Résultats si le modèle prédit toujours 0 sur le jeu test (mouvement 4 temps neutre

	prédit		
vérité		0	1
	0	12	0
	1	55	0

Le taux de bonne prédiction pour le test est de 18%. Logiquement, nous observons le même problème. Il y a un très grand écart entre le taux d'apprentissage et le test. Mais cette fois, le taux d'apprentissage est meilleur que le taux de l'échantillon test. Il y a très peu de données type 0 dans le jeu test. Donc si le modèle prédit toujours 0, le taux de prédiction sera médiocre.

4.5.3 Analyse des différents paramètres

Nous présentons ici l'analyse de la variation des différents paramètres des modèles que nous avons effectués. Dans tous les modèles que nous avons pu créer, le paramètre qui discrimine le plus la qualité des modèles est la longueur des séquences utilisées. Lorsque nous gardons des séquences de longueurs différentes, les taux de bonne prédiction des modèles dépassent 24% dans seulement 22% des cas et dépassent 50% dans seulement 8% des cas. Les prédictions s'améliorent lorsque nous utilisons des séquences interpolées de longueurs égales à 315. Les prédictions sont encore meilleures avec des séquences interpolées de longueurs égales à 152. Un deuxième paramètre essentiel est le choix de la fonction d'activation. Nous avons comparé les résultats entre la fonction tangente hyperbolique et la fonction ReLu. Nous voyons dans nos résultats que la fonction ReLu, lorsqu'elle est utilisée dans un LSTM donne de très mauvais résultats. Dans 52% des cas, les modèles ne convergent pas ou ont un taux de prédiction nul. Voici plusieurs explications possibles à ce problème. Tout d'abord, cela pourrait être dû à une mauvaise initiation des poids et des biais dans le modèle, probablement trop élevés. Une deuxième explication possible est le phénomème du "dying ReLu". En effet, la fonction ReLu est égale à 0 pour toutes les valeurs négatives. Un neurone qui a des valeurs négatives va donc mourir car il produira uniquement des valeurs nulles. Si ce phénomème se produit pour un grand nombre de neurones, alors une grande partie du réseau sera inactive. Une autre explication est que la fonction tangente hyperbolique donne empiriquement de meilleurs résultats dans les LSTMs que la fonction ReLu. La fonction ReLu permet d'éviter le phénomène de la disparition du gradient tout comme l'architecture du LSTM. Ainsi, la fonctio ReLu dans un LSTM fait double emploi. La fonction tanh utilisée dans le LSTM permet de mieux généraliser lors de la rétro-propagation, là où la fonction ReLu simplifie trop et peut dégrader la qualité de la prédiction. Presque tous les modèles de LSTMs avec la fonction tanh convergent. Tous les modèles de LSTMs avec la fonction tanh qui utilisent des données interpolées convergent, 58% de ces modèles ont un taux de bonne prédiction supérieur à 50%. Seulement 6 modèles ne convergent pas. Ces modèles utilisent des séquences de longueurs différentes où les performances sont beaucoup plus faibles qu'avec les données interpolées et un taux d'apprentissage de 0.1. En effet, un paramètre qui joue aussi dans la convergence est le taux d'apprentissage. Utiliser un plus petit taux d'apprentissage peut faire augmenter la performance du modèle. Comme vu lors de l'explication de la rétro-propagation, plus le taux d'apprentissage est petit, plus l'ajustement des poids et des biais sera lent et donc la convergence sera probablement plus lente mais plus certaine. Dans nos résultats, 41% des LSTMs avec une fonction ReLu et un taux d'apprentissage de 0.1 convergent alors que 57% de ces mêmes modèles convergent avec un taux de 0.001. Si nous détaillons ces résultats et que nous comparons les modèles avec données interpolées et non interpolées, la différence se confirme. 49% des LSTMs convergent, avec une fonction ReLu, un taux d'apprentissage de 0.1 et des données interpolées alors que seulement 17% des modèles réalisés avec les données non interpolées convergent. 63% des LSTMs, avec une fonction ReLu, un taux d'apprentissage de 0.001 et des données interpolées convergent alors que seulement 31% des modèles réalisés avec les données non interpolées convergent. Nous voyons clairement dans ces résultats qu'utiliser les données non interpolées fait baisser le taux de bonne prédiction et qu'utiliser un plus petit taux d'apprentissage permet d'augmenter la qualité de la prédiction. Les résultats des modèles LSTM avec la fonction tanh renforcent l'idée que les données interpolées sont meilleures et qu'un taux d'apprentissage de 0.001 est meilleur que 0.1. 31% des modèles avec un taux de 0.1 et des données interpolées ont un taux de bonne prédiction supérieur à 50% alors que lorsque nous diminuons le taux d'apprentissage à 0.001, il y a 75% des modèles avec les données interpolées qui ont un taux de prédiction supérieur à 50%. Lorsque nous utilisons les données non interpolées avec un taux d'apprentissage de 0.1, 25% des modèles ont un taux supérieur à 50%. Nous voyons bien que diminuer le taux d'apprentissage permet d'améliorer la performance. Lorsque nous regardons les résultats des modèles RNN avec la fonction ReLu, nous voyons une grande amélioration de la performance. En effet, dans ce cas, il n'y a pas de double emploi. Tous les modèles avec des séquences interpolées convergent. 48% des modèles RNN avec des séquences interpolées et une fonction ReLu ont un taux de bonne prédiction de plus de 50% et 60% des modèles RNN avec

des séquences interpolées et une fonction tanh ont un taux de bonne prédiction de plus de 50%. Nous voyons donc que la fonction tanh a une meilleure performance que la fonction ReLu, dans notre cas, pour l'architecture RNN et LSTM.

Concentrons maintenant notre analyse sur les modèles les plus performants, c'est-à-dire ceux qui utilisent une fonction tanh, des données interpolées et un taux d'apprentissage de 0.001. Comparons maintenant la qualité des modèles avec et sans bornement puis entre les données avec des séquences de 315 et des séquences de 152 et l'architecture du modèle RNN face à l'architecture LSTM. A cette étape, 75% des modèles avec un bornement de 1 ont un taux de prédiction supérieur à 50%. Seulement 61.7% des modèles sans bornement ont un taux de prédiction supérieur à 50%. En décomposant les résultats en fonction de l'architecture du réseau (RNN vs LSTM), nous trouvons que 71% des modèles RNN avec bornement sont supérieurs à 50% alors que 78% des LSTMs avec bornement sont supérieurs à 50%. Sans le bornement, 59% des RNN ont un taux supérieur à 50% alors que 73% des LSTM calculés ont un taux supérieur à 50%. Ces résultats nous montrent d'une part que le bornement a un effet positif sur le taux de prédiction et que l'architecture LSTM est meilleure que l'architecture RNN. Finalement, nous pouvons comparer les modèles avec les deux types de données interpolées. Nous avons testé 36 modèles, avec différentes époques, lots et nombre de neurones. 75% des modèles testés avec les données ayant des séquences de longueurs 315 ont un taux de prédiction supérieur à 50% alors que 81% des modèles testés avec les données ayant des séquences de longueurs 152 ont un taux supérieur à 50%. Logiquement, le meilleur modèle que nous avons trouvé répond à ces critères : c'est un LSTM, il utilise les données interpolées de longueurs 152, il a une fonction tanh, il est borné et a un taux d'apprentissage de 0.001. Nous avons testé différentes valeurs d'époques, de lots et de nombre de neurones. Avec les paramètres que nous avons présentés, nous avons essayé des modèles avec 10, 50 et 100 époques. Nous avons vu pour ces paramètres que le nombre d'époques permettait de faire augmenter la performance du modèle. En effet, le taux de prédiction moyen pour 10 époques est de 50.35%, pour 50 époques est de 62.41% et est de 65.25% pour 100 époques. Le nombre d'époques peut être source de surapprentissage et ainsi dégrader la qualité de la prédiction. Ici ce n'est pas le cas. Nous avons essayé différentes valeurs pour le nombre de neurones. La performance moyenne augmente avec le nombre de neurones. Lorsque le nombre d'unités est 5, le taux de prédiction moyen est de 49.88%, lorsque le nombre d'unités est 20, le taux moyen est de 54.14%, lorsque le nombre d'unités est 50, le taux moyen est 64.30% et lorsque le nombre d'unités est 100, le taux moyen est 69.03%. Nous avons essayé trois valeurs différentes de lots, 5, 25 et 50, pour facliter le calcul des poids. Cependant, aucune tendance ne se dégage. Le meilleur modèle est obtenu avec une taille de lot de 5. La taille du jeu de données peut expliquer le fait qu'aucune tendance ne se dégage pour ce paramètre. Ainsi, le meilleur modèle est un LSTM, effectué avec les données dont les séquences sont de longueur 152, et ayant un taux de bonne prédiction de 89,36%. Le modèle a vu 100 fois les données (époques), il y a 100 neurones dans la couche cachée et 5 lots. La fonction d'activation utilisée est la fonction *tanh*. Les paramètres sont bornés à 1 et le taux d'apprentissage du modèle est de 0,001. Ce modèle prédit mieux les résultats que le modèle simple qui avait un taux de prédiction de 82% sur l'échantillon test.

Le tableau 4.5 ci-dessous est un récapitulatif des modèles que nous avons faits avec les principaux paramètres et le taux de bonne prédiction du meilleur modèle trouvé.

TABLE 4.5 – Tableaux récapitulatif des modèles

Modèle	Longueur des séquences	Activation	Bornement	Taux d'apprentissage	Nombre total de modèles	Nombre de modèles avec taux de bonne prédiction supérieur à 70%	% de modèles avec taux de bonne prédiction supérieur à 70%	Meilleur modèle (taux de bonne prédiction)
RNN	315	tanh	non	0,001	210	11	9,17	76,60
RNN	315	ReLu	non	0,001	188	35	23,81	76,60
RNN	315	tanh	1	0,001	16	1	6,25	70,21
RNN	315	ReLu	1	0,001	16	3	18,75	76,60
RNN	315	tanh	1	0,1	16	11	68,75	76,60
RNN	315	ReLu	1	0,1	36	4	11,11	76,60
LSTM	315	tanh	non	0,001	64	8	22,22	78,72
LSTM	315	ReLu	non	0,001	56	5	8,93	76,60
LSTM	315	tanh	1	0,001	36	7	19,44	80,85
LSTM	315	ReLu	1	0,001	36	4	11,11	76,60
LSTM	315	tanh	1	0,1	36	11	30,56	76,60
LSTM	315	ReLu	1	0,1	36	7	19,44	76,60
RNN	152	tanh	non	0,001	288	27	16,07	76,60
RNN	152	ReLu	non	0,001	288	45	31,25	80,85
RNN	152	tanh	1	0,001	36	0	0	68,09
RNN	152	ReLu	1	0,001	36	9	25	85,11
RNN	152	tanh	non	0,1	36	18	50	76,60
RNN	152	ReLu	non	0,1	36	8	22,22	76,60
RNN	152	tanh	1	0,1	36	18	50	76,60
RNN	152	ReLu	1	0,1	36	8	22,22	76,60
LSTM	152	tanh	non	0,001	36	10	27,78	82,98
LSTM	152	ReLu	non	0,001	36	3	8,33	76,60
LSTM	152	tanh	1	0,001	36	9	25	**89,36**
LSTM	152	ReLu	1	0,001	36	7	19,44	76,60
LSTM	152	tanh	non	0,1	36	12	33,33	76,60
LSTM	152	ReLu	non	0,1	36	6	16,67	76,60
LSTM	152	tanh	1	0,1	36	11	30,56	76,60
LSTM	152	ReLu	1	0,1	36	7	19,44	76,60
RNN	variable	tanh	1	0,0001	16	1	6,25	76,60
RNN	variable	tanh	non	0,0001	16	0	0,00	57,45
RNN	variable	ReLu	non	0,0001	16	0	0,00	23,40
RNN	variable	tanh	non	0,1	16	0	0,00	55,31
RNN	variable	ReLu	non	0,1	16	0	0,00	0
LSTM	variable	tanh	1	0,0001	36	0	0,00	23,40
LSTM	variable	ReLu	1	0,0001	36	0	0,00	59,57
LSTM	variable	tanh	1	0,1	36	9	25,00	76,60
LSTM	variable	ReLu	1	0,1	36	3	8,33	76,60

4.5.4 Analyse des erreurs

TABLE 4.6 – Analyse des erreurs du meilleur modèle sur l'échantillon test

	prédit		
vérité		0	1
	0	10	2
	1	5	50

Malgré le peu de données et la répartition des données très différente dans le jeu de données d'apprentissage et test, le modèle arrive à bien généraliser. Les erreurs sont réparties proportionnellement entre les 2 classes.

Chapitre 5

Discussion

5.1 Retour sur les résultats

Nous avons bien répondu à nos deux objectifs. Tout d'abord, nous avons retrouvé mathématiquement la différence entre le côté droit et le côté gauche. Nos analyses démontrent clairement que les deux bras ne sont pas synchronisés et fonctionnent indépendamment. Nous avons tout d'abord effectué des statistiques descriptives pour avoir un aperçu du contenu de nos données. Nous avons fait des corrélations entre les différentes variables en notre possession. Des corrélations faibles entre un capteur du côté droit et un capteur du côté gauche pour les variables de position dans les différents axes X, Y et Z ont démontré l'indépendance des deux bras. Par exemple, nous avons trouvé que le capteur de la main droite et celui de la main gauche dans l'axe des X avaient une corrélation de -0.19. Logiquement, nous avons trouvé des corrélations élevées pour les capteurs d'un même côté. Par exemple, le capteur de la main droite et du bras droit ont une corrélation de 0.98 dans l'axe des X.

Nous avons ensuite effectué des analyses plus détaillées. Les données en notre possession étant des données de mouvement, il était possible de traiter ces données comme des données fonctionnelles. Nous avons donc, dans un premier temps, transformé nos données en fonctions splines puis nous avons lissé ces fonctions à l'aide de la technique de régularisation pour essayer de réduire le bruit présent dans les données d'origine et ainsi augmenter la qualité des données utilisées dans les analyses. Pour choisir le bon paramètre de lissage, nous avons utilisé le critère GCV (General Cross Validation). Nous avons abouti notre analyse avec un lissage avec 400 fonctions splines et un paramètre de régularisation de 10^{-3}. Nous avons ensuite effectué des analyses par composantes principales sur ces données fonctionnelles. Ces analyses nous ont permis de décomposer et quantifier la variabilité des positions et ainsi de répondre à notre problématique de départ. Dans les résultats, nous retrouvons clairement une distinction entre les capteurs de droite et de gauche. Ce qui montre bien leur indépendance. Dans la direction Y par exemple, un facteur explique 98,8% de la variabilité des positions et ce facteur discrime les capteurs entre gauche et droite très clairement. Le deuxième facteur discrime l'ordre des capteurs sur le côté mais cela est moins clair. Les tendances sont similaires dans les autres directions. Nous avons également fait deux analyses par composantes principales distinctes avec les variables de positions de tous les capteurs droits et de tous les capteurs gauches séparemment. Dans les deux cas, il suffit de deux facteurs pour expliquer plus de 98% de la variabilité. Le premier facteur discrimine les directions des positions (X, Y ou Z), le deuxième facteur explique également les directions et discrimine aussi par rapport à l'ordre des capteurs sur le bras. Nos analyses démontrent que le côté explique très bien la variabilité du mouvement, mais la position du capteur explique aussi une partie de la variabilité du mouvement.

Ensuite, nous avons construit un modèle qui permet de prédire le type de mouvement que le chef d'orchestre effectue à partir des positions de ses mouvements. Pour le construire, nous avons cherché un modèle qui prenne en compte les données séquentielles. En effet, chaque mouvement que le chef effectue avec sa main droite est une séquence que nous pouvons labelliser car il existe des mouvements précis et

définis. Nous avons exploré les modèles de réseaux de neurones récurrents et les modèles LSTMs qui sont des modèles de réseaux de neurones récurrents plus sophistiqués. Pour ce problème, nous avons utilisé un très petit jeu de données car nous n'avions que 267 séquences à notre disposition. Cela a limité la qualité de l'apprentissage des modèles créés et testés. Il est très difficile d'entrainer un modèle avec un si petit jeu de données. Nous avons fait face à des problèmes de convergence qui ne seraient peut-être pas arrivés avec plus de données. Nous n'avons pas pu avoir de jeu de données de validation et test séparés car nous avions trop peu de données. Cela ne nous a donc pas permis de modifier les paramètres et d'évaluer la performance du modèle séparemment. L'évaluation du modèle est donc plus biaisée. Cependant, nous devons mentionner que ce projet reste un projet exploratoire et à but académique. Il nous a permis de découvrir ces modèles, comment les utiliser, comment optimiser les paramètres et les développer. Notre problème était un problème de classification. Parce que la représentation des différentes classes était très déséquilibrée, nous avons choisi de réduire notre classification à une classification binaire : le type de mouvement qui revient 51% du temps et les autres types de mouvements. Nous avons construit différents modèles, avec des architectures différentes en faisant varier les paramètres à l'intérieur du modèle. Nous avons également utilisé différents types de données : des données interpolées avec des séquences égales de longueurs 152 et 315 et des séquences inégales. Finalement nous avons réussi à construire un modèle qui prédit à 89% le type de mouvement du chef d'orchestre à partir des positions de ses mouvements. Ce modèle est un LSTM, effectué avec les données interpolées dont les séquences sont de longueur 152, il a vu 100 fois les données, il y a 100 neurones dans la couche cachée et 5 lots. La fonction d'activation utilisée est la fonction *tanh*. Les paramètres sont bornés à 1 et le taux d'apprentissage du modèle est de 0,001. Avec ce modèle, nous avons répondu à notre deuxième objectif.

5.2 Limites et amélioration du projet

La plus grande limite de notre projet est la petite taille du jeu de données. Nous avons un seul relevé avec un seul chef d'orchestre. Pour développer une analyse plus solide, il faudrait inclure plusieurs relevés de données. Idéalement, les relevés devraient provenir de plusieurs chefs d'orchestre sur plusieurs morceaux de musique différents. Avec plus de données et donc des conclusions plus fiables, il serait intéressant de continuer l'analyse de données fonctionnelles du chapitre 3 pour arriver à différencier le mouvement technique du reste du mouvement du bras droit. Le mouvement technique est le squelette qui revient à chaque mouvement du même type. Pour un chef d'orchestre, plus les mouvements se rapprochent des mouvements présentés par Max Rudolf dans (42), plus ils sont bien effectués. Cette décomposition permettrait donc de voir à quel point les mouvements sont clairs et précis. Pour les chefs d'orchestre apprentis, cela les aiderait à mieux cerner la qualité de leurs mouvements et donc les améliorer plus facilement. Les modèles du chapitre 4 sont des modèles très expérimentaux. Ils nous ont permis de découvrir des modèles de réseaux de neurones sophistiqués tels que les réseaux de neurones récurrents et les LSTMs. En améliorant et en augmentant la taille du jeu de données, il serait intéressant d'essayer ces modèles avec de nouveaux paramètres et d'ajouter de la régularisation. Nos modèles ne contiennent qu'une seule couche de cellules RNN ou LSTM. Il serait intéressant d'essayer des modèles plus complexes avec plusieurs couches si le jeu de données le permet. De plus, nous n'avons pas exploré le modèle GRU. Ce modèle pourrait être une alternative à explorer car c'est un modèle plus simple que le LSTM mais plus sophistiqué que le RNN basique.

Annexe A

Chapitre 2 : les données de capteurs de mouvements et le chef d'orchestre

A.1 Présentation des données

Table A.1 – Tableau récapitulatif des variables

Nom	Description
RH_XP, RH_YP, RH_ZP	Position du capteur de la main droite dans les axes X, Y et Z
RFA_XP, RFA_YP, RFA_ZP	Position du capteur de l'avant-bras droit dans les axes X, Y et Z
RA_XP, RA_YP, RA_ZP	Position du capteur du bras droit dans les axes X, Y et Z
LH_XP, LH_YP, LH_ZP	Position du capteur de la main gauche dans les axes X, Y et Z
LFA_XP, LFA_YP, LFA_ZP	Position du capteur de l'avant-bras gauche dans les axes X, Y et Z
LA_XP, LA_YP, LA_ZP	Position du capteur du bras gauche dans les axes X, Y et Z
RH_Xacc, RH_Yacc, RH_Zacc	Accélération du capteur de la main droite dans les axes X, Y et Z en m/s^2
RFA_Xacc, RFA_Yacc, RFA_Zacc	Accélération du capteur de l'avant-bras droit dans les axes X, Y et Z m/s^2
RA_Xacc, RA_Yacc, RA_Zacc	Accélération du capteur du bras droit dans les axes X, Y et Z en m/s^2
LH_Xacc, LH_Yacc, LH_Zacc	Accélération du capteur de la main gauche dans les axes X, Y et Z en m/s^2
LFA_Xacc, LFA_Yacc, LFA_Zacc	Accélération du capteur de l'avant-bras gauche dans les axes X, Y et Z en m/s^2
LA_Xacc, LA_Yacc, LA_Zacc	Accélération du capteur du bras gauche dans les axes X, Y et Z en m/s^2
RH_Xspeed, RH_Yspeed, RH_Zspeed	Vitesse de rotation du capteur de la main droite dans les axes X, Y et Z en $radian/s$
RFA_Xspeed, RFA_Yspeed, RFA_Zspeed	Vitesse de rotation du capteur de l'avant-bras droit dans les axes X, Y et Z en $radian/s$
RA_Xspeed, RA_Yspeed, RA_Zspeed	Vitesse de rotation du capteur du bras droit dans les axes X, Y et Z en $radian/s$
LH_Xspeed, LH_Yspeed, LH_Zspeed	Vitesse de rotation du capteur de la main gauche dans les axes X, Y et Z en $radian/s$

TABLE A.1 – Tableau récapitulatif des variables

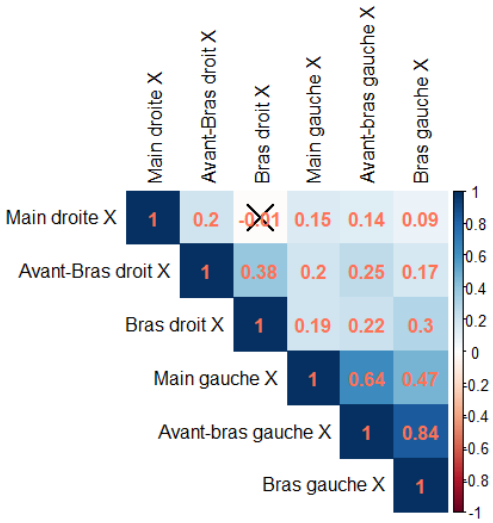
LFA_Xspeed, LFA_Yspeed, LFA_Zspeed	Vitesse de rotation du capteur de l'avant-bras gauche dans les axes X, Y et Z en $radian/s$
LA_Xspeed, LA_Yspeed, LA_Zspeed	Vitesse de rotation du capteur du bras gauche dans les axes X, Y et Z en $radian/s$
RH_Xmag RH_Ymag RH_Zmag	Le nord magnétique du capteur de la main droite dans les axes X, Y et Z
RFA_Xmag RFA_Ymag RFA_Zmag	Le nord magnétique du capteur de l'avant-bras droit dans les axes X, Y et Z
RA_Xmag RA_Ymag RA_Zmag	Le nord magnétique du capteur du bras droit dans les axes X, Y et Z
LH_Xmag LH_YmagLH_Zmag	Le nord magnétique du capteur de la main gauche dans les axes X, Y et Z
LFA_Xmag LFA_Ymag LFA_Zmag	Le nord magnétique du capteur de l'avant-bras gauche dans les axes X, Y et Z
LA_Xmag LA_Ymag LA_Zmag	Le nord magnétique du capteur du bras gauche dans les axes X, Y et Z
RH_roll RH_pitch RH_yaw	Angle d'Euler roll, pitch, yaw du capteur de la main droite
RFA_roll RFA_pitch RFA_yaw	Angle d'Euler roll, pitch, yaw du capteur de l'avant-bras droit
RA_roll RA_pitch RA_yaw	Angle d'Euler roll, pitch, yaw du capteur du bras droit
LH_roll LH_pitch LH_yaw	Angle d'Euler roll, pitch, yaw du capteur de la main gauche
LFA_roll LFA_pitch LFA_yaw	Angle d'Euler roll, pitch, yaw du capteur de l'avant-bras gauche
LA_roll LA_pitch LA_yaw	Angle d'Euler roll, pitch, yaw du capteur du bras gauche
temps_mesure_ms	Temps dans l'enregistrement du début de la mesure en millisecondes
temps_min	Temps dans l'enregistrement du début de la mesure en minutes
temps_mesure_s	Temps dans l'enregistrement du début de la mesure en secondes
duree_mesure_ms	Durée totale de la mesure en millisecondes
Page	Numéro de la page de la partition
id_mesure	Numéro de la mesure
nb_temps	Nombre de temps dans la mesure (notation numérique au début de chaque mesure, lorsqu'il n'y a qu'un chiffre correspond à ce dernier, lorsqu'il y en a 2, correspond au chiffre du haut)
temps_dénom	Note de référence. 2= blanche ; 4=noire ; 8=croche
Move_name	Nom du mouvement effectué par le chef d'orchestre durant la mesure
move_id	Identifiant donné à chaque type mouvement ; 1=5beat_alternate_3+2 ; 2=2beat_neutre ; 3=3beat_neutre ; 4=4beat_neutre ; 5=autre
move_id_bi	Identifiant binaire donné à chaque mouvement du chef d'orchestre 1=4beat_neutre ; 0=autre
tempo	Valeur du tempo
time	Temps dans l'enregistrement de chaque relevé
temps_ref_st	Temps écoulé dans le morceau avec comme point de départ 0ms

TABLE A.1 – Tableau récapitulatif des variables

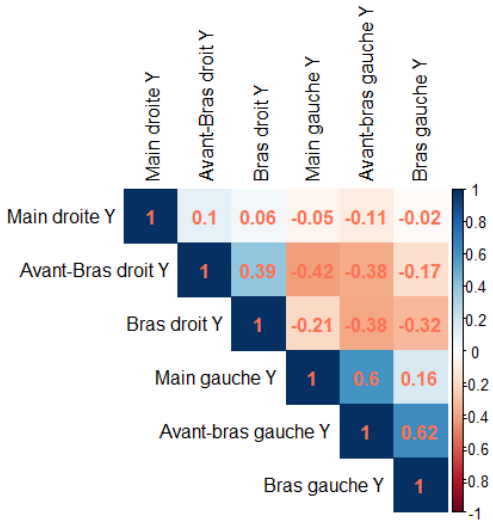
temps_mesure_ms_st	Temps dans le morceau de début de la mesure en millisecondes avec comme point de départ 0ms
duree_mesure	Durée écoulée de la mesure en millisecondes.

A.2 Corrélation des autres variables

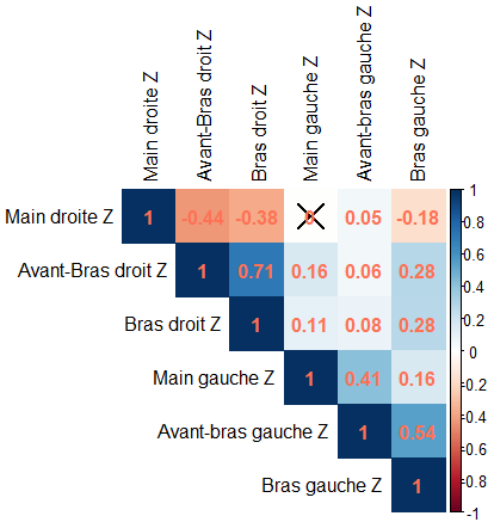
A.2.1 Accélération



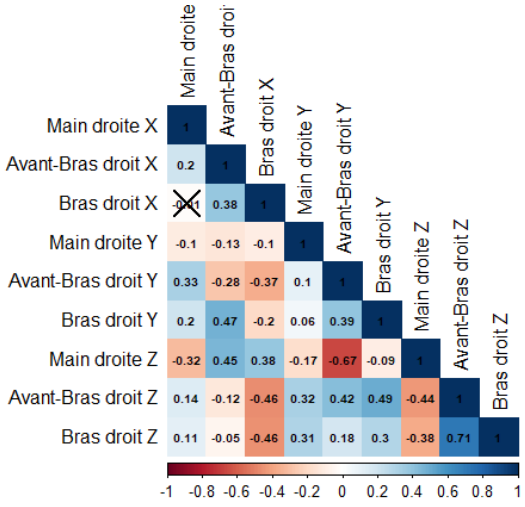
(a) Corrélation de Pearson de l'accélération des 3 capteurs du côté droit dans les 3 axes X, Y et Z à travers l'ensemble du morceau



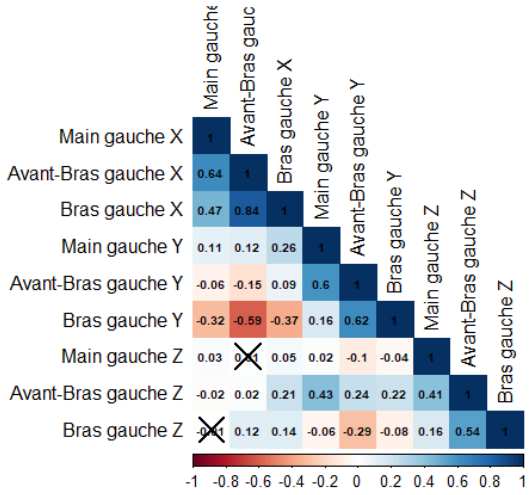
(b) Corrélation de Pearson de l'accélération des 3 capteurs du côté gauche dans les 3 axes X, Y et Z à travers l'ensemble du morceau



(c) Corrélation de Pearson de l'accélération des 6 capteurs dans l'axe des X à travers l'ensemble du morceau



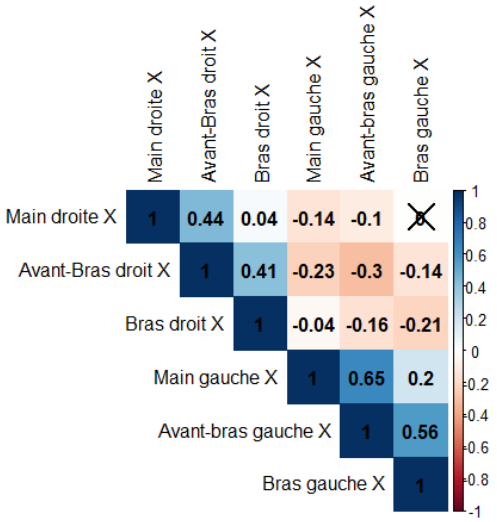
(d) Corrélation de Pearson de l'accélération des 6 capteurs dans l'axe des Y à travers l'ensemble du morceau



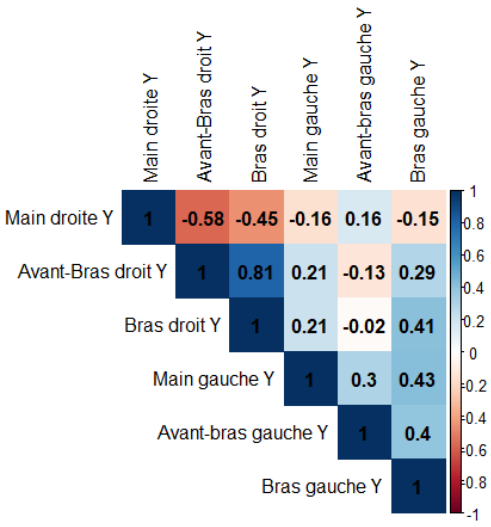
(e) Corrélation de Pearson de l'accélération des 6 capteurs dans l'axe des Z à travers l'ensemble du morceau

FIGURE A.1 – Corrélation de Pearson de l'accélération à travers l'ensemble du morceau

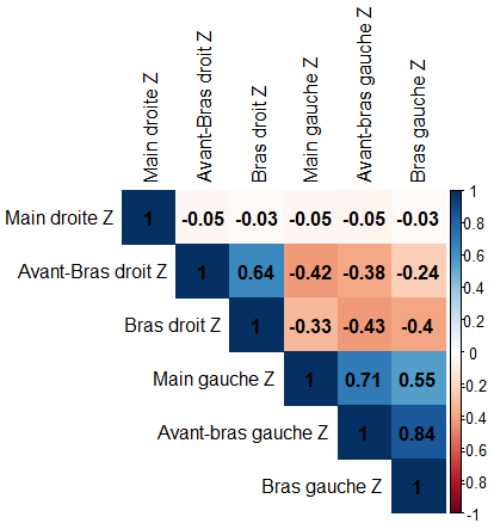
A.2.2 Vitesse de rotation



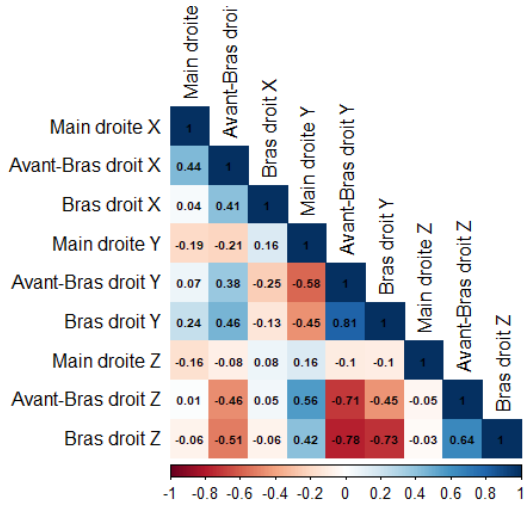
(a) Corrélation de Pearson de la vitesse de rotation des 6 capteurs dans l'axe des X à travers l'ensemble du morceau



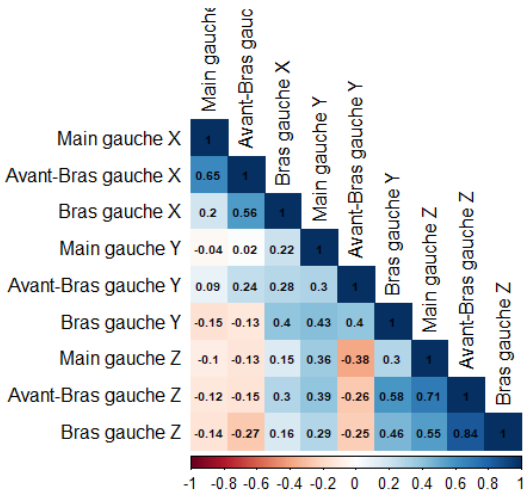
(b) Corrélation de Pearson de la vitesse de rotation des 6 capteurs dans l'axe des Y à travers l'ensemble du morceau



(c) Corrélation de Pearson de la vitesse de rotation des 6 capteurs dans l'axe des Z à travers l'ensemble du morceau



(d) Corrélation de Pearson de la vitesse de rotation des 3 capteurs droits dans les 3 axes X, Y et Z à travers l'ensemble du morceau



(e) Corrélation de Pearson de la vitesse de rotation des 3 capteurs gauches dans les 3 axes X, Y et Z à travers l'ensemble du morceau

FIGURE A.2 – Corrélation de Pearson de la vitesse de rotation à travers l'ensemble du morceau

Annexe B

Chapitre 3 : inférence et données fonctionnelles

B.1 Tableaux

TABLE B.1 – GCV des données lissées pour la variable position des capteurs du côté gauche

Bases	λ	Variables								
		Avant-bras en X	Avant-bras en Y	Avant-bras en Z	Main en X	Main en Y	Main en Z	Bras en X	Bras en Y	Bras en Z
400	1e+06	5,29	4,30	10,08	14,99	14,25	51,57	22,80	19,25	78,65
400	1e+03	5,12	4,15	9,78	14,58	13,83	50,09	22,30	18,73	76,42
400	10	5,12	4,15	9,78	14,58	13,83	50,09	22,30	18,73	76,42
400	1e-03	5,07	4,02	9,77	14,30	13,53	50,26	21,92	18,43	76,81
400	1e-06	5,07	4,02	9,77	14,30	13,53	50,26	21,92	18,43	76,81
400	0	5,07	4,02	9,77	14,30	13,53	50,26	21,92	18,43	76,81
500	1e+06	4,08	3,35	8,00	12,14	11,41	41,17	19,11	15,57	63,22
500	1e+03	3,48	2,92	6,93	10,74	10,13	35,79	17,29	13,91	55,26
500	10	3,48	2,92	6,93	10,74	10,13	35,79	17,29	13,91	55,26
500	1e-03	3,67	2,87	7,50	11,02	10,04	38,62	17,66	13,77	59,26
500	1e-06	3,67	2,87	7,50	11,02	10,04	38,62	17,66	13,77	59,26
500	0	3,67	2,87	7,50	11,02	10,04	38,62	17,66	13,77	59,26
600	1e+06	3,70	2,99	7,36	11,31	10,31	37,85	18,00	14,18	58,27
600	1e+03	2,71	2,21	5,60	9,03	7,94	28,97	15,03	11,15	45,00
600	10	2,71	2,21	5,60	9,03	7,94	28,97	15,03	11,15	45,00
600	1e-03	2,79	2,27	5,36	9,25	8,15	28,10	15,29	11,36	43,79
600	1e-06	2,79	2,27	5,36	9,25	8,15	28,10	15,29	11,36	43,79
600	0	2,79	2,27	5,36	9,25	8,15	28,10	15,29	11,36	43,79
700	1e+06	3,50	2,83	6,87	10,79	9,81	35,51	17,26	13,51	54,65
700	1e+03	2,20	1,79	4,29	7,75	6,67	22,66	13,13	9,48	35,17
700	10	2,20	1,79	4,29	7,75	6,67	22,66	13,13	9,48	35,17
700	1e-03	2,02	1,75	4,01	7,37	6,45	21,21	12,66	9,16	32,93
700	1e-06	2,02	1,75	4,01	7,37	6,45	21,21	12,66	9,16	32,93
700	0	2,02	1,75	4,01	7,37	6,45	21,21	12,66	9,16	32,93
800	1e+06	3,34	2,70	6,58	10,37	9,40	34,04	16,64	12,99	52,39
800	1e+03	1,62	1,36	3,36	6,07	5,21	17,58	10,64	7,64	27,27
800	10	1,62	1,36	3,36	6,07	5,21	17,58	10,64	7,64	27,27
800	1e-03	1,62	1,36	3,36	6,07	5,21	17,58	10,64	7,64	27,27
800	1e-06	1,62	1,36	3,36	6,07	5,21	17,58	10,64	7,64	27,27
800	0	1,62	1,36	3,36	6,07	5,21	17,58	10,64	7,64	27,27

TABLE B.2 – GCV des données lissées pour la variable accélération des capteurs du côté droit

Bases	λ	Variables								
		Main en X	Avant-bras en X	Bras en X	Main en Y	Avant-bras en Y	Bras en Y	Main en Z	Avant-bras en Z	Bras en Z
400	1e-03	79,05	32,85	10,72	82,07	66,75	27,48	560,78	56,11	13,26
400	1e-06	79,05	32,85	10,72	82,07	66,75	27,48	560,78	56,11	13,26
400	0	79,05	32,85	10,72	82,07	66,75	27,48	560,78	56,11	13,26
500	1e-03	78,66	32,72	10,52	81,45	66,42	27,37	562,51	56,21	13,20
500	1e-06	78,66	32,72	10,52	81,45	66,42	27,37	562,51	56,21	13,20
500	0	78,66	32,72	10,52	81,45	66,42	27,37	562,51	56,21	13,20
600	1e-03	78,15	32,57	10,37	80,64	66,25	27,31	564,06	56,20	13,11
600	1e-06	78,15	32,57	10,37	80,64	66,25	27,31	564,06	56,20	13,11
600	0	78,15	32,57	10,37	80,64	66,25	27,31	564,06	56,20	13,11
700	1e-03	77,86	32,48	10,18	80,42	66,27	27,34	565,66	56,28	13,10
700	1e-06	77,86	32,48	10,18	80,42	66,27	27,34	565,66	56,28	13,10
700	0	77,86	32,48	10,18	80,42	66,27	27,34	565,66	56,28	13,10
800	1e-03	77,58	32,45	10,09	80,29	66,30	27,36	567,70	56,43	13,12
800	1e-06	77,58	32,45	10,09	80,29	66,30	27,36	567,70	56,43	13,12
800	0	77,58	32,45	10,09	80,29	66,30	27,36	567,70	56,43	13,12

TABLE B.3 – GCV des données lissées pour la variable accélération des capteurs du côté gauche

Bases	λ	Variables								
		Main en X	Avant-bras en X	Bras en X	Main en Y	Avant-bras en Y	Bras en Y	Main en Z	Avant-bras en Z	Bras en Z
400	1e-03	63,56	21,76	11,48	94,25	32,25	16,89	104,28	9,96	5,41
400	1e-06	63,56	21,76	11,48	94,25	32,25	16,89	104,28	9,96	5,41
400	0	63,56	21,76	11,48	94,25	32,25	16,89	104,28	9,96	5,41
500	1e-03	59,32	18,41	9,83	93,51	32,00	16,09	102,64	9,62	5,27
500	1e-06	59,32	18,41	9,83	93,51	32,00	16,09	102,64	9,62	5,27
500	0	59,32	18,41	9,83	93,51	32,00	16,09	102,64	9,62	5,27
600	1e-03	56,81	15,85	8,31	92,73	31,87	15,64	101,27	9,34	5,19
600	1e-06	56,81	15,85	8,31	92,73	31,87	15,64	101,27	9,34	5,19
600	0	56,81	15,85	8,31	92,73	31,87	15,64	101,27	9,34	5,19
700	1e-03	53,14	14,03	7,31	92,20	31,71	15,25	99,65	9,15	5,14
700	1e-06	53,14	14,03	7,31	92,20	31,71	15,25	99,65	9,15	5,14
700	0	53,14	14,03	7,31	92,20	31,71	15,25	99,65	9,15	5,14
800	1e-03	51,31	13,07	6,85	91,96	31,65	15,04	98,93	9,04	5,10
800	1e-06	51,31	13,07	6,85	91,96	31,65	15,04	98,93	9,04	5,10
800	0	51,31	13,07	6,85	91,96	31,65	15,04	98,93	9,04	5,10

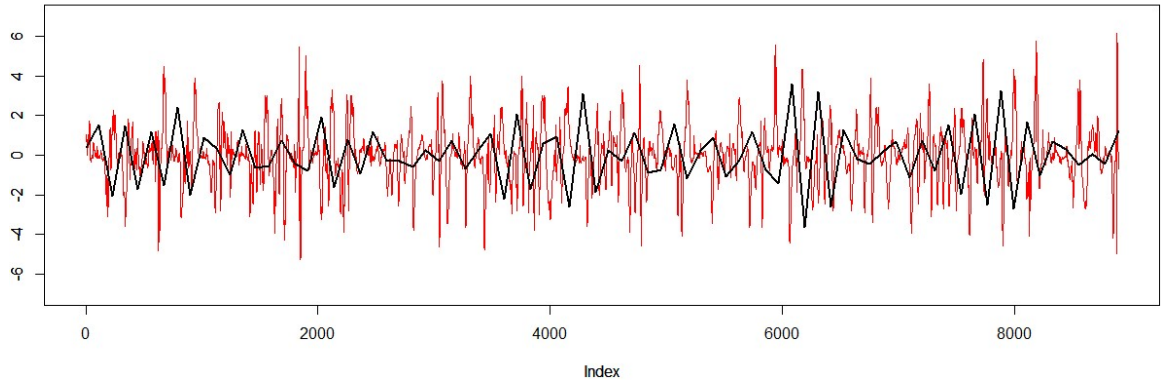
TABLE B.4 – GCV des données lissées pour la variable vitesse de rotation des capteurs du côté droit

Bases	λ	Variables								
		Main en X	Avant-bras en X	Bras en X	Main en Y	Avant-bras en Y	Bras en Y	Main en Z	Avant-bras en Z	Bras en Z
400	1e-03	12,46	3,23	1,22	10,71	3,95	1,86	10,16	4,17	2,62
400	1e-06	12,46	3,23	1,22	10,71	3,95	1,86	10,16	4,17	2,62
400	0	12,46	3,23	1,22	10,71	3,95	1,86	10,16	4,17	2,62
500	1e-03	12,46	3,21	1,20	10,74	3,95	1,85	10,14	4,17	2,61
500	1e-06	12,46	3,21	1,20	10,74	3,95	1,85	10,14	4,17	2,61
500	0	12,46	3,21	1,20	10,74	3,95	1,85	10,14	4,17	2,61
600	1e-03	12,47	3,21	1,20	10,75	3,94	1,85	10,09	4,16	2,60
600	1e-06	12,47	3,21	1,20	10,75	3,94	1,85	10,09	4,16	2,60
600	0	12,47	3,21	1,20	10,75	3,94	1,85	10,09	4,16	2,60
700	1e-03	12,48	3,20	1,19	10,71	3,92	1,85	10,03	4,14	2,56
700	1e-06	12,48	3,20	1,19	10,71	3,92	1,85	10,03	4,14	2,56
700	0	12,48	3,20	1,19	10,71	3,92	1,85	10,03	4,14	2,56
800	1e-03	12,50	3,19	1,18	10,67	3,90	1,85	9,96	4,09	2,53
800	1e-06	12,50	3,19	1,18	10,67	3,90	1,85	9,96	4,09	2,53
800	0	12,50	3,19	1,18	10,67	3,90	1,85	9,96	4,09	2,53

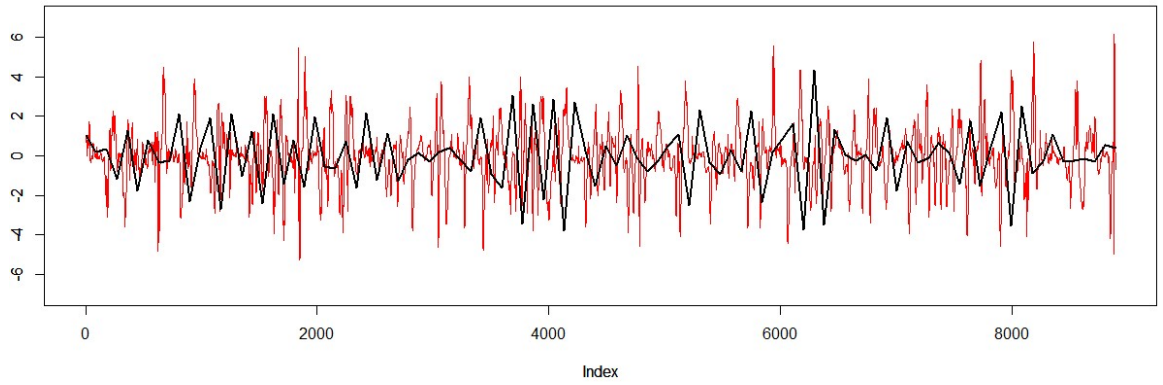
TABLE B.5 – GCV des données lissées pour la variable vitesse de rotation des capteurs du côté gauche

Bases	λ	Variables								
		Main en X	Avant-bras en X	Bras en X	Main en Y	Avant-bras en Y	Bras en Y	Main en Z	Avant-bras en Z	Bras en Z
400	1e-03	10,05	2,06	5,41	10,69	1,34	1,13	8,80	6,07	1,98
400	1e-06	10,05	2,06	5,41	10,69	1,34	1,13	8,80	6,07	1,98
400	0	10,05	2,06	5,41	10,69	1,34	1,13	8,80	6,07	1,98
500	1e-03	9,97	2,03	5,27	10,48	1,31	1,07	8,66	5,81	1,87
500	1e-06	9,97	2,03	5,27	10,48	1,31	1,07	8,66	5,81	1,87
500	0	9,97	2,03	5,27	10,48	1,31	1,07	8,66	5,81	1,87
600	1e-03	9,87	2,01	5,19	10,23	1,28	1,03	8,48	5,54	1,74
600	1e-06	9,87	2,01	5,19	10,23	1,28	1,03	8,48	5,54	1,74
600	0	9,87	2,01	5,19	10,23	1,28	1,03	8,48	5,54	1,74
700	1e-03	9,77	1,98	5,14	9,89	1,25	0,98	8,33	5,27	1,63
700	1e-06	9,77	1,98	5,14	9,89	1,25	0,98	8,33	5,27	1,63
700	0	9,77	1,98	5,14	9,89	1,25	0,98	8,33	5,27	1,63
800	1e-03	9,72	1,96	5,10	9,65	1,22	0,94	8,17	5,05	1,54
800	1e-06	9,72	1,96	5,10	9,65	1,22	0,94	8,17	5,05	1,54
800	0	9,72	1,96	5,10	9,65	1,22	0,94	8,17	5,05	1,54

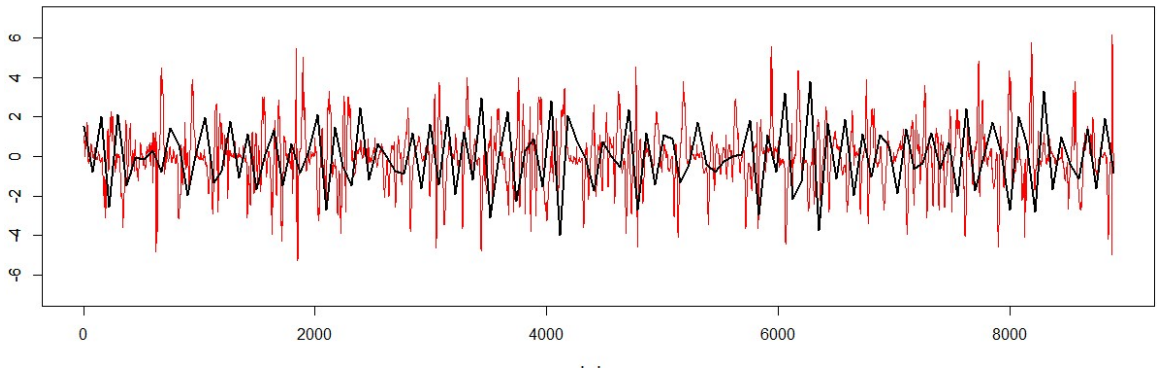
B.2 Graphiques



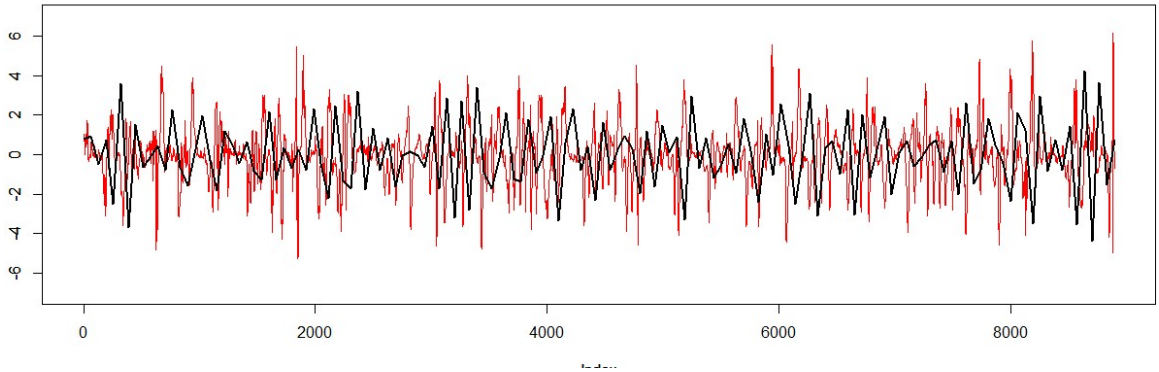
(a) Données originales et lissées de LA_Zspeed K= 400 et lambda = 0,001



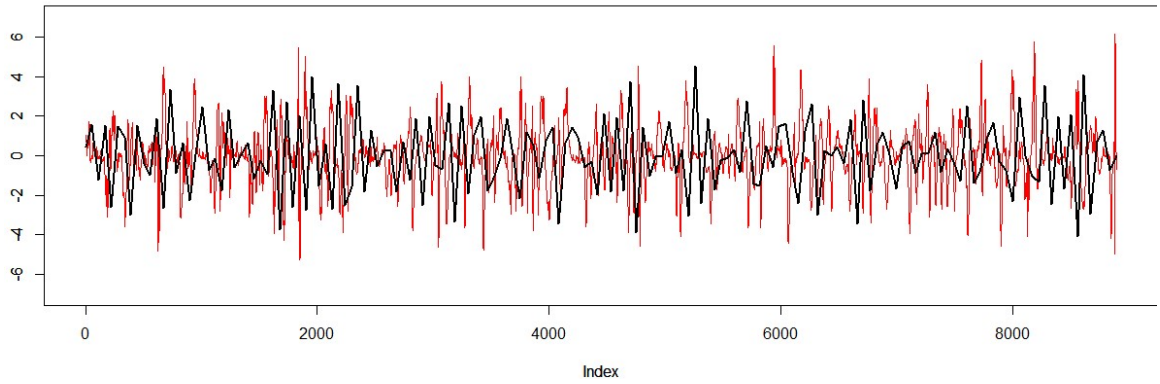
(b) Données originales et lissées de LA_Zspeed K= 500 et lambda = 0,001



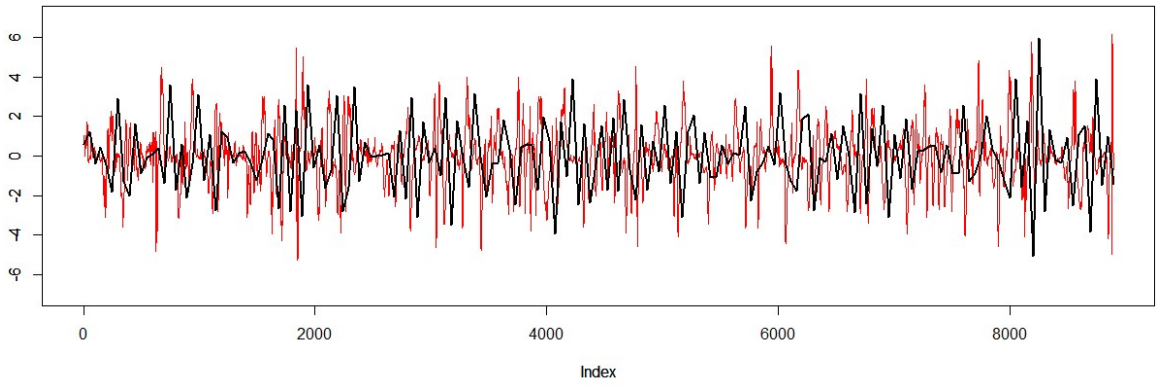
(c) Données originales et lissées de LA_Zspeed K= 600 et lambda = 0,001



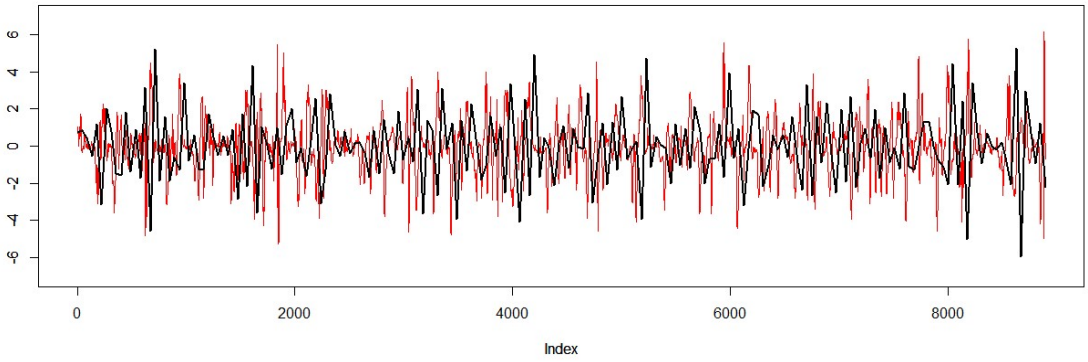
(d) Données originales et lissées de LA_Zspeed K= 700 et lambda = 0,001



(e) Données originales et lissées de LA_Zspeed K= 800 et lambda = 0,001



(f) Données originales et lissées de LA_Zspeed K= 900 et lambda = 0,001



(g) Données originales et lissées de LA_Zspeed K= 1000 et lambda = 0,001

FIGURE B.1 – Comparaison entre les données originales et lissées de la position de la main droite en X avec différentes valeurs de lissage pour les premiers 20% des données

Bibliographie

[1] 3Blue1Brown (2017a). But what *is* a neural network ? | chapter 1, deep learning. video, 3Blue1Brown, Youtube. Récupéré de : https ://www.youtube.com/watch ?v=aircAruvnKk.

[2] 3Blue1Brown (2017b). Gradient descent, how neural networks learn | chapter 2, 3blue1brown, deep learning. video, Youtube. Récupéré de : https ://www.youtube.com/watch ?v=IHZwWFHWa-w.

[3] 3Blue1Brown (2017c). What is backpropagation really doing ? | chapter 3, deep learning. video, 3Blue1Brown, Youtube. Récupéré de : https ://www.youtube.com/watch ?v=Ilg3gGewQ5U.

[4] Bellomia, P. (2016). Nouvelle formule du concert symphonique : projections en temps réel d'images et de couleurs puisées dans la chironomie - chef d'orchestre virtuel assisté par ordinateur. Fonds de recherche du Québec - Société et culture. Récupéré le 15 septembre 2017 de : http ://www.frqsc.gouv.qc.ca/fr/partenariat/nos-projets-de-recherche/projet/nouvelle-formule-du-concert-symphonique-projections-en-temps-reel-d-images-et-de-couleurs-puisees-dans-la-chironomie-chef-d-orchestre-virtuel-assiste-par-ordinateur-twvnw9ju1468263544853.

[5] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer-Verlag, Berlin, Heidelberg. chap. 5.

[6] Britz, D. (8 octobre 2015). Recurrent neural networks tutorial, part 3 – backpropagation through time and vanishing gradients. *wildml.com*. Récupéré de : http ://www.wildml.com/2015/10/recurrent-neural-networks-tutorial-part-3-backpropagation-through-time-and-vanishing-gradients/.

[7] Brownlee, J. (20 janvier 2016). Boston symphony orchestra hopes ipads will bring kids back to classical. *cultofmac.com*. Récupéré de : https ://www.cultofmac.com/407837/boston-symphony-orchestra-hopes-ipads-will-bring-kids-back-to-classical/.

[8] Brownlee, J. (23 juin 2017). A gentle introduction to backpropagation through time. *machinelearningmastery.com*, Section Long Short-Term Memory Networks. Récupéré de : https ://machinelearningmastery.com/gentle-introduction-backpropagation-time/.

[9] Buffoni, B. (2016). Gradient, dérivés directionnelles, différentielle. Notes de cours. Ecole Polytechnique fédérale de Lausanne.

[10] Buscema, M. (1998). Back propagation neural networks. *Substance use and misuse*, vol. 33(2) :p. 233–270.

[11] Butorac, D. (2015). The language of conducting | darko butorac | tedxumontana. video, TEDx Talks, Youtube. Récupéré de : https ://www.youtube.com/watch ?v=xcR1-WhjZys.

[12] Chung, J., Gülçehre, Ç., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, vol. abs/1412.3555.

[13] Daffertshofer, A., Lamoth, C. J., Meijer, O. G., and Beek, P. J. (2004). Pca in studying coordination and variability, a tutorial. *Clinical Biomechanics*, vol. 19(4) :p. 415–428.

[14] deeplearning.ai (2018a). Different types of rnns. video, Coursera, spécialisation DeepLearning, cours : sequence models. Récupéré de : https ://fr.coursera.org/lecture/nlp-sequence-models/different-types-of-rnns-BO8PS.

[15] deeplearning.ai (2018b). Logistic regression. video, Coursera, spécialisation DeepLearning, cours : Neural networks and DeepLearning, Week 2. Récupéré de :.

[16] deeplearning.ai (2018c). Logistic regression cost function. video, Coursera, spécialisation DeepLearning, cours : Neural networks and DeepLearning, Week 2. Récupéré de :.

[17] deeplearning.ai (2018d). Vanishing gradients with rnns. video, Coursera, spécialisation DeepLearning, cours : sequence models. Récupéré de : https ://www.coursera.org/lecture/nlp-sequence-models/vanishing-gradients-with-rnns-PKMRR.

[18] Deloitte (2018). Interview with kevin westcott : 2018 media and entertainment industry outlook. Rapport technique, Deloitte Center for Technology, Media and Telecommunications. Récupéré de : https ://www2.deloitte.com/content/dam/Deloitte/us/Documents/technology-media-telecommunications/us-tmt-2018-media-and-entertainment-industry-outlook.pdf.

[19] Donoghue, O., Harrison, A., Laxton, P., and Jones, R. (2008). Lower limb kinematics of subjects with chronic achilles tendon injury during running. *Res Sports Med*, vol. 16(1) :p. 23–38.

[20] Escabiasana, M., Aguilera, M., Heredia-Jimenezeva, J. M., and Orantes-Gonzalez (2017). *Functional Statistics and Related Fields*, chapter Functional data analysis in kinematics of children going to school, pages chap. 13 p 95–103. Springer, Espagne, A Coruna.

[21] Gagné, C. (2016). Gif-4101 / gif-7005, apprentissage et reconnaissance. Notes de cours : Perceptron Multicouche. Université de Laval.

[22] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http ://www.deeplearningbook.org.

[23] Guo, J. (2013). Backpropagation through time. Rapport technique, Semantic Scholar. Récupéré de : https ://pdfs.semanticscholar.org/c77f/7264096cc9555cd0533c0dc28e909f9977f2.pdf.

[24] harini suresh (9 octobre 2016). Vanishing gradients and lstms. *harinisuresh.com*. Récupéré de : http ://harinisuresh.com/2016/10/09/lstms/.

[25] Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6(2) :p. 107–116.

[26] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, vol. 9(8) :p. 1735–1780.

[27] HUBinternational (27 novembre 2017). Entertainement industry outlook 2018 : Who is really driving the future of the industry ? *hubinternational.com*, HUB Insights. Récupéré de : https ://www.hubinternational.com/blog/2017/11/entertainment-industry-trends-in-2018/.

[28] Karipidou, K. (2015). Modelling the body language of a musical conductor using gaussian process latent variable models. Mémoire, KTH Computer Science and Communication, Stockholm.

[29] Karpathy, A. (21 mai 2015). The unreasonable effectiveness of recurrent neural networks. *blog d'Andrej Karpathy*. Récupéré de : http ://karpathy.github.io/2015/05/21/rnn-effectiveness/.

[30] Lindgren, M. and Sivertsson, A. (2016). Visualizing the body language of a musical conductor using gaussian process latent variable models. Thèse de baccalauréat, KTH Computer Science and Communication, Stockholm.

[31] Lynn, S. K., Noffal, G. J., Wu, W. F., and A.Vandervoort, A. (2012). Using principal components analysis to determine differences in 3d loading patterns between beginner and collegiate level golfers. *International Journal of Golf Science*, vol. 1(1) :p. 25–41.

[32] Mahanta, J. (28 avril 2017). Keep it simple ! how to understand gradient descent algorithm. *kdnuggets.com*, Section Tutorials, Overviews. Récupéré de : https ://www.kdnuggets.com/2017/04/simple-understand-gradient-descent-algorithm.html.

[33] Mazur, M. (17 mars 2015). A step by step backpropagation example. *blog de Matt Mazur*. Récupéré de : https ://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/.

[34] Mezzofiore, G. (15 septembre 2017). Watch this robot conduct andrea bocelli and an entire symphonic orchestra in italy. *Mashable*, Section Culture. Récupéré de : https ://mashable.com/2017/09/15/robot-orchestra-conductor-yumi-abb-italy/ ?europe=true#RgFzPFwJtkqg.

[35] Mikolov, T., Karafiát, M., Burget, L., Černocký, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *11ème conférence annuelle de l'International Speech Communication Association (INTERSPEECH)*, Makuhari, Chiba, Japan.

[36] Nancy, D. (23 mars 2015). La science au service des chefs d'orchestre. *UDEMnouvelles*, Section Sciences. Récupéré de : http ://nouvelles.umontreal.ca/article/2015/03/23/la-science-au-service-des-chefs-dorchestre/.

[37] Nielsen, M. A. (2015). *Neural Networks and Deep Learning*. Determination Press.

[38] Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *30ème conférence internationale sur le Machine Learning*, Atlanta, GA, USA.

[39] Quantzig (2018). Popular entertainement industry trends to tune into in 2018. *quantzig.com*, blog. https ://www.quantzig.com/blog/entertainment-industry-trends.

[40] Ramsay, J. and Silverman, B. (2005). *Funtional Data Analysis*. Springer Series in Statistics. Springer, New York, second edition edition.

[41] Robertson, D. (2010). Tedxstlouis - david robertson - the art of conducting. video, TEDx Talks, Youtube. Récupéré de : https ://www.youtube.com/watch ?v=xKak7V6al1I.

[42] Rudolf, M. (1994). *The grammar of Conducting : a comprehensive guide to baton technique and interpretation*. Schirmer Books, 3rd edition edition.

[43] Ryan, W., Harrison, A., and Hayes, K. (2010). Functional data analysis of knee joint kinematics in the vertical jump. *Sports Biomechanics*, vol. 5(1) :p. 121–137.

[44] Sadeghi, H. (2003). Local or global asymmetry in gait of people without impairments. *Gait Posture*, vol. 17(3) :p. 197–204.

[45] Sadeghi, H., Allard, P., and Duhaime, M. (1997). Functional gait asymmetry in able-bodied subjects. *Human Movement Science*, vol. 16(2-3) :p. 243–258.

[46] Selvin, S., R, V., Gopalakrishnan, E. A., Menon, V., and Kp, S. (2017). Stock price prediction using lstm, rnn and cnn-sliding window model. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Udupi, Inde.

[47] Spruyt, V. (5 mars 2014). What are eigenvectors and eigenvalues ? *visiondummy.com*, Section Linear algebra. Récupéré de : http ://www.visiondummy.com/2014/03/eigenvalues-eigenvectors/.

[48] Stewart, E. (14 mai 2008). Robot conductor debuts in detroit. *The Guardian*, Section Technology. Récupéré de : https ://www.theguardian.com/technology/2008/may/14/usa.

[49] Subbaraman, N. (11 janvier 2016). Boston symphony orchestra hopes ipads will bring kids back to classical. *Beta Boston - The Boston Globe*, Section art. Récupéré de : http ://www.betaboston.com/news/2016/01/11/ipads-at-the-orchestra-boston-symphonys-casual-friday-series-okays-tech-jeans/.

[50] Tallec, C. and Ollivier, Y. (2018). Unbiasing truncated backpropagation through time. In *ICLR 2018 Conference*, Vancouver, Canada.

[51] Troje, N. F. (2002). Decomposing biological motion, a framework for analysis and synthesis of human gait patterns. *Jounal of Vision*, vol. 2(5) :p. 371–387.

[52] Yan, S. (13 mars 2016). Understanding lstm and its diagrams. *medium.com*, Section ML Review. Récupéré de : https ://medium.com/mlreview/understanding-lstm-and-its-diagrams-37e2f46f1714.