

# THÈSE DE L'UNIVERSITÉ DE LYON

Délivrée par

UNIVERSITÉ CLAUDE BERNARD LYON 1

DIPLOÔME DE DOCTORAT

*Ma spécialité*

## DÉCODAGE DES INTENTIONS ET DES REPRÉSENTATIONS MOTRICES CHEZ L'HOMME : ANALYSE MULTI-ÉCHELLE ET APPLICATION AUX INTERFACES CERVEAU-MACHINE

par

Etienne Combrisson

Thèse soutenue le 09/2016 devant le jury composé de :

M <sup>me</sup>	ERIKA RATÉ	Université à la Menthe	(Rapporteur)
M.	JACQUES OUILLE	Université à la Fraise	(Rapporteur)
M.	HENRI ZOTO	Laboratoire laborieux	(Rapporteur)
M.	JEAN FILE	Indienne	(Directeur)
	etc.		



*À mes parents, Isabelle et Didier Prévot,  
Merci*

**Titre** Décodage des intentions et des représentations motrices chez l'homme : analyse multi-échelle et application aux interfaces cerveau-machine

**Résumé** Le résumé en français ( $\approx$  1000 caractères)

**Mots-clés** Les mots-clés en français

---

**Title** Le titre en anglais

**Abstract** Le résumé en anglais ( $\approx$  1000 caractères)

**Keywords** Les mots-clés en anglais

# REMERCIEMENTS

J' voudrais tout d'abord exprimer mes plus profonds remerciements à...  
AHÂÂÂH!

Je conclurai en remerciant de tout cœur (l'être aimé).

Montréal, le 15 mai 2017.

# TABLE DES MATIÈRES

TABLE DES MATIÈRES	vi
LISTE DES FIGURES	viii
NOTATIONS	1
<b>I Introduction</b>	<b>3</b>
<b>1 DÉCODAGE DE L'ACTIVITÉ CÉRÉBRALE</b>	<b>5</b>
<b>1.1 LES INTERFACE CERVEAU-MACHINE . . . . .</b>	<b>5</b>
1.1.1 Définition et objectifs . . . . .	5
1.1.2 Techniques d'acquisition de l'activité neuronale . . . . .	12
1.1.3 ICM synchrones/asynchrones et invasives/non-invasives . . . . .	18
1.1.4 Signaux physiologiques pour le contrôle d'une ICM . . . . .	19
<b>1.2 Data-mining EN NEUROSCIENCES . . . . .</b>	<b>27</b>
1.2.1 Exploration des données . . . . .	27
1.2.2 Outils de validation . . . . .	27
<b>2 ICM ET NEUROPHYSIOLOGIE</b>	<b>29</b>
<b>2.1 BASES PHYSIOLOGIQUES LIÉES À LA MOTRICITÉ . . . . .</b>	<b>29</b>
2.1.1 Notions sur le cortex . . . . .	29
2.1.2 L'activité rythmique et lien avec la motricité . . . . .	30
<b>2.2 DÉCODAGE DIRECTIONNEL DES MEMBRES SUPÉRIEURS . . . . .</b>	<b>31</b>
2.2.1 Décodage directionnel . . . . .	32
2.2.2 Prédiction continu de la cinétique du mouvement . . . . .	33
<b>3 OBJECTIFS DE LA THÈSE</b>	<b>35</b>
<b>4 MÉTHODOLOGIE</b>	<b>37</b>
<b>4.1 EXTRACTION DES FEATURES . . . . .</b>	<b>37</b>
4.1.1 Pré-requis . . . . .	37
4.1.2 Puissance spectrale . . . . .	39
4.1.3 Phase . . . . .	40
4.1.4 Phase-amplitude coupling . . . . .	40
<b>4.2 APPRENTISSAGE SUPERVISÉ . . . . .</b>	<b>46</b>
4.2.1 Labellisation et apprentissage . . . . .	47
4.2.2 <i>Training, testing</i> et validation-croisée . . . . .	47
4.2.3 Classificateurs . . . . .	49
4.2.4 Évaluation de la performance de décodage . . . . .	51

4.2.5	Seuil de chance et évaluation statistique de la performance de décodage . . . . .	52
4.2.6	Du single au multi-features . . . . .	52
4.2.7	Généralisation temporelle . . . . .	55
<b>5</b>	<b>DÉVELOPPEMENTS INFORMATIQUES</b>	<b>59</b>
5.1	CHOIX DU LANGAGE : PYTHON . . . . .	59
5.2	PAQUETS DÉVELOPPÉS DURANT CETTE THÈSE . . . . .	60
5.2.1	ipywksp . . . . .	60
5.2.2	Brainpipe . . . . .	61
5.2.3	Visbrain . . . . .	64
<b>6</b>	<b>DONNÉES EXPÉRIMENTALES</b>	<b>69</b>
6.1	DONNÉES INTRACRÂNIENNES . . . . .	69
6.1.1	Acquisition . . . . .	69
6.1.2	Avantages et limitations . . . . .	69
6.1.3	Inspection visuelle . . . . .	71
6.1.4	Prétraitements . . . . .	71
6.2	DONNÉES D'ÉTUDE . . . . .	71
6.2.1	Données <i>Center-out</i> . . . . .	72
6.2.2	Autres données . . . . .	74
6.3	DELAYED TASK : PROTOCOLE EXPÉRIMENTAL . . . . .	74
	<b>CONCLUSION GÉNÉRALE</b>	<b>75</b>
<b>A</b>	<b>ANNEXES</b>	<b>76</b>
A.1	CARTES DES INTERFACES CERVEAU-MACHINE ( <a href="#">GRAIMANN ET AL., 2009</a> )	77
A.2	JEUX DE DONNÉES EN LIBRE ACCÈS ( <i>BCI competition</i> ) . . . . .	78
A.3	COMPARATIF DE MÉTHODES PAC ( <a href="#">TORT ET AL., 2010</a> ) . . . . .	79
A.4	PIPELINE STANDARD DE CLASSIFICATION . . . . .	80
A.5	COMPARATIF DE CLASSIFIEURS ( <a href="#">PEDREGOSA ET AL., 2011</a> ) . . . . .	82
A.6	EXEMPLE DE SCHÉMA D'IMPLANTATION . . . . .	84
A.7	DOCUMENTATION DE BRAINPIPE . . . . .	85
	<b>BIBLIOGRAPHIE</b>	<b>155</b>

# LISTE DES FIGURES

1.1	Schéma d'une Interface Cerveau-Machine (Pfurtscheller et al., 2008) . . . . .	9
1.2	Paralysie causée par la SLA et accident vasculaire cérébrale (Kübler et al., 2001) . . . . .	10
1.3	BCI competition et visibilité Tangermann et al. (2012) . . . . .	12
1.4	ICM utilisant des micro-électrodes (Hochberg et al., 2006, 2012) . . . . .	14
1.5	Micro et macro-électrodes pour l'Électrocorticographie (Schalk and Leuthardt, 2011, Yanagisawa et al., 2012a) . . . . .	15
1.6	Production des champs électriques et magnétiques. Recueil EEG et MEG, limitations et complémentarité (Sato et al., 1991) . . . . .	15
1.7	Exemple de casque EEG . . . . .	17
1.8	Méthodes d'acquisition de l'activité cérébrale (Waldert et al., 2009) . . . . .	18
1.9	Spectre de puissance d'un signal EEG contenant des SSVEP (Lalor et al., 2005) . . . . .	20
1.10	ERP et onde P300 (Kübler et al., 2001) . . . . .	21
1.11	Comparaison des aires actives lors d'un mouvement imaginé ou exécuté (Hanakawa et al., 2008) . . . . .	23
1.12	Exemple d'apprentissage pour contrôler les Slow Cortical Potential (Kübler et al., 2001) . . . . .	24
1.13	Utilisation des ERD et du $\mu$ pour contrôler une ICM (Kübler et al., 2001) . . . . .	25
1.14	Décomposition d'un signal en phase et amplitude . . . . .	26
2.1	Présentation du cortex et des aires destinées au contrôle moteur . . . . .	30
2.2	Exemple d'activités rythmiques dans les différentes bandes de fréquences pour un essai unique issue de l'activité sEEG . . . . .	31
2.3	Directional tuning et décodage directionnel . . . . .	32
2.4	Décodage continu de la cinétiqe d'un mouvement 2D . . . . .	33
4.1	Évaluation statistique à base de permutations . . . . .	39
4.2	Exemple de représentation temps-fréquence de puissance normalisées z-score (Ossandon et al., 2011) . . . . .	40
4.3	Densité de probabilité d'une distribution d'amplitudes en fonction de tranches de phases . . . . .	42
4.4	(A) Exemple de cartes temps-fréquence phase locked sur le $\beta$ , (B) Exemple de comodulogramme . . . . .	44
4.5	Exemple de phase préférentielle . . . . .	46
4.6	Labellisation des données . . . . .	47
4.7	Exemple d'une cross validation 3-folds . . . . .	48
4.8	Principe du Linear Discriminant Analysis (Lotte et al., 2007, Naseer and Hong, 2015) . . . . .	49

4.9	Principe du Support Vector Machine ( <a href="#">Lotte et al., 2007</a> , <a href="#">Naseer and Hong, 2015</a> ) . . . . .	50
4.10	Principe du k-Nearest Neighbor ( <a href="#">Weinberger et al., 2005</a> ) . . . . .	50
4.11	Entraînement puis test d'un classifieur linéaire . . . . .	51
4.12	Calcul de l'acuité de décodage . . . . .	51
4.13	Exemple d'une <i>Forward feature selection</i> appliquée sur six features . . . . .	54
4.14	Exemple d'une <i>Backward feature elimination</i> appliquée sur six features . . . . .	55
4.15	Exemple de décodage temporel ( <a href="#">Waldert et al., 2008</a> ). Ici, l'auteur décode 4-directions de mouvements de la main dans le temps. A chaque instant, un classifieur est créé, entraîné puis testé à ce même instant. . . . .	56
4.16	Exemple de généralisation temporelle ( <a href="#">King and Dehaene, 2014</a> ) . . . . .	57
5.1	<i>ipywksp</i> : Exemple de workspace pour <i>Jupyter</i> . . . . .	60
5.2	Exemple de calcul PAC avec brainpipe . . . . .	62
5.3	Exemple de plot d'un signal et de sa déviation . . . . .	64
5.4	Exemples des principales fonctionnalités de <b>Brain</b> . . . . .	65
5.5	Exemples des principales fonctionnalités de <b>Sleep</b> . . . . .	66
5.6	Exemples des principales fonctionnalités de <b>Ndviz</b> . . . . .	67
5.7	Exemple de mise en page avec le module <b>Figure</b> . . . . .	68
6.1	Comparatif de résolution spatiale et temporelle pour différentes techniques d'imagerie ( <a href="#">Lachaux et al., 2003</a> ) . . . . .	71
6.2	Détails cliniques des sujets ayant participé à la tâche <i>Center-out</i> . . . . .	72
6.3	Implantation intracrâniale et couverture corticale de six sujets épileptiques ayant passés la tâche <i>Center-out</i> . . . . .	73
6.4	Descriptif de la tâche <i>Center-out</i> . . . . .	73
A.1	Cartes des Interfaces Cerveau-Machine ( <a href="#">Graimann et al., 2009</a> ) . . . . .	77
A.2	Jeux de données en libre accès ( <i>BCI competition</i> ) . . . . .	78
A.3	Comparatif de méthodes PAC ( <a href="#">Tort et al., 2010</a> ) . . . . .	79
A.4	Pipeline standard de classification . . . . .	80
A.5	Comparatif de classifieurs ( <a href="#">Pedregosa et al., 2011</a> ) . . . . .	82
A.6	Exemple de schéma d'implantation . . . . .	84



# NOTATIONS

## Général

ICM	Interface Cerveau-Machine
ICO	Interface Cerveau-Ordinateur
BCI	Brain Computer Interface
BMI	Brain Machine Interface

## Enregistrements

EEG	Électroencéphalographie
MEG	Magnétoencéphalographie
SUA	Single Unit Activity
MUA	Multi Unit Activity
SEEG	Stéréoélectroencéphalographie
ECoG	Électrocorticographie
IRMf	Imagerie par Résonance Magnétique fonctionnelle
fNIRS	Functional near-infrared spectroscopy

## Marqueurs/Motifs/Attributs/Pattern/Features

SCPS	Slow Cortical Potential
SSEP	Steady-State Evoked Potentials
ERS	Event-Related Synchronization
ERD	Event-Related Desynchronization
PAC	Phase-Amplitude Coupling

## Classificateurs

LDA	Linear Discriminant Analysis
SVM	Support Vector Machine
RF	Random Forest
KNN	k-Nearest Neighbor
NB	Naive Bayes



**Première partie**

**Introduction**



# DÉCODAGE DE L'ACTIVITÉ CÉRÉBRALE

## 1.1 LES INTERFACE CERVEAU-MACHINE

### 1.1.1 Définition et objectifs

Cette première section a pour but d'introduire et de définir le concept d'Interface Cerveau-Machine . Nous verrons dans quel contexte elles sont apparues, les personnes à qui elles sont destinées ainsi que les principaux éléments qui les composent.

Point de vue lexicale, on utilisera indifféremment *Interface Cerveau-Machine (ICM)*, *Interface Cerveau-Ordinateur (ICO)* ou les termes anglais correspondant à savoir *Brain Computer Interface (BCI)* et *Brain Machine Interface (BMI)*.

#### 1.1.1.1 Contexte d'apparition des ICM

En 1964, Dr. Grey Walter connecte des électrodes directement dans le cortex moteur d'un patient et lui demande de presser un bouton pour faire avancer un rétro-projecteur. En même temps, il enregistre l'activité neuronale de telle sorte que elle aussi, puisse le faire avancer. Là où l'expérience devient remarquable, c'est que le rétro-projecteur avance avant que le patient ne presse le bouton! Tout l'appareil musculaire du sujet est court-circuité et le contrôle se fait sans mouvement. Contrôler par la *pensée*, un sujet de science fiction qui devient une réalité. Cette anecdote décrite par [Graimann et al. \(2009\)](#), permet de placer la naissance de la possibilité d'une Interface Cerveau-Machine (ICM) dans l'histoire. C'est le point d'entrée qui a ensuite conduit une grande diversité de chercheurs à se passionner pour ce sujet. Le terme *Brain Computer Interface* fait son apparition, au début des années 70, dans les publications de Jacques Vidal ([Vidal, 1973, 1977](#)) où il était question de contrôler un curseur sur un écran.

La progression des ICM et de l'intérêt de la communauté scientifique à véritablement commencé dans les années 2000. Trois facteurs sont à l'origine ([Wolpaw et al., 2002](#)) :

1. Une amélioration des connaissances des processus neuro-physiologiques et des techniques d'imagerie.
2. L'arrivée d'ordinateur bon marché et l'amélioration constante de leur performances et des composants électroniques (processeurs, mémoire vive, logiciel...)
3. Une prise de conscience sociétale des besoins de personnes souffrant de problèmes neuro-musculaires.

A noter que, à l'heure actuelle, l'arrivée de cartes graphiques bon marché est entrain de révolutionner l'approche computationnelle des ICM permettant des calculs plus lourds en moins de temps (notamment pour le *Deep Learning*). Outre la continue amélioration des composants et de leur miniaturisation, la prochaine révolution concernera certainement les ordinateurs quantiques, déjà en phase de test dans les domaines de la génétique et de la chimie.

#### **1.1.1.2 Interactions naturelles avec l'environnement**

Pour interagir avec son environnement, l'individu se sert des voies de communications naturelles, c'est-à-dire via son système nerveux et musculaire. Le processus de communication débute par une intention qui active certaines régions dans le cerveau. Il en résulte un signal cérébral qui est ensuite envoyé par le système nerveux périphérique en directions des muscles ([Besserve, 2007](#)). C'est ce processus simplifié qui permet à une personne d'interagir avec ce qui l'entoure.

#### **1.1.1.3 Un canal de communication alternatif**

Il existe plusieurs maladies ou accidents qui entraînent une dégénérescence des performances motrices. Parmi elles, on peut par exemple citer la Sclérose Latérale Amyotrophique (ou SLA), les accidents vasculaires cérébraux, certaines formes de sclérose, les lésions de la moelle épinière... Toutes ont en commun la possibilité de problème moteur. Dans ce contexte, [Wolpaw et al. \(2002\)](#) introduit trois options pour restaurer ces fonctions :

1. Augmenter les capacités des facultés motrices restantes. Autrement dit, donner un sens nouveau aux mouvements que l'individu est toujours en capacité de faire. A titre d'exemple, le guitariste virtuose Jason Becker, reconnu pour sa vélocité et dont tout le monde s'entendait sur son incroyable talent, fut un jour frappé par la SLA le conduisant au fur et à mesure à l'immobilité totale. Il convenu alors d'un langage basé sur les mouvements oculaires et, avec la complicité de son père, continua de composer.
2. Contourner la lésion. L'auteur donne à titre d'exemple, une lésion de la moelle épinière que l'on peut contourner en utilisant l'activité des muscles situés au dessus de la lésion pour stimuler les muscles paralysés.
3. Enfin, la dernière façon de restaurer des fonctions motrices qui prend tout son sens lorsque les deux précédentes ne sont pas possibles, c'est d'établir un nouveau canal de communication directe entre le cerveau et un ordinateur, et ce, indépendamment de l'activité musculaire. D'où le nom, *Interface Cerveau-Machine*.

Une ICM est un autre système de communication où les voies naturelles sont cout-circuitées. Au lieu de passer par le système nerveux puis musculaire, le signal cérébral est directement intercepté au niveau du cerveau et va ensuite être transformé en commandes. Une ICM est donc un système permettant de traduire une activité neuronale en commande extérieure. Le terme *traduire* est à prendre au sens linguistique c'est-à-dire que les signaux cérébraux forment un langage, composé de règles, de motifs ou *pattern*, que l'on va essayé de décoder (via un ordinateur) pour les transformer en opérations. D'où le terme "Interface Cerveau-Machine".

[Pfurtscheller et al. \(2008\)](#) et ([Graimann et al., 2009](#)) introduisent quatre éléments qui composent une ICM :

1. Enregistrer l'activité directement depuis le cerveau. Cet enregistrement pourra être invasif ou non-invasif (cf. 1.1.2)
2. Générer un retour ou *feedback* pour l'utilisateur
3. L'enregistrement et le *feedback* doivent être en temps réel
4. Enfin, l'interface doit être contrôlable par l'utilisateur, de manière active, via un ensemble d'intentions.

A titre d'exemple et pour illustrer ce dernier point, un utilisateur pourrait par exemple décider de bouger un curseur de souris sur un écran en imaginant des mouvements soit de la main gauche soit de la main droite.

#### 1.1.1.4 Principales composantes d'une ICM

Même si chaque Interface Cerveau-Machine se destine à une utilisation particulière et contient des traitements qui lui sont propres, on peut globalement dire qu'une ICM s'articule autour de cinq grandes étapes ordonnées (Pfurtscheller et al., 2008, Graimann et al., 2009) :

1. L'acquisition de l'activité neuronale
2. Les pré-traitements
3. L'extraction de marqueurs
4. La classification
5. La transformation en commande

Ces étapes sont interdépendantes c'est-à-dire que chacune s'appuie sur les résultats de l'étape précédente. De plus, cette cascade de stades doit se faire en temps réel pour que l'expérience utilisateur soit la plus fluide possible et qu'elle reflète fidèlement ce qui se déroule à chaque instant.

**1.1.1.4.1 Acquisition de l'activité neuronale** L'acquisition de l'activité neuronale constitue le point d'entrée d'une ICM. Les différentes techniques pour enregistrer sont plus ou moins accessibles (certaines sont portatives, d'autres nécessitent un appareil très lourd...). C'est, entre autre, l'accessibilité qui va influer sur le nombre de sujets d'une étude. Autre point très important que nous décrirons plus bas, la qualité du signal (ou le rapport signal sur bruit (RSB)) qui aura un impact immédiat sur les performances et sur les limitations d'une ICM. Enfin, on parlera d'enregistrements *invasifs* (cf. 1.1.2.1) quand ceux-ci nécessiteront une implantation chirurgicale d'électrodes et *non-invasif* (cf. 1.1.2.2) pour les techniques d'acquisition se faisant en dehors de la boîte crânienne.

**1.1.1.4.2 Pré-traitements** Les pré-traitements regroupent un ensemble de techniques destinées à nettoyer le signal pour faire ressortir, autant que possible, le signal utile par rapport au signal bruité. Parmi ces traitements, on peut citer le nettoyage d'artefacts oculaires, cardiaques ou musculaires, le référencement (essentiellement pour l'EEG), la bipolarisation (pour la SEEG), le filtrage pour supprimer certaines composantes spectrales... Ces pré-traitements sont propres à chaque technique d'enregistrement. Une description plus détaillée des pré-traitements appliqués dans le cadre de données SEEG est proposée dans la section 6.1.4.

**1.1.1.4.3 Extractions de marqueurs** En imaginant que l'on répète dix fois le même mouvement, il y aura dans l'activité neuronale une partie similaire permettant de reproduire chacune de ces répétitions. Le signal entier sera très probablement différent à chaque fois, mais, à l'intérieur de ce signal, on pourra trouver un "sous-signal" dont le contenu sera similaire à chacune de ces répétitions.

C'est le but de cette étape d'extraction de marqueurs, la recherche de ce "sous-signal". Une fois l'activité cérébrale nettoyée, on va chercher à extraire des marqueurs qui matérialisent l'état instantané d'un sujet. Par exemple, si celui-ci bouge le bras vers la gauche ou vers la droite, on doit pouvoir extraire une information de ce signal qui encode chacun de ces états. En pratique, on peut distinguer deux types de marqueurs : les marqueurs *locaux*, qui reflètent l'activité d'une "petite" population de neurones prises localement (cf. 1.1.4), et les marqueurs d'*interaction* qui quantifie un degrés de couplage à distance entre deux régions du cerveau.

Cette étape est véritablement au cœur du bon fonctionnement d'une ICM puisque, en fonction de la qualité de ce marqueur, la machine sera plus moins enclue à reconnaître les différentes commandes d'un sujet.

En lieu et place du terme *marqueur*, on pourra utiliser indifféremment *motif*, *pattern*, *feature* ou *attribut*.

**1.1.1.4.4 Classification** En reprenant l'exemple de la section précédente, supposons que l'on dispose d'un marqueur et on cherche à savoir si celui-ci appartient à une des deux classes de mouvement de bras, vers la gauche ou vers la droite. C'est le problème de classification. A partir d'un motif dont on ignore la provenance, on utilise un algorithme permettant de reconnaître la classe dont est issue ce pattern. Parmi les algorithmes les plus fréquemment rencontrés, on peut citer le Linear Discriminant Analysis , le Support Vector Machine ou le k-Nearest Neighbor .

En pratique, cette reconnaissance de classes est mise en place en deux étapes :

1. L'entraînement ( ou *training* ) : durant une certaine période, on va apprendre à une machine à reconnaître des événements. Pour cela, on utilise des données dont on connaît la provenance (c'est ce que l'on appelle la *labellisation*)
2. Le test ( ou *testing* ) : une fois la machine entraînée à partir d'une série de marqueurs labélisés, on teste l'algorithme avec des nouvelles données pour évaluer l'acuité de la machine à identifier ces événements.

Quelque soit l'algorithme de classification, celui-ci doit s'adapter à chaque utilisateur suivant trois niveaux ([Wolpaw et al., 2002](#)) :

1. En premier lieu, et de manière assez évidente, il doit pouvoir s'adapter au marqueur du sujet
2. Ensuite, l'algorithme doit s'adapter en temps-réel aux variations spontanées. En effet, l'expérience utilisateur va varier en fonction d'un certain nombre de paramètres comme le moment de la journée, la fatigue, la maladie, le taux d'hormones, la motivation/concentration/frustration... ([Curran, 2003](#))
3. Enfin, il doit permettre de prendre en compte l'adaptation du sujet. Durant l'expérience, l'utilisateur module son activité et fournit des efforts pour s'adapter au fonctionnement de la machine. En contrepartie, le software doit prendre en compte cette amélioration en fournissant des performances accrues.

Cette étape de classification, décrite plus largement dans la section 4.2, est très fortement dépendante de la qualité des marqueurs extraits en amont. Autrement

dit, plus ces *features* reflètent fidèlement un état, plus le travail de la machine à identifier ces événements sera facilité.

**1.1.1.4.5 Transformation en commande** Dernière étape du processus d'une ICM, lorsque l'algorithme de classification pense avoir identifié le type de marqueurs, on attribue une commande physique. Par exemple, si la machine reconnaît un mouvement de bras vers la droite, on pourrait attribuer une commande où l'on déplace le curseur d'une souris sur un écran dans la même direction. Ainsi, on procure à l'utilisateur un *feedback* sur la transformation que la machine a réussit à faire à partir de l'activité de son cerveau.

Cette transformation en commande est ensuite physiquement appliquée à un système externe. L'efficacité de l'ensemble de l'ICM est donc évaluée en fonction de son accuité à restituer, avec plus ou moins de fidélité, la commande désirée par l'utilisateur.

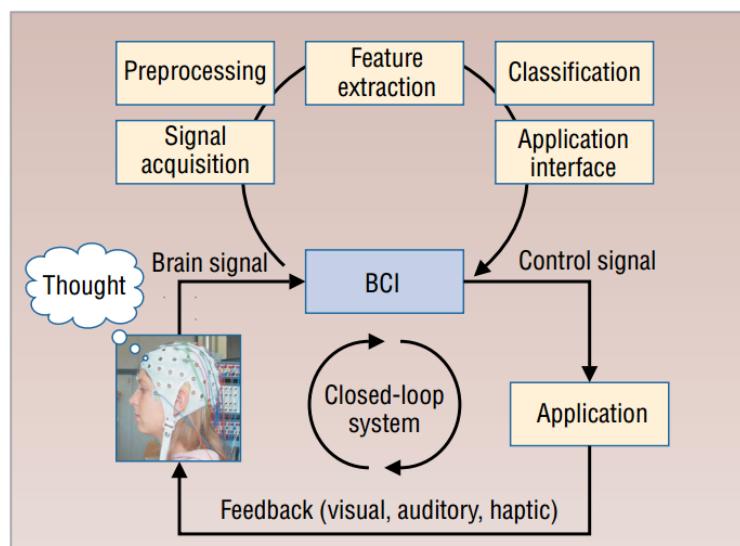


FIGURE 1.1 – L'activité neuronale est enregistrée (*Signal acquisition*) puis nettoyée (*Preprocessing*).

Ensuite, on extrait des motifs ou patterns qui caractérisent la commande que souhaite envoyer le sujet (*Feature extraction*). Enfin, la machine tente de reconnaître ces motifs (*Classification*) et de les transformer en commande (*Application interface*). Cette boucle se termine en donnant un feedback à l'utilisateur sur l'état actuel de la machine. (Pfurtscheller et al., 2008)

Les ICM partagent globalement ces cinq étapes mais se différencient donc le type d'enregistrement de l'activité neuronale, par les pré-traitements associés, par le type de marqueurs étudié, par l'algorithme de classification choisi et surtout, par l'application concrète de cette Interface Cerveau-Machine .

### 1.1.1.5 Applications des ICM : cliniques et non-cliniques

La réalisation concrète des ICM s'est articulée autour des applications cliniques, c'est-à-dire destinées à essayer d'améliorer les conditions de vie de certaines personnes puis, dans un second temps, pour des applications ludiques et tout public.

**1.1.1.5.1 Applications cliniques** A l'heure actuelle, il existe de nombreuses maladies soit dont on ignore l'origine, soit qui sont pour le moment incurables. Parmi ces pathologies, certaines évoluent en enfermant les patients dans des conditions de vies

difficiles. C'est dans ce contexte clinique qu'apparaissent de nombreuses ICM qui ne sont donc pas des traitements, mais bien des solutions *temporaires* pour aider certaines personnes à mieux vivre avec leur handicap.

**Pathologies et ICM :** Prélever directement l'activité neuronale et donc, *bypasser* les voies naturelles, permet de s'affranchir d'éventuelles limitations physiologiques. C'est pourquoi les ICM représentent un enjeu majeur pour la réhabilitation motrice ou handicap moteur ou encore pour la communication palliative. Les applications concrètes des Interface Cerveau-Machine visent donc les personnes disposant de leurs capacités cognitives mais qui sont privées de facultés motrices.

Un accident vasculaire cérébral (AVC) peut engendrer un état d'enfermement (*locked-in state (LIS)*). Les personnes dans cet état sont pleinement conscientes de leur corps, de l'environnement, ils peuvent ressentir les sensations de toucher et de douleur mais n'ont plus de facultés motrices, hormis peut-être, les mouvements de paupières ou des yeux. Un autre exemple est celui de la sclérose latérale amyotrophique (ou SLA) qui est une dégénérescence des neurones moteur (motoneurones). Progressivement, les personnes atteintes de SLA perdent l'usage des bras, des jambes, de la parole des muscles faciaux et enfin de la déglutition mais la conscience et les facultés cognitives demeurent intactes. L'évolution de la maladie amène à deux possibilités ([Chaudhary et al., 2015](#)) : accepter une dépendance totale (respiration artificielle et nutrition) ou un décès par insuffisance respiratoire. Dans le premier cas, la maladie entraîne progressivement les patients en LIS ne leur laissant qu'un minimum de fonctions motrices. Lorsque le patient perd tout contrôle musculaire, ce qui finit souvent par les muscles des yeux, il rentre dans un état d'enfermement complet (*completely locked-in state (CLIS)*).

SLA et *locked-in syndrome* ne sont que deux exemples expliquant l'intérêt social du développement des ICM. Redonner un peu de contrôle ou établir un canal de communication avec les familles des patients expliquent l'engouement qui existe depuis maintenant plus de 40 ans pour les ICM.

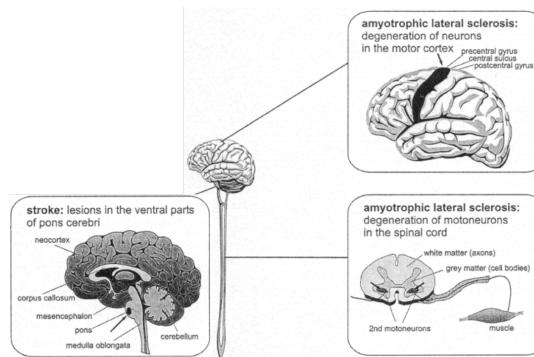


FIGURE 1.2 – Paralysie causée par la SLA et accident vasculaire cérébral ([Kübler et al., 2001](#))

**Exemples d'applications :** les applications cliniques peuvent être divisées en deux grandes familles : les ICM destinées à rétablir un lien de communication (*communication palliative*) et les ICM pour redonner de la mobilité.

**Communication palliative :** ces ICM présentent en général des lettres ou groupe de lettres qui doivent pouvoir être sélectionnés volontairement par l'utilisateur. Le *P300-Speller* ([Farwell and Donchin, 1988](#), [Donchin et al., 2000](#)), utilisant l'onde P300, est sans aucun doute l'interface la plus connue et la plus exploitée à ce jour pour

la communication palliative. Le *Hex-o-spell* (Blankertz et al., 2007) est une autre interface contrôlable par l'utilisateur via l'imagerie motrice.

**Mobilité, robotique et prothèse :** ces systèmes sont destinés à aider, améliorer ou remplacer des facultés motrices lésées. Ces applications dont la mobilité est au centre peuvent afficher différents degrés de complexité :

- Curseur 1D, 2D et 3D : certainement la première étape, le but ici est de permettre au sujet de pouvoir contrôler le curseur sur un moniteur. Ce curseur pourra par exemple servir de bouton *yes/no* ou pour le déplacement (Wolppaw et al., 1991, Wu et al., 2003, Trejo et al., 2006, Kayagil et al., 2009, Kim et al., 2011, Vadera et al., 2013) ou tout simplement pour de la navigation comme le *web-browsing* (Mugler et al., 2010). Ces catégories de déplacement de curseur diffèrent par le nombre de degrés de liberté qu'elles offrent.
- Contrôle d'un fauteuil roulant : comme le nom l'indique, l'idée ici est de permettre à un utilisateur de contrôler son fauteuil roulant via son activité neuronale seulement. En effet, si de nombreuses personnes handicapées peuvent encore se servir de leur membres supérieurs, d'autres sont complètement dépendantes. Ce type d'applications permettra donc à terme de redonner une liberté de mouvement à ces personnes (Tanaka et al., 2005, Leeb et al., 2007, Galán et al., 2008a, Philips et al., 2007, Vanacker et al., 2007, Pires et al., 2008, Rebsamen, 2009, Lin et al., 2010, Diez et al., 2013).
- Prothèse : enfin, dernière application liée à la mobilité, le contrôle de prothèse est un large défi. A titre d'exemple, le contrôle d'un bras robotisé doit permettre le contrôle de celui-ci dans l'espace ainsi que des mouvements de main, de coude... C'est un problème à haute dimensionnalité et donc complexe. Toutefois, modulo un certain degrés de réussite, certaine équipe ont proposé de tel systèmes (Fetz and others, 1999, Hochberg et al., 2012, Yanagisawa et al., 2012a, Sunny et al., 2016)

#### 1.1.1.5.2 Applications non-cliniques et récréatives

Les ICM ont également été exploitées à d'autres fins que des applications cliniques :

**Jeux :** les ICM dédiées aux jeux utilisent l'activité neuronale pour contrôler un vaisseau spatial, un objet ou un personnage dans un environnement virtuel (Lalor et al., 2004, Nijholt, 2008, Oude Bos and Reuderink, 2008, Coyle et al., 2011). Outre le caractère récréatif de ce type de dispositif, les ICM basées sur le jeux pourraient très bien servir pour entraîner la machine d'une manière plus ludique et moins fatigante.

**Art :** pour finir, les ICM peuvent avoir des applications artistiques comme pour la composition et la pratique musicale (Miranda et al., 2003, Miranda, 2006, Hamadicharef et al., 2010) ou pour peindre (Münßinger et al., 2010, Zickler et al., 2013)

#### 1.1.1.6 BCI competition et open-data

Les *BCI competitions* sont proposées par l'équipe du *Berlin Brain-Computer Interface* (BBCI). L'idée de ces compétitions est, sur un même jeu de données découpées en *training* et *testing* (cf. 4.2.2) mettre les équipes de recherche en ICM en compétitions puis élire celle qui arrivera au meilleur décodage. A l'heure où cette thèse est écrite, quatre de ces compétitions ont eu lieu et chacune est associée à un article

de présentation (*BCI competition I* : [Sajda et al. \(2003\)](#), *II* : [Blankertz et al. \(2004\)](#), *III* : [Blankertz et al. \(2006\)](#), *IV* : [Tangermann et al. \(2012\)](#)). La liste des *datasets* en accès libres est disponible en annexe (cf. A.2. Remarque : il semble que les données de la première compétition ne soient plus accessibles, c'est la raison pour laquelle elles n'apparaissent pas dans le tableau)

Ces compétitions présentent de nombreux avantages :

1. Visibilité des équipes : ces compétitions sont de plus en plus "virales" et sont relayées par la suite par de nombreux articles (voir figure ci-dessous).
2. Permet aux équipes de disposant pas de systèmes d'acquisition de profiter de jeux de données pour développer des méthodes. C'est également un moyen d'initier les étudiants à un ensemble de techniques.
3. Autre avantage, tout le monde se retrouve sur un pied d'égalité en travaillant sur les mêmes données. Ce qui représente par la suite, une méthode rigoureuse pour sélectionner les meilleurs algorithmes.

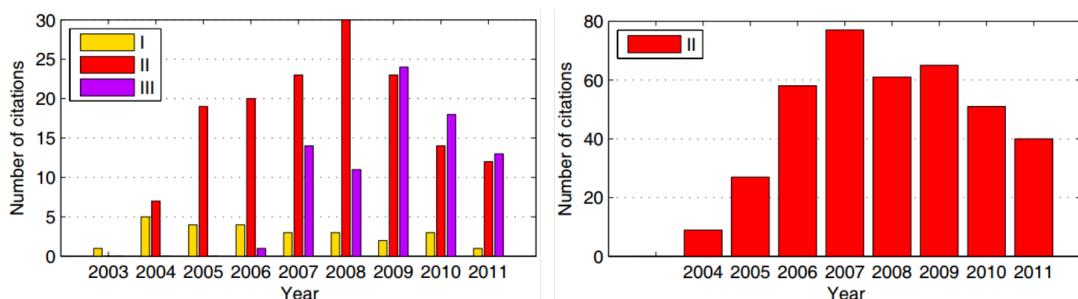


FIGURE 1.3 – (A gauche) Évolution du nombre de citations par an des articles de présentation pour les compétitions I, II et III, (A droite) Évolution du nombre de citations par an pour l'équipe ayant gagné la deuxième compétitions ([Tangermann et al., 2012](#))

### Conclusion sur la présentation des ICM

Les personnes atteintes de problèmes moteurs sont certainement les premiers bénéficiaires de l'avancée des Interface Cerveau-Machine . Ces systèmes leur permettraient de rétablir un canal de communication avec leur environnement. Pour cela, les ICM partent de l'enregistrement de l'activité de leur cerveau puis, recherche dans cette activité des motifs relatant un état. Si cet état est *compris* par la machine, il sera ensuite transformé en commande. De nombreux systèmes se sont développés, principalement autour de la communication palliative ou de la mobilité. L'efficacité de ces systèmes reposent en grande partie sur deux points : l'acuité des algorithmes de classification, pour cette raison, ils peuvent être comparés notamment grâce à des initiatives comme les *BCI competitions*. Le deuxième point concerne l'enregistrement de l'activité neuronale qui va très largement conditionner la qualité du signal et des marqueurs qui en seront extraits.

### 1.1.2 Techniques d'acquisition de l'activité neuronale

Les types d'acquisition peuvent être classés par leur degrés de pénétration dans le corps. Les enregistrements dits *invasifs* vont nécessiter une intervention chirurgicale, donc avec risque potentiel d'infection, mais donnent accès à des signaux de

grande qualité permettant des contrôles complexes comme une prothèse ou un bras robotisé (Hochberg et al., 2012, Taylor, 2002). Les enregistrements *non-invasifs* sont globalement plus faciles à mettre en place car ne nécessitant aucune chirurgie mais la qualité du signal est moindre car l'activité neuronale est enregistrée en dehors de la boîte crânienne, donc filtrée par l'os et la peau.

Au sein de ces deux catégories, on trouvera un ensemble de méthodes d'acquisition qui se différencient par la taille des populations de neurones qu'elles enregistrent ou par le type de signal (électrique, magnétique, mesure indirecte...).

#### 1.1.2.1 Enregistrements invasifs

On parlera d'intracrânien pour les enregistrements à l'intérieur de la boîte crânienne puis d'intracortical pour des électrodes implantées dans le cortex (Engel et al., 2005, Jerbi et al., 2009b).

**1.1.2.1.1 Enregistrements unitaires** *Single Unit Activity (SUA)* et *Multi Unit Activity (MUA)* sont des micro-électrodes déposées directement au contact de neurones et enregistrent des décharges neuronales. SUA et MUA se distinguent par la taille des populations enregistrées. Les décharges sont événements très courts dans le temps, donc pour être en mesure de les capter, les systèmes d'acquisition possèdent des fréquences d'échantillonnage très élevées (de l'ordre de 30khz). Le signal obtenu en filtrant en dessous de 300hz, est appelé *Local Field Potential (LFP)* et représente l'activité électrique d'une assemblée de neurones prise dans un petit volume.

Dans le cadre des ICM, les micro-électrodes peuvent être implantées directement dans le cortex moteur primaire et permettre un contrôle de BCI de la plus haute précision et fidélité. Chez l'animal, des rats ont pu contrôler en temps réel un bras robotisé pour obtenir de l'eau (Chapin et al., 1999) et des singes ont également pu contrôler un bras robotisé ainsi qu'une pince au bout (Velliste et al., 2008). Auparavant, en 2006, Hochberg et al. (2006) démontre qu'un patient tétraplégique implanté avec 96 micro-électrodes, peut contrôler un curseur sur un écran d'ordinateur, d'ouvrir et fermer une main artificielle ainsi que d'effectuer des mouvements rudimentaires à l'aide d'un bras robotisé. En complément, Kim et al. (2011) utilise les micro-électrodes pour contrôler et cliquer à l'aide un pointeur sur un écran 2D. Mais c'est en 2012 que le sujet contrôle complètement le bras robotisé, dans l'espace, lui permettant d'attraper et de boire son café (Hochberg et al., 2012). Même si le mouvement n'est pas aussi fluide et rapide qu'un mouvement réel, ce fût une avancée majeure et une preuve du concept chez l'homme. Collinger et al. (2013) décrivent également le contrôle d'un bras robotisé par un patient implanté avec 96-microélectrodes. Avec des taux de décodage élevés, le sujet réussit à contrôler un bras robotisé à 7-dimensions (3 degrés de translation, 3 degrés de rotation et un degrés de saisie).

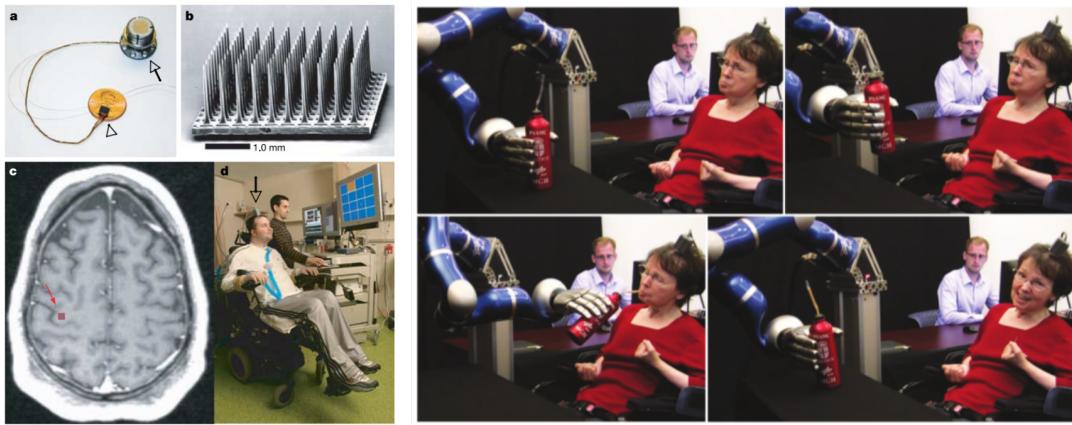


FIGURE 1.4 – (A gauche) Sujet tétraplégique implanté avec des micro-éléctrodes contrôlant un curseur sur un écran (Hochberg et al., 2006), (A droite) (Hochberg et al., 2012)

**1.1.2.1.2 Stéréoélectroencéphalographie (SEEG)** Macro-électrodes enregistrant des populations plus larges que les micro-électrodes. Contrairement à la SUA ou MUA où l'on peut compter le nombre de fois qu'un ou plusieurs neurones déchargent, la SEEG enregistre des potentiels électriques ( $\mu V$ ). La fréquence d'échantillonnage est plus faible que celle des micro-électrodes (de l'ordre de 1kHz) et donne donc accès au signal LFP.

Les données utilisant la Stéréoélectroencéphalographie sont globalement rares dans la littérature. Leur utilisation dans le cadre des ICM n'est pas fréquente mais a quand même été exploré dans le cadre d'ICM dédiée à la communication (Krusienski and Shih, 2011a, Shih and Krusinski, 2012) ou au contrôle de curseur 2D (Vadera et al., 2013)

C'est le type d'enregistrement qui a été le plus exploité durant cette thèse. Une section complète lui est donc accordée (cf. 6.1).

**1.1.2.1.3 Électrocorticographie (ECoG)** L'ECoG est une grille flexible composée d'une matrice d'électrodes. Celle-ci est ensuite déposée à la surface corticale. Parce que cette méthode n'est pas intracortical elle est dite *semi-invasive*. En comparaison avec l'EEG (voir section suivante), l'ECoG présente de nombreux atouts (Schalk and Leuthardt, 2011). Tout d'abord, la résolution spatiale est meilleure (de l'ordre du millimètre contre quelques centimètres pour l'EEG). Une meilleure amplitude dont le maximum peut-être jusqu'à 5 fois supérieure à celle de l'EEG (amplitude max pour l'EEG est d'environ  $10\text{-}20\mu V$ ). L'ECoG est également moins sensible aux artefacts (mouvements musculaires et oculaires). Enfin, elle permet d'enregistrer des phénomènes plus large bande (entre 0 et 500Hz contre 0-40Hz pour l'EEG).

L'utilisation de l'ECoG dans le cadre des ICM est fréquente (Schalk and Leuthardt, 2011, Shih et al., 2012), que ce soit pour le décodage appliqué aux membres supérieurs comme les mouvements de doigts (Scherer et al., 2009, Acharya et al., 2010), de trajectoire de main (Schalk et al., 2007, Gunduz et al., 2009) ou de bras (Pistohl et al., 2008). Il a également été démontré qu'il est possible de décoder des mouvements fins de saisie avec la main, comme des saisies précises avec le bout des doigts versus des saisies avec la main entière (Pistohl et al., 2012), ou encore des mouvements de saisie, de pincée, d'ouverture de la main, de flexion et d'extension du coude ou de bras robotisé (Muller-Putz and Pfurtscheller, 2008, Yanagisawa et al., 2012a). De plus, l'ECoG a également été exploité pour le contrôle d'un curseur 1 ou 2D (Leuthardt et al., 2004, Wilson et al., 2006, Felton et al., 2007, Milekovic et al.,

2012) ainsi que pour la communication palliative (Brunner et al., 2011, Krusienski and Shih, 2011b).

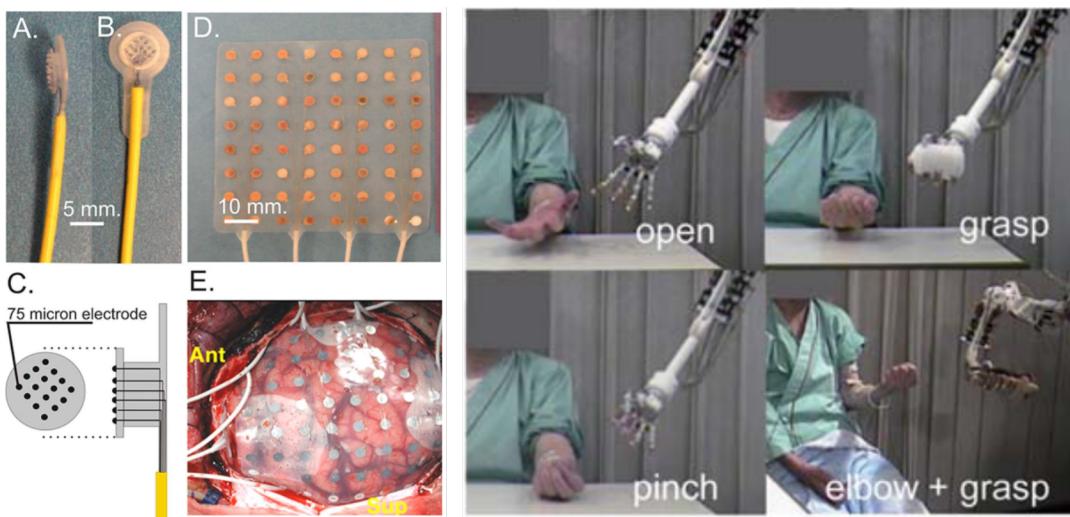


FIGURE 1.5 – A gauche - (A-B) Micro-électrode, (C) Représentation schématique d'une électrode, (D) Grille d'électrodes déposée directement au contact du cortex, (E) Placement chirurgical de la grille d'électrodes (Schalk and Leuthardt, 2011). A Droite - Décodage de mouvements de la main et contrôle d'un bras robotisé (Yanagisawa et al., 2012a)

### 1.1.2.2 Enregistrements non-invasifs

**1.1.2.2.1 Introduction à la production des champs électriques et magnétiques** Avant de présenter les différentes techniques d'enregistrement non-invasives, il m'est apparu intéressant de présenter succinctement la façon dont une population de neurones peut produire des champs électriques et magnétiques. Cela permettra d'une part de comprendre un peu mieux les mécanismes sous-jacents à l'EEG et à la MEG, leur limitations et surtout, leur complémentarité.

L'explication des production de champs électromagnétique s'appuiera sur la figure ci-dessous, issue de Sato et al. (1991) ainsi que sur l'article de Garner et al. (1998).

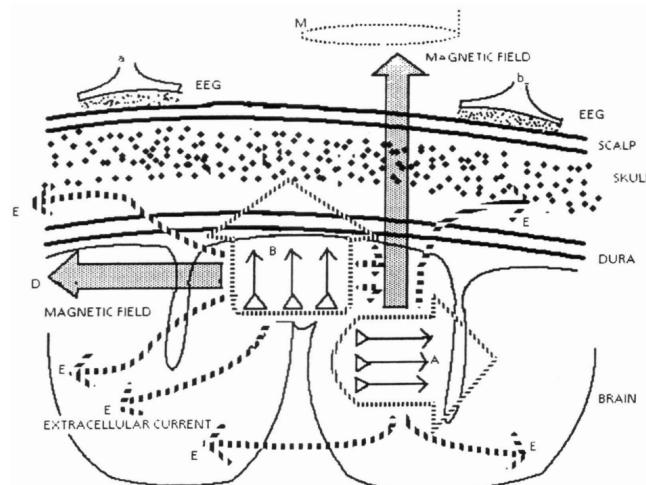


FIGURE 1.6 – Production des champs électriques et magnétiques. Recueil EEG et MEG, limitations et complémentarité (Sato et al., 1991)

a et b sont deux capteurs EEG disposés au contact du scalp. M est un capteur

MEG. **A** et **B** sont deux dipôles matérialisant la somme des courants issus d'une macro-colonne de neurones. Lorsqu'un neurone est excité, il y a libération d'ions au niveau de la membrane des synapses. Ces flux ioniques vont engendrés des courants dits *primaires* ou *sources* et qui seront à l'origine des signaux EEG et MEG. Ces courants primaires vont à leur tour engendrés des courants dits *secondaires* ou *volumiques*, circulant dans tout le volume de la tête (*extracellular current*). Un signal MEG résultera principalement des champs magnétiques produits par les courants sources, contrairement au signaux EEG qui sont majoritairement issus des courants extracellulaires et volumiques. **A** est un dipôle tangentiel (donc parallèle à la surface du crâne) disposé dans un sillon. Le champs magnétique **C** produit par ce dipôle, qui n'est pas filtré par les méninges, l'os et la peau, est bien capté par le capteur MEG. La colonne **B**, qui correspond aux activations situées dans les parties courbes du cortex (gyri) est un dipôle radiale donc perpendiculaire au scalp. Celui-ci produit un champs magnétique **D**, parallèle à la bobine **M**, qui sera donc pas ou peu détecté par la MEG. En revanche, cette source radiale entraîne un fort potentiel électrique. C'est donc un premier point de complémentarité entre la MEG et l'EEG. **A** et **B** engendrent des champs électriques qui seront fortement dispersés, atténus et distordus par la dure-mère, l'os et la peau. Ce phénomène de dispersion explique pourquoi il est beaucoup plus difficile de reconstruire les sources (problème inverse) en EEG plutôt qu'en MEG. Enfin, les propriétés physiques du champs magnétique font qu'il décroît davantage avec la distance que les champs électrique, ce qui limite l'étude des sources profondes mais c'est un deuxième argument justifiant leur complémentarité pour observer l'ensemble des phénomènes.

**1.1.2.2.2 Électroencéphalographie (EEG)** C'est la technique la plus utilisée dans le domaine des ICM pour son aspect pratique, portatif et peu dispendieux. On dispose à la surface de la boîte crânienne un ensemble d'électrodes qui enregistrent, sous forme de potentiel électrique, l'activité résultant d'une population relativement large de neurones. Une électrode (souvent sur le front ou le nez) est considérée comme référence. Le potentiel de cette électrode de référence est ensuite soustrait à toutes les autres pour obtenir une tension. On pourra utiliser un gel entre l'électrode et la boîte crânienne pour adapter l'impédance. L'EEG dispose d'une excellente résolution temporelle mais sa résolution spatiale est moins bonne (en partie dû au fait que les méninges, l'os puis la peau filtre le signal).

Les premiers enregistrements non-invasifs utilisant l'EEG chez l'homme, ont été rapporté par Berger (1929). Depuis, ils n'ont cessé de s'améliorer et à l'heure actuelle on peut trouver des systèmes sans fils et peu dispendieux permettant le contrôle d'une BMI (Lin et al., 2010, Liao et al., 2012, Liu et al., 2012). La transmission sans fil occasionne une perte de signal supplémentaire, donc la plupart des études utilisant l'EEG continu avec les bons vieux câbles et amplificateur. Il existe plusieurs sous-catégories d'ICM utilisant l'EEG. Celles-ci sont basées sur différents types de marqueurs (cf. 1.1.4).

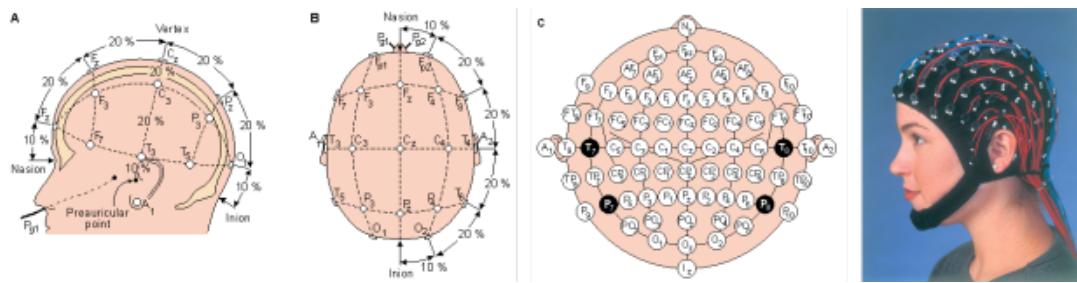


FIGURE 1.7 – Exemple de casque EEG (figure extraite et adaptée de <http://www-psych.nmsu.edu/~jkroger/lab/principles.html>

**1.1.2.2.3 Magnétoencéphalographie (MEG)** Nettamente moins portable que l'EEG, la MEG enregistre les champs magnétiques résultant d'une production de courants intracellulaires. Ces champs magnétiques peuvent être jusqu'à 10 milliards de fois plus faibles que le champ magnétique terrestre ce qui explique l'utilisation d'un blindage pour limiter les artefacts (un blindage en  $\mu$ -métal peut atténuer de  $10^3$  à  $10^4$  l'influence des champs externes). Cet appareil, qui peut être composé de 100 à 300 capteurs, dispose également d'une excellente résolution temporelle et d'une résolution spatiale supérieure à celle de l'EEG.

L'aspect inamovible de la MEG limite son utilisation pour les ICM mais on trouve quand même quelques études ayant testé son utilisation (Mellinger et al., 2007, Walder et al., 2008).

**1.1.2.2.4 Imagerie par Résonance Magnétique fonctionnelle (IRMf)** L'IRMf permet de mesurer l'oxygénation d'aires actives grâce à l'apport en sang. Cette technique se base sur des différences de susceptibilité magnétique du fer entre le sang oxygéné (diamagnétique) et désoxygéné (paramagnétique). Le signal BOLD (Blood Oxygenation Level Dependent) mesure les variations locales du temps de relaxation causées par les modifications hémodynamiques. Si la résolution spatiale de l'IRMf est excellente puisqu'elle est de l'ordre du millimètre, la résolution temporelle, quant à elle, est assez faible (de l'ordre de la seconde) ce qui limite la captation de phénomènes temporellement courts.

L'utilisation de l'IRMf pour les ICM a été exploitée Sitaram et al. (2007a). Cette étude, qui propose plusieurs ICM-IRMf existantes, explique qu'une des principales limitations de cette technique d'enregistrement est son coût et la complexité liée à son usage. Enfin, Weiskopf et al. (2004) explique qu'une autre limitation est qu'il peut s'écouler entre 3 et 6 secondes pour observer les changements hémodynamiques.

**1.1.2.2.5 Functional near-infrared spectroscopy (fNIRS)** Technique d'imagerie relativement récente, puisque la première description du principe fut décrite par (Jobsis, 1977), celle-ci exploite la lumière infra-rouge (longueur d'ondes entre 650 et 1000nm) pour mesurer les variations de concentration de l'hémoglobine oxygénée ( $\text{HbO}$ ) et l'hémoglobine désoxygénée ( $\text{HbR}$ ). La principale limitation de cette technique est qu'elle ne permet pas l'étude de structures profondes (profondeur de 3cm maximum à partir du sommet du crâne).

Cette technique est de plus en plus rencontrée dans la littérature ICM dû à son moindre coût et à sa portabilité. La première ICM utilisant la fNIRS a été décrite par Coyle et al. (2004). Dans cette étude, les auteurs utilisent l'imagerie motrice (compression d'une balle en caoutchouc) et déterminent si l'activité du sujet et en

activité ou au repos. Depuis, bien d'autres études ont suivi, dont beaucoup se focalisent sur l'imagerie motrice (Sitaram et al., 2007b, Nagaoka et al., 2010, Fazli et al., 2012, Mihara et al., 2013, Zimmermann et al., 2013, Kaiser et al., 2014). Naseer and Hong (2015) propose une review récente des principales ICM-fNIRS.

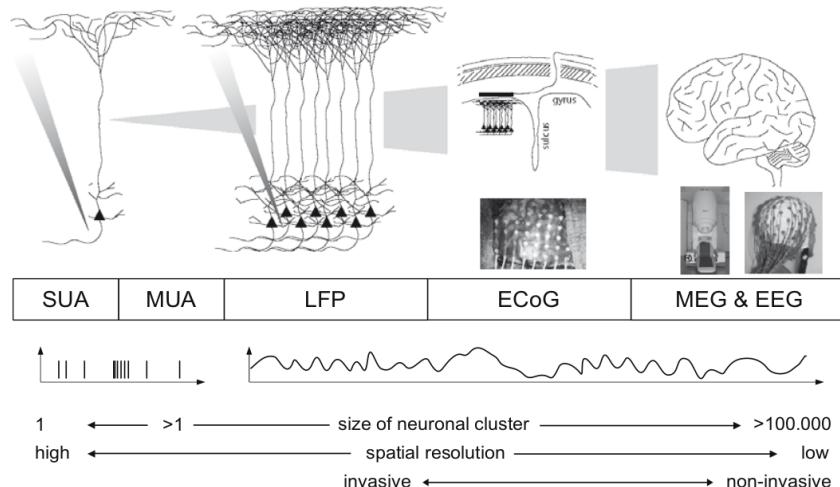


FIGURE 1.8 – Méthodes d’acquisition de l’activité cérébrale (Waldert et al., 2009) classées par invasivité. La figure indique également la taille des populations de neurones enregistrés ainsi que la résolution spatiale intimement liée à l’invasivité.

### Conclusion sur les techniques d’enregistrements

Les méthodes non-invasives ne nécessitent aucune intervention chirurgicale et peuvent donc être appliquées sur n’importe quel individu. De plus, elles sont particulièrement accessibles, c’est-à-dire que leur mise en place est relativement simple.

Toutefois, ces méthodes de mesures souffrent encore de compromis. Si l’EEG et la MEG ont une excellente résolution temporelle, la résolution spatiale est peu précise. A contrario, l’IRMf ou la PET offrent une excellente résolution spatiale mais la dimension temporelle est lésée. Les méthodes invasives ne souffrent pas de ces compromis, puisqu’elles offrent à la fois une excellente résolution temporelle et spatiale, au détriment d’une chirurgie invasive. Enfin, les méthodes non-invasives sont beaucoup plus sensibles aux artefacts (musculaires et oculaires) et ont un rapport signal sur bruit (RSB) inférieur aux enregistrements invasifs.

Malgré tout, les techniques d’enregistrement non-invasives jouissent d’un engouement certain de la part de la communauté scientifique, notamment grâce au critère d’accessibilité. L’EEG a très certainement un bel avenir devant lui, en attendant que des chercheurs réussissent à mettre des roulettes à la MEG ou à l’IRM. Sans nul doute, les techniques non-invasives devraient être au cœur des Interfaces Cerveau-Machine du futur.

### 1.1.3 ICM synchrones/asynchrones et invasives/non-invasives

On peut distinguer différents types d’ICM, sur la base de deux critères (Donoghue, 2002, Lebedev and Nicolelis, 2006, Besserve, 2007, Bekaert et al., 2009) :

- **La synchronisation** : ce critère va définir le fonctionnement interne de l'ICM c'est-à-dire qu'il va fixer la façon dont un utilisateur va pouvoir interagir avec elle. Soit de façon volontaire en modifiant son activité neuronale, ce sont les *ICM asynchrones*, soit de façon imposée par l'utilisation de stimuli externes qui permettront de piloter l'interface, ce sont les *ICM synchrones*.
- **Le type d'enregistrement** : on distinguera les *ICM invasives* des *ICM non-invasives* de part l'utilisation de technique d'enregistrement nécessitant ou non, une chirurgie pour planter des électrodes (cf. 1.1.2).

#### 1.1.3.1 Synchronisation : ICM synchrones et asynchrones

1. *ICM synchrones ou exogènes* : exploitent la réponse du cerveau à des stimuli externe (visuels, auditifs...). Par exemple, un damier composé de cases blanches et noires entraînera une forte variation dans les potentiels visuels. Cette différence de potentiel peut ensuite être détectée puis transformée en commande. Ce type d'ICM présente l'avantage de nécessiter que très peu d'entraînement. En revanche, la réponse étant dépendante du stimulus, le comportement est booléen ce qui limite dans les possibilités pour un contrôle progressif et continu.
2. *ICM asynchrones ou endogène* : ici, grâce à un *feedback*, l'utilisateur change volontairement son activité neuronale pour influer sur le comportement de l'ICM. En pratique, on pourra par exemple se servir de l'imagerie motrice pour avoir un contrôle continu et progressif d'un curseur de souris. Toutefois, les ICM asynchrones nécessitent une longue période d'apprentissage avant de pouvoir reconnaître les *patterns* propres à chaque sujet.

#### 1.1.4 Signaux physiologiques pour le contrôle d'une ICM

Les signaux peuvent classés en deux catégories (Wolpaw et al., 2002, Pfurtscheller et al., 2008) :

- **Les réponses évoquées ou exogènes** : sont produites, sans que le sujet en ait conscience, suite à un stimuli externe.
- **Les réponses spontanées ou endogènes** : celles-ci peuvent être volontairement modifiées par l'utilisateur.

##### 1.1.4.1 Réponses évoquées

Comme décrit plus haut, les signaux évoqués sont provoqués par un stimuli externe et engendrent une réponse spécifique, c'est-à-dire localement dans le cerveau et à des instants précis. Ces signaux ainsi provoqués prendront des valeurs différentes en fonction de l'intention du sujet. Et c'est grâce à ces variations de valeurs que le sujet pourra contrôler la BMI. Le sujet est donc dépendant des stimulus qu'on lui envoie. La première conséquence, c'est que les ICM utilisant les signaux évoqués ne nécessitent pas d'apprentissage particulier. En revanche, elles peuvent entraîner une grande fatigue pouvant altérer les performances de l'ICM (Wolpaw et al., 2002, Curran, 2003). Enfin, les stimulus externes peuvent être de nature différents, que ce soit visuel, auditif ou tactile. Ce type peut s'adapter en fonction de la condition physique du sujet.

Ces réponses sont donc liées à un événement (*event-related*). La plus connue et la plus ancienne est le *Potentiel Évoqué* (PE) recueillis en Électroencéphalographie (et

puisque l'il est lié à l'apparition d'un événement, on parlera d'*Event-Related Potential* (ERP)) et *Champs magnétique Évoqué* en Magnétoencéphalographie . Ces ondes sont obtenues en moyennant un grand nombre d'essais par rapport à l'apparition du stimuli, afin d'écraser les variabilité inter-signal (bruits) et renforcer l'émergence des phénomènes communs aux essais. Parmi ces phénomènes, deux sont particulièrement exploités dans le cadre des ICM, les *Steady-State Evoked Potentials* et l'onde *P300*.

- 1.1.4.1.1 Steady-State Evoked Potentials (SSEP)** Les SSEP sont une réponse naturelle du cerveau à des stimulus envoyés à une certaine fréquence. Bien souvent, le stimuli est de nature visuelle, on parlera donc de *Steady-State Visual Evoked Potentials* (SSVEP). Généralement, les stimulus visuels sont envoyés à des fréquences comprises entre 3.5hz et 75hz et génèrent, dans le cortex visuel, des réponses aux mêmes fréquences (et harmoniques) ([Wolpaw et al., 2002](#), [Beverina et al., 2003](#)).

[Sutter \(1992\)](#) décrit une BCI basée sur les SSVEP (SSVEP-BCI) où l'utilisateur fait face à un écran composé de lettres et de symboles disposés dans une matrice 8x8. Chaque groupe de lettre est flashé à des vitesses différentes pour établir le profil standard du sujet. On demande ensuite à l'individu de choisir une lettre. En flashant de nouveau les éléments de la matrices, les réponses dans le cortex visuel vont différencier du profil standard et c'est ça qui permettra d'établir le choix de l'utilisateur. Depuis, les SSVEP ont été utilisées dans de nombreux autres systèmes, comme la sélection binaire ([Allison et al., 2008](#)), pour épeler ([Cecotti, 2010](#)), le contrôle continu d'un curseur 1D ou 2D ([Trejo et al., 2006](#)), les prothèses ([Muller-Putz and Pfurtscheller, 2008](#)) ou encore pour le jeux ([Lalor et al., 2005](#)).

Ces SSVEP-BCI nécessitent que l'individu ai, d'une part, un restant de contrôle oculo-moteur pour orienter son regard et d'autre part, un système visuelle fonctionnelle. Enfin, puisque des images sont flashées à des vitesses assez importantes, ces ICM ne correspondent pas aux personnes épileptiques.

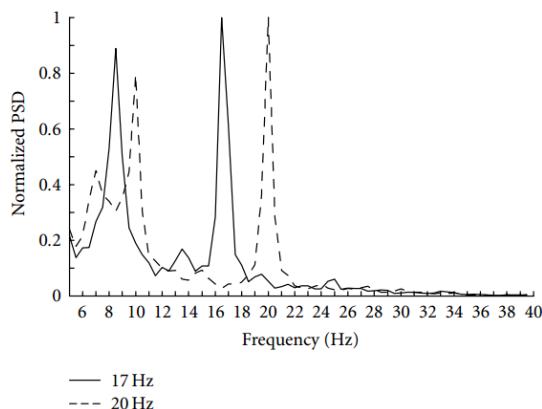


FIGURE 1.9 – Spectre de puissance d'un signal EEG contenant des SSVEP à 17Hz (ligne pleine) et à 20hz (ligne pointillées). Dans cette étude, les auteurs utilisent les SSVEP pour une prise de décision binaire permettant un contrôle d'un jeux virtuel en 3D ([Lalor et al., 2005](#))

- 1.1.4.1.2 Onde P300** Comme décrit ci-dessus, les ERP sont une réponse exogène à un stimulus extérieur pouvant être de nature visuelle, auditive ou tactile. Après moyennage à travers les essais, on peut constater l'apparition d'une série de pic de tension à alternance positive (P) et négative (N) dont la localisation temporelle est fixe (ou contenue dans un intervalle) et connue ([Wolpaw et al., 2002](#), [Beverina et al., 2003](#)).

Ces ondes sont étudiées au dessus du cortex pariétal. Par convention, leur nom dérive de l'instant temporel d'apparition. Ainsi, on parlera des ondes N100 (ou N1), de la P200 (ou P2) et enfin de la P300.

Dans le cadre des ICM, on va surtout s'intéresser à cette dernière, la P300 (bien que la N1 est en souvent également prise en compte). A l'instar des SSVEP, des objets sont aléatoirement mis en surbrillance sur un écran et on demande au sujet de choisir un de ces élément et de compter le nombre de fois que celui-ci est flashé. Enfin, lorsqu'une P300 est détectée dans l'activité neuronale, en remontant 300ms plus tôt, il est possible de retrouver l'élément mis en surbrillance à cet instant. Une autre petite beauté de la P300, c'est que son amplitude est d'autant plus forte que le sujet à réussi à dénombrer le nombre d'apparition.

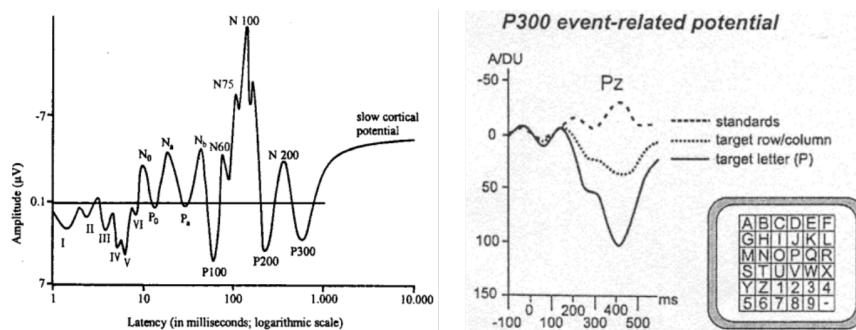


FIGURE 1.10 – (A gauche) ERP générée par un stimuli auditif et apparition des différentes ondes, (A droite) Onde P300. On constate une augmentation d'amplitude lorsque l'utilisateur se focalise sur la lettre P (target letter (P) et target row/column), comparé au profil standard (standards) (Kübler et al., 2001)

La première ICM utilisant la P300 a été introduite par [Farwell and Donchin \(1988\)](#), [Donchin et al. \(2000\)](#). Ce système appelé le *P300-speller*, permet à l'utilisateur d'épeler des mots. Le sujet est devant un écran formé d'une matrice de lettres aléatoirement flashées. Cette BCI est toujours d'actualité et de nombreuses études continuent de la développer ([Vaughan et al., 2006](#), [Hoffmann et al., 2008](#)). L'utilité de la P300 pour contrôler un fauteuil roulant a également été exploré ([Vanacker et al., 2007](#), [Pires et al., 2008](#)). Enfin, [Mugler et al. \(2010\)](#) décrivent une ICM testée sur des sujets sains et atteints de SLA, permettant de contrôler un navigateur internet.

Les systèmes présentés utilisent des stimulus visuels. Toutefois, d'autres ICM reposent sur des stimulus auditifs ([Sellers et al., 2006](#), [Sellers and Donchin, 2006](#), [Furdea et al., 2009](#), [Schreuder et al., 2010](#)) ou tactiles ([Muller Putz et al., 2006](#), [Brouwer and Van Erp, 2010](#)).

#### 1.1.4.2 Signaux spontanés

Les signaux spontanés correspondent à un ensemble de signaux cérébraux que l'utilisateur peut apprendre à moduler. Cet apprentissage, qui peut s'avérer assez long, va permettre de contrôler une Interface Cerveau-Machine .

Parmi ces signaux, on rencontre les *potentiels corticaux lents* ainsi que les *rythmes sensorimoteurs*. Ces derniers sont largement plus présents dans la littérature ICM. Avant tout, pour apprendre à moduler sont activité neuronale, l'utilisateur devra utiliser une stratégie mentale

**1.1.4.2.1 Stratégies mentales pour le contrôle d'une ICM** Pour assimiler le contrôle des signaux spontanés, il est nécessaire que l'utilisateur passe par une phase d'apprentissage pouvant être soit autonome, soit guidée via l'utilisation de l'imagerie motrice (Curran, 2003).

1. Apprentissage autonome (*operant conditioning, implicit learning ou operant self-control*) : le choix de la stratégie mentale est laissé à l'utilisateur. C'est à lui de trouver celle qui lui convient le mieux. Dans ce cas, le système doit impérativement fournir un feedback au sujet afin que celui-ci comprenne comment la machine.
2. Imagerie motrice : pour cette stratégie mentale, on demandera donc à l'utilisateur d'imaginer des mouvements qui lui seront imposés en amont (cf. 1.1.4.2.2)

Ces stratégies mentales conditionnent le mode d'interaction de l'utilisateur avec la machine. Il est fréquent que, dans les premiers stades de l'entraînement, les sujets soumis à l'apprentissage autonome utilisent naturellement l'imagerie motrice (Wolpaw et al., 1991, Birbaumer et al., 1999).

L'*operant conditioning* demande un long apprentissage (de plusieurs semaines à années) mais les systèmes l'utilisant rapportent de bonnes performances et une grande stabilité (Fetz, 1969, Wolpaw et al., 1991, Birbaumer et al., 1999, Wolpaw and McFarland, 2004a, Birbaumer, 2006, Vaughan et al., 2006, Wolpaw, 2007). Certaines tâches motrices peuvent ne pas convenir aux personnes ayant un déficit moteur de longue date ou depuis la naissance tout comme les tâches visuelles pour les personnes aveugles de naissance (Curran, 2003). Donc, autre avantage de l'apprentissage autonome, il permet de prendre en compte la préférence ou le confort d'utilisation qui pourrait jouer un rôle dans les performances de l'ICM (Pfurtscheller et al., 2000).

**1.1.4.2.2 Imagerie motrice** L'imagerie motrice (IM) est définie par la représentation d'une action qui n'est pas suivie de son exécution. Dans les sections précédentes, nous avons vu que l'IM permettait de contrôler une Interface Cerveau-Machine. Les substrats neuronaux mis en jeu lors de l'imagination d'un mouvement sont sensiblement les mêmes que lors de l'exécution de ce mouvement, ce qui permet donc aux personnes à mobilité réduite de solliciter les aires motrices sans la possibilité d'accomplir le mouvement.

Cette section a pour objectif de raffiner l'imagerie motrice, c'est-à-dire introduire les différents types d'imagerie, différentes applications et enfin, leur utilisation pour les ICM.

1. **Types d'IM :** les différents types d'imagerie reposent sur l'exploitation de modalités sensorielles, c'est-à-dire sur les sens (Kosslyn et al., 1990). On distingue donc l'imagerie visuelle, tactile, auditive et olfactive. L'imagerie visuelle se décompose en deux sous-divisions : interne (ou à la première personne) et externe (ou à la troisième personne) (Ruby and Decety, 2001, Jackson et al., 2006, Lorey et al., 2009). Dans le premier cas, on est directement acteur de l'action, donc par exemple, on pourra imaginer effectuer un mouvement de bras de manière similaire à une réelle exécution. En imagerie visuelle externe, on est spectateur d'un mouvement pouvant être effectué par une autre personne, ou par soi-même. À ces différents types d'imagerie, s'ajoute l'imagerie kinesthésique basée sur des informations proprioceptives. Bien que à priori proches, imagerie visuelle et kinesthésique sollicitent des substrats neuronaux différents (Solodkin, 2004, Guillot et al., 2009).

Si une personne imagine avoir une balle dans la main, c'est de l'imagerie

visuelle interne. Si elle imagine que cette balle est tenue par une tierce personne, c'est de l'imagerie externe. Enfin, si la personne imagine les sensations que peut procurer les propriétés de cette balle (comme sa texture, sa malléabilité, son poids ou sa taille) c'est de l'imagerie kinesthésique. De nombreuses études ont comparé les différents types d'imagerie permettant de mettre en valeur les réseaux associés (Jiang et al., 2015, Seiler et al., 2015).

2. **Utilisation de l'IM :** dans le paragraphe précédent nous avons vu que l'IM peut être utilisée comme stratégie mentale pour le contrôle d'une ICM (cf. 1.1.4.2.1). L'IM peut également être utilisée pour faciliter et améliorer un apprentissage, notamment pour la pratique sportive de haut niveau (Driskell et al., 1994, Guillot et al., 2008, Schuster et al., 2011, Di Rienzo et al., 2016), pour la réhabilitation motrice (Jackson et al., 2001, Sharma et al., 2006, de Vries et al., 2011, Malouin et al., 2013, Di Rienzo et al., 2014).

Dans le cadre des Interface Cerveau-Machine, Neuper et al. (2005) ont étudié le décodage de différentes modalités d'imagerie en comparaison avec le repos. L'intérêt de cette étude et de renseigner sur le type d'IM a privilégié pour le contrôle d'une ICM. Pour cela, les sujets effectuent différentes tâches avec une balle : soit ils exécutent des mouvements (ME) (pression continue), soit ils observent une main animée (OOM), soit ils imaginent les sensations procurées par cette balle (MIK) (imagerie kinesthésique) soit ils imaginent une main effectuant des mouvements sous forme de film (MIV) (imagerie visuelle). Neuper et al. (2005) montre alors que le décodage avec la condition MIK (67%) est nettement supérieur à celui atteint avec la condition MIV (58%). De plus, le décodage MIK se focalise essentiellement autour des aires sensorimotrices, tout comme pour l'exécution, alors qu'il n'y a pas de pattern réellement émergent pour le MIV. En conclusion, les auteurs conseillent plutôt l'utilisation de l'imagerie kinesthésique à l'imagerie visuelle comme stratégie mentale pour le contrôle d'une ICM.

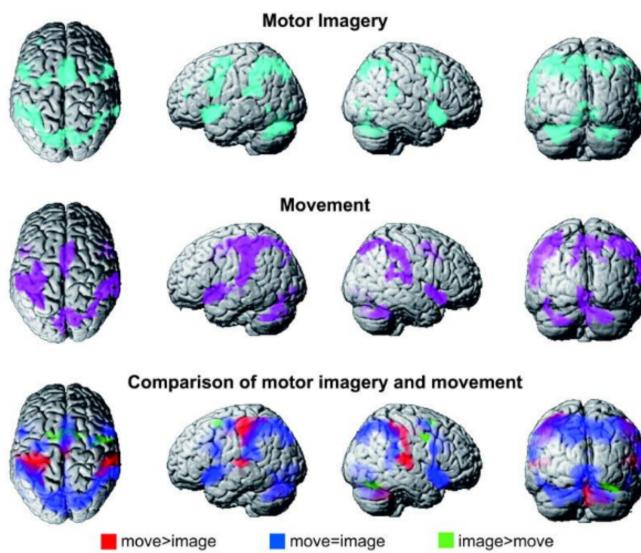


FIGURE 1.11 – Comparaison des aires actives lors d'un mouvement imaginé ou exécuté (Hanakawa et al., 2008)

- 1.1.4.2.3 **Potentiel corticaux lents** Les potentiels corticaux lents (ou *Slow Cortical Potential* (SCP)) sont des variations lentes du potentiel cortical qui ont lieu entre 500ms et 10s (Birbaumer et al., 1990, Birbaumer, 1997, Birbaumer et al., 2003, Birbaumer, 1999,

Kleber and Birbaumer, 2005). Un utilisateur peut apprendre à contrôler l'amplitude de ces signaux, notamment grâce à un bio-feedback où l'utilisateur voit son activité se moduler en temps réel sur un écran. les SCP peuvent prendre des valeurs positives, généralement lors d'un mouvement ou tout autre fonction impliquant une activation corticale, ou négative lors d'une réduction de l'activité corticale (Rockstroh, 1989, Birbaumer, 1997, Wolpaw et al., 2002).

Les ICM basées sur les SCP vont nécessiter un seuil et, en fonction de ce seuil, le sujet module son activité lui permettant un contrôle binaire (Birbaumer et al., 1999). Toutefois, l'apprentissage peut s'avérer extrêmement long. Le dispositif de traduction de pensées (Kübler et al., 1999) (*Thought Translation Device (TTD)*) , est un appareil destiné à entraîné des sujets puis à tester leur apprentissage pour épeler des mots (Perelmouter et al., 1999, Birbaumer et al., 2003). Le software est développés en C++ et utilise des outils de *BCI2000*, plate-forme de développement du groupe central de Wadsworth (Wolpaw et al., 2003, Schalk et al., 2004). Le TTD a été développé pour les sujets complètement paralysés (LIS et CLIS). Tout d'abord, les sujets s'entraînent de manière autonome à moduler leur SCP notamment grâce à un retour visuel et auditif et un renforcement positif (visage souriant et musique lorsque le contrôle est réussi). Après cette phase d'apprentissage machine, les sujets sont ensuite testés pour sélectionner des lettres ou des mots (Birbaumer et al., 2000, 2003).

Les SCP ont également été testées à des fins cliniques. Rockstroh et al. (1993), Kotchoubey et al. (1998) ont montré qu'après un an et demi d'apprentissage autonome à moduler positivement et négativement les SCP, des sujets atteints d'épilepsies pharmacorésistantes ont vu leur crise diminuer de 50% en moyenne (certains sujets n'avaient plus aucune crise tandis que d'autres n'ont eu aucun changement).

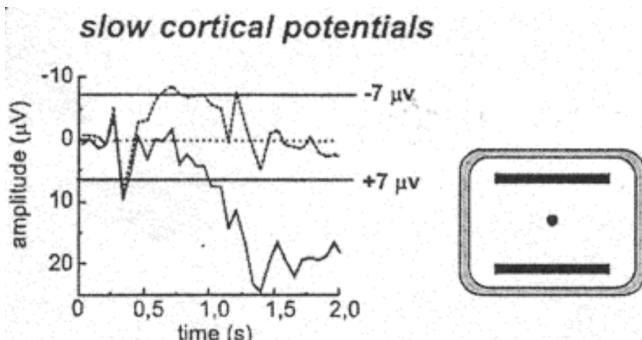


FIGURE 1.12 – Feedback visuel des SCP renvoyées par le TTD - A droite, un curseur peut se déplacer entre les deux objectifs (en haut et en bas). Les négativités corticales bougent le curseur vers le haut (ligne du haut en pointillé) à l'inverse, les positivités corticales permettent de faire descendre le curseur (ligne pleine du bas). Toute modulation de 7 $\mu$ V est considérée comme une réussite (Kübler et al., 2001)

- 1.1.4.2.4 Le Bereitschaftspotential** Le Bereitschaftspotential (BP), ou *readiness potential*, est un potentiel moteur focalisé principalement dans la pré-aire motrice supplémentaire (pré-SMA) et SMA (Kornhuber and Deecke, 1965, Ball et al., 1999, Brunia and Van Boxtel, 2000). Le BP comprend deux composantes dont la première, la composante précoce (ou *early BP*), commence environ deux secondes avant le début du mouvement qui est ensuite suivie d'une pente négative tardive (ou *late BP*), environ 400ms avant le début du mouvement (Shibasaki and Hallett, 2006). Il a été montré que le BP dépend de paramètres de mouvements tels que l'état de préparation ou encore la répétition et précision de mouvement (Birbaumer et al., 1990). Enfin, le

BP est d'avantage présent lors de mouvements auto-initiés, comparés à des mouvements imaginés ou basés sur un *cue* externe (Deiber et al., 1999, Jenkins et al., 2000, Jankelowitz and Colebatch, 2002).

- 1.1.4.2.5 Rythmes sensorimoteurs (RSM)** Les RSM (ou *Sensorimotor rhythms* (SMR)) correspondent à l'amplitude de signaux, au dessus du cortex sensorimoteur, dans des bandes de fréquences spécifiques. Les plus fréquents sont les rythmes  $\mu$  8-13hz et  $\beta$  13-30hz. Un utilisateur peut apprendre à moduler l'amplitude de son activité cérébrale dans ces bandes pour contrôler une Interface Cerveau-Machine notamment par le biais de l'imagerie motrice (cf. 1.1.4.2.2). On appelle *Event-Related Synchronization* lorsque l'amplitude augmente et *Event-Related Desynchronization* lorsqu'elle diminue (Pfurtscheller and Lopes da Silva, 1999, Pfurtscheller et al., 2008). Ces rythmes ont très largement été exploités pour contrôler un BMI, que ce soit pour déplacer un curseur dans une, deux ou trois dimensions (Wolpaw and McFarland, 2004b, McFarland et al., 2008, Kayagil et al., 2009, McFarland et al., 2010, Doud et al., 2011), pour épeler (Neuper et al., 2006, Vaughan et al., 2006), pour contrôler une prothèse (Pfurtscheller et al., 2000, Müller-Putz et al., 2005, McFarland and Wolpaw, 2008) ou un fauteuil roulant (Tanaka et al., 2005, Galán et al., 2008b)

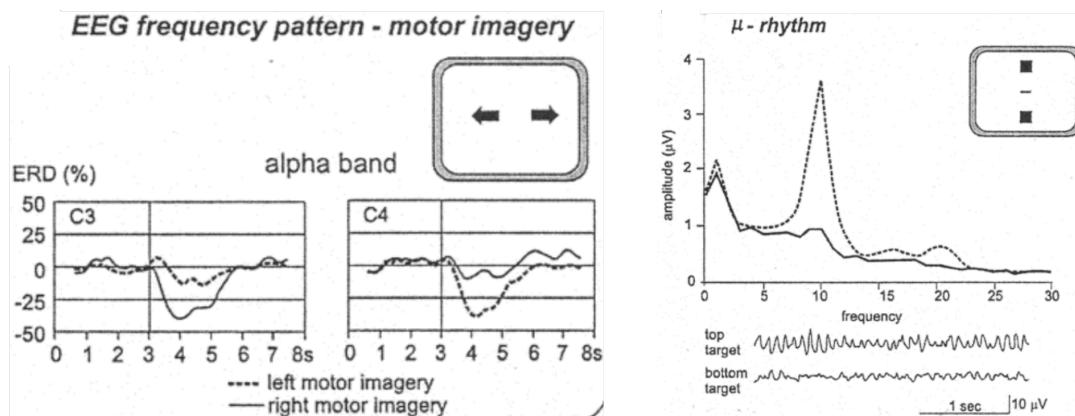


FIGURE 1.13 – (A gauche) Utilisation de l'imagerie motrice pour contrôler la direction d'un curseur et tracé de l'activité neuronale pour les électrodes C3 (hémisphère gauche) et C4 (hémisphère droit) en utilisant l'activité neuronale contenue dans la bande  $\alpha$  - L'utilisateur imagine des mouvements des mains droite ou gauche pour déplacer le curseur. L'imagination d'un mouvement de main gauche entraîne une ERD dans l'hémisphère droit (C4) mais l'activité est maintenue sur C3 (A droite) Modulation du  $\mu$  au dessus de C3-C4 pour contrôler un curseur vers le haut (augmentation du  $\mu$ , ligne pointillée) ou vers le bas (baisse du  $\mu$ , ligne pleine) (Kübler et al., 2001)

- 1.1.4.2.6 Autres marqueurs** Les marqueurs présentés ci-dessus sont ceux que l'on retrouve le plus largement à travers la littérature. Toutefois, d'autres marqueurs de l'activité neuronale sont étudiés essentiellement d'un point de vue neuro-scientifique, c'est-à-dire pour améliorer la compréhension des phénomènes physiologiques.

**Décomposition phase-amplitude :** tout signal temporel réel peut être décomposé en un signal d'amplitude (ou enveloppe car elle va suivre les maxima du signal) et un signal de phase qui indique la situation instantanée d'un cycle (pic, creux, passage à zéro ...). La transformée d'Hilbert permet de décomposer ainsi n'importe quel signal (4.1.1.2).

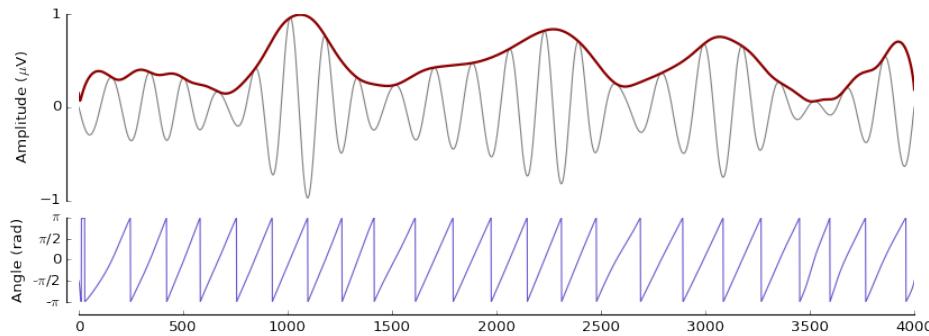


FIGURE 1.14 – (En gris) Un signal original, (En rouge) L'amplitude du signal, (En bleu) la phase instantanée

**Phase :** si l'amplitude a très largement été étudiée, que ce soit dans le cadre des ICM, du décodage et de son implication physiologique, la phase reste à l'heure actuelle l'objet d'un nombre plus restreint d'études. Toutefois, quelques études ont montré l'implication de la phase basse-fréquence dans l'encodage neuronal de mouvements (Hammer et al., 2013, 2016).

**Couplage inter-fréquences locaux ou à distance :** ou *Cross-Frequency Coupling* (CFC) regroupe un ensemble de marqueurs qui, contrairement à ceux présentés ci-dessus, étudient une forme de corrélation entre deux signaux pouvant être soit locaux (comme provenant d'une même électrode) ou à distance pour évaluer des synchronisations d'aires cérébrales. Ces *features* sont étudiés dans des bandes de fréquences particulières et vont donc nécessiter une filtrage en amont. Ensuite, on pourra extraire de ces signaux filtrés les informations de phase et d'amplitude afin d'étudier différentes formes de couplage :

**Couplage à distance phase-phase :** le *Phase-Locking Value* (PLV) (Lachaux et al., 1999, 2000) est un des outils permettant de déterminer la synchronisation de phase. Le couplage inter-fréquence phase-phase semble jouer un rôle dans la communication inter-structures (Fries, 2005, Gregoriou et al., 2009, Siegel et al., 2009, van Elswijk et al., 2010)

**Couplage à distance amplitude-amplitude :** bien que le rôle physiologique du couplage inter-fréquences amplitude-amplitude soit encore incertain, de tel couplages ont été décrits dans la littérature (Friston, 1997, Shirvarkar et al., 2010, Siegel et al., 2009)

**Couplage local ou à distance phase-amplitude :** ou *Phase-amplitude coupling* (PAC), fait intervenir des signaux pris dans deux bandes de fréquence et permet d'évaluer la façon dont ces signaux évoluent l'un avec l'autre. Plus précisément, on considère un signal dans les basses fréquences (BF), typiquement dans les bandes delta, thêta ou alpha et on extrait la phase de ce signal. Ensuite, on considère l'amplitude d'un signal haute fréquence (HF), souvent dans la bande gamma. Le PAC renseigne si la phase des BF et l'amplitude des HF évoluent de manière synchrones. A noter ici que il n'est pas question d'inférer une relation de cause/conséquence entre la phase des BF et l'amplitude des HF. Autrement dit, le PAC ne permet pas de conclure que l'un vient moduler l'autre. Plusieurs outils méthodologiques permettant une mesure du PAC ont été proposés (cf. 4.1.4)

Le rôle physiologique du PAC est encore discuté (Canolty and Knight, 2010,

Hyafil et al., 2015) tout comme son implémentation méthodologique (Aru et al., 2015), mais il a été observé dans des tâches variées (Bruns and Eckhorn, 2004, Voytek, 2010, Soto and Jerbi, 2012), dans la maladie de Parkinson (de Hemptinne et al., 2013), dans la prise de risques (Lee and Jeong, 2013) ou dernièrement dans l'encodage mémoriel (Lega et al., 2016). Enfin, Yanagisawa et al. (2012b) ont pu montrer l'existence d'un couplage alpha-gamma dans le cortex sensorimoteur durant une période de repos. Ensuite, ce couplage diminue avec l'exécution de tâches motrice (mouvements de saisie, de pincée et d'ouverture de main). Enfin, cette étude a montré que ce couplage alpha-gamma, dans cette région sensorimotrice, ne permettait pas de décoder ces différents types de mouvements. A noter que généralement l'amplitude est prise dans la bande gamma mais Cohen et al. (2008) ont démontré l'existence d'un couplage entre la phase du delta et thêta avec l'amplitude de l'alpha et du gamma dans la prise de décision.

#### 1.1.4.3 Enregistrement : ICM invasives et non-invasives

1. *ICM invasives ou directes* : basées sur un enregistrement invasif de l'activité neuronal, la qualité du signal étant excellente les possibilités d'exploitations et d'améliorations futures sont grandes. En revanche, cela va avec les risques que comporte la chirurgie.
2. *ICM non-invasives ou indirectes* : de la même manière, ces ICM utilisent des enregistrements non-invasifs. Le terme *indirecte* signifie que l'on enregistre pas directement des décharges de neurones mais un phénomène lié à une population de neurones (consommation d'oxygène pour Imagerie par Résonance Magnétique fonctionnelle , champs électrique et magnétique pour l'EEG et la MEG). Ces ICM sont les plus répandues et représentent un enjeu majeure de part leur accessibilité.

## 1.2 Data-mining EN NEUROSCIENCES

### 1.2.1 Exploration des données

### 1.2.2 Outils de validation



# ICM ET NEUROPHYSIOLOGIE

Cette chapitre a pour objectif d'introduire les concepts neurophysiologiques sous-jacents au contrôle et à la définition d'une Interface Cerveau-Machine . Le but sera d'expliquer l'origine et la composition des différents types de signaux cérébraux permettant à un utilisateur de piloter une ICM. Pour cela, nous verrons tout d'abord les bases physiologiques liées à la motricité, puis de là, nous verrons quels types de signaux sont couramment extraits et exploités des principales régions motrices pour le contrôle d'une ICM.

## 2.1 BASES PHYSIOLOGIQUES LIÉES À LA MOTRICITÉ

Cette section servira avant tout à identifier les principaux acteurs de la planification et de l'action motrice. Pour cela, on introduira les notions nécessaires sur le cortex ainsi que les principales aires sollicitées dans le cadre des ICM. Puis, de ces aires, nous verrons le principe du rythme cérébral qui permettra ensuite de justifier les principaux processus neuronaux liés à la motricité.

### 2.1.1 Notions sur le cortex

Le cerveau est l'organe le mieux protégé du corps. Tout d'abord, la peau est une première barrière avec l'extérieur. Ensuite, l'os (le crâne) assure une protection mécanique c'est-à-dire qu'il va protéger en cas de coups ou de chocs. Viennent ensuite les méninges (dure-mère, l'arachnoïde et la pie-mère), qui sont des membranes enveloppant le système nerveux central (SNC) et qui vont permettre d'une part d'amortir les chocs et d'autre part, empêcher le cerveau de s'abîmer avec l'intérieur du crâne.

La couche suivante est appelée cortex, mesure entre 1 et 5mm d'épaisseur et correspond au corps cellulaire des neurones. Le cortex se divise en quatre régions :

- Lobe frontal : lié à la planification, la prise de décision et à l'exécution motrice
- Lobe occipital, pour le traitement de la vision, temporal pour l'audition et somatosensorielle pour le touché.
- Lobe pariétal, qui est considéré comme un cortex associatif c'est-à-dire qu'il joue un rôle important dans l'intégration des informations sensorielles.

Le cortex moteur, situé dans le lobe frontal, est un ensemble d'aires destinées à la planification et à l'exécution de mouvements volontaires. Il se compose du *cortex pré-moteur* qui va planifier et organiser les mouvements en fonction des informations sensorielles et qui seront ensuite exécutés par le *cortex moteur primaire*. Enfin, l'*aire motrice supplémentaire* (SMA) est impliquée dans la planification des mouvements complexes et dans la coordination.

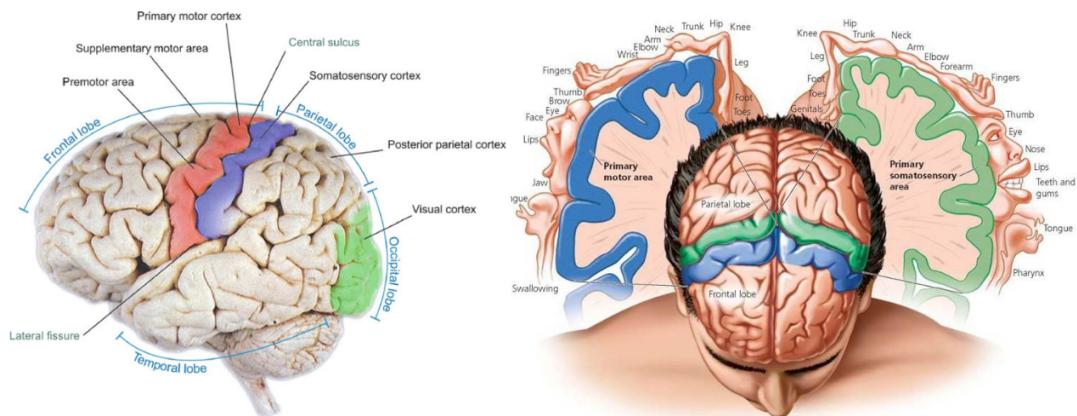


FIGURE 2.1 – (A gauche) - Principaux lobes constituant le cortex et localisation de la SMA, du cortex pré moteur et moteur primaire (Graimann et al., 2009), (A droite) - Vue en coupe du cortex moteur et sensorimoteur permettant le contrôle de différentes régions (<http://schoolbag.info/biology/humans/9.html>)

## 2.1.2 L'activité rythmique et lien avec la motricité

L'activité rythmique correspond à l'activité neuronale prise dans des bandes de fréquence précises (Niedermeyer, 2004, Niedermeyer and da Silva, 2005) (bien qu'elle peut différer légèrement d'une publication à l'autre). Leur extraction se fera donc à l'aide de filtrage.

**2.1.2.0.1 Définition et rôle fonctionnel des rythmes cérébraux** : l'apparition de ces rythmes résulte d'une synchronisation/désynchronisation d'une population large de neurones et vont permettre de caractériser l'état cognitif d'un individu. On peut distinguer six rythmes cérébraux ayant des propriétés spatio-temporelle différentes. La littérature attribue un très grand nombre de rôles fonctionnels à ces bandes. Les descriptions ci-dessous illustrent une partie des rôles qui leur sont affectés :

**Delta** ( $\delta \in [2, 4] \text{Hz}$ ) : principalement présentes chez les très jeunes enfants, ces rythmes très lents et de forte amplitude sont particulièrement présents durant le sommeil profond (Amzica and Steriade, 1998, Silber et al., 2007).

**Theta** ( $\theta \in [5, 7] \text{Hz}$ ) : les rythmes thêta sont proéminents pour la mémorisation à long terme (Klimesch, 1999) ou dans les états de somnolence et d'hypnose (Schacter, 1977).

**Alpha** ( $\alpha \in [8, 13] \text{Hz}$ ) : la puissance dans la bande alpha est réputée pour augmenter lorsque les yeux sont fermés (Berger, 1929). Ce rythme est également important dans la mémorisation (Klimesch, 1999). Markand (1990), Klimesch et al. (2007) décrit plus spécifiquement les rôles supposés de l'alpha.

**Mu** ( $\mu \in [8, 13] \text{Hz}$ ) : bande de fréquence identique à celle de l'alpha mais le rythme  $\mu$  correspond en fait à l'alpha pris dans les aires sensorimotrices (Markand, 1990). Un des phénomènes important avec ce rythme qui sera particulièrement exploité par les ICM, c'est la baisse occasionnée lors de tâche motrice ou d'imagerie motrice dans l'hémisphère contralatéral (Salmelin and Hari, 1994, Crone et al., 1998, Pfurtscheller and Lopes da Silva, 1999, Pfurtscheller et al., 2008).

**Beta** ( $\beta \in [13, 30] \text{Hz}$ ) : tout comme le rythme  $\mu$ , on observe une baisse dans la bande  $\beta$  lors de tâche d'exécution motrice au dessus du cortex moteur sauf que celle-ci est suivie par une augmentation une à deux secondes après la

fin du mouvement (Pfurtscheller and Berghold, 1989, Cassim et al., 2001). Ce phénomène s'appelle le rebond bêta.

**Gamma** ( $\gamma \geq 40\text{Hz}$ ) : les oscillations gamma sont présentes en réponse à un stimulus sensoriel, lors de tâche motrices (Crone, 1998, Tallon-Baudry and Bertrand, 1999) ou dans des tâches de recherche visuelle (Tallon-Baudry et al., 1997). La bande haute du  $\gamma$  sera limitée par la fréquence d'échantillonnage (critère de Shannon) et donc, par la technique d'enregistrement (l'ECoG et la SEEG permettent d'étudier des phénomènes plus large bandes que la MEG ou l'EEG par exemple)

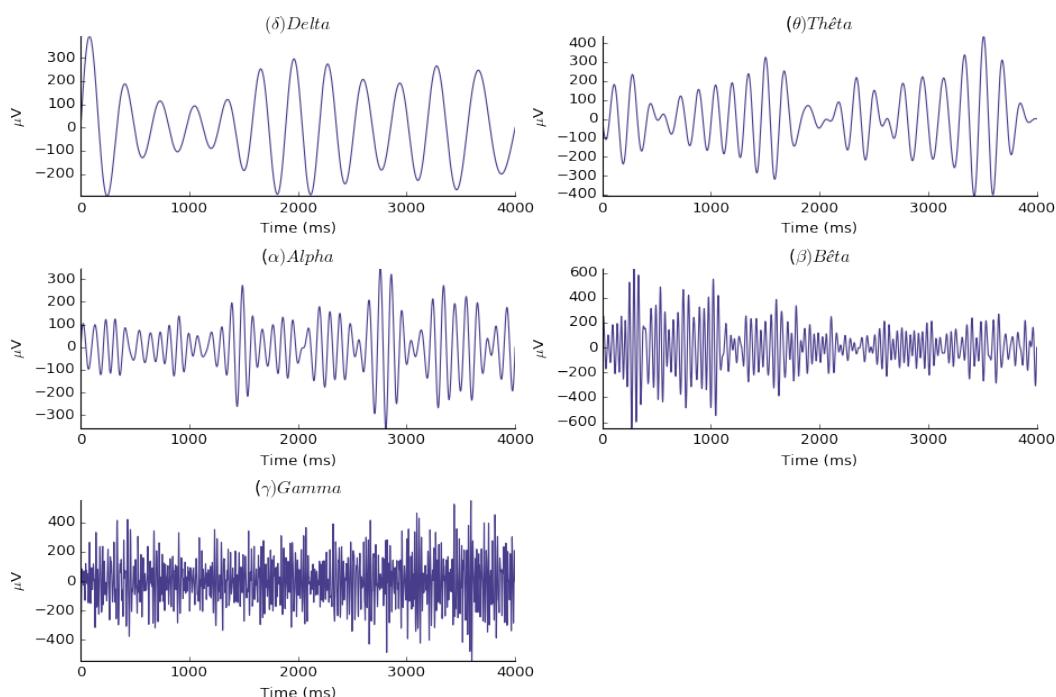


FIGURE 2.2 – Exemple d’activités rythmiques dans les différentes bandes de fréquences pour un essai unique issue de l’activité sEEG

## 2.2 DÉCODAGE DIRECTIONNEL DES MEMBRES SUPÉRIEURS

Que ce soit pour déplacer un curseur, pour le contrôle d'une prothèse ou pour toute autre application, le décodage de directions est au centre du fonctionnement d'un bon nombre d'Interface Cerveau-Machine . Pour ce faire, les équipes utilisent bien souvent une tâche appelée *center-out* où le sujet bouge son bras du centre vers une direction imposée, que ce soit avec le curseur d'une souris ou avec un joystick. A noter ici que certaines équipes ce sont également intéressées au moment où le sujet prépare le mouvement. Le but sera donc, en *offline*, d'essayer de retrouver la direction effectuée par le sujet que ce soit pendant la période d'exécution ou de préparation en utilisant l'activité neuronale seule.

Le décodage directionnel fait intervenir trois principales composantes :

1. L'aire cérébrale étudiée (cortex moteur primaire, pré moteur, pariétal, préfrontal...). Ces aires peuvent apporter des informations complémentaires à différents instants temporels durant la tâche, c'est pourquoi elles sont bien souvent combinées pour améliorer l'acuité de décodage.

2. L'optimisation des marqueurs : le travail en *offline* permet une exploration plus vaste des marqueurs puisqu'il n'y a pas la contrainte d'application temps-réel.
3. L'optimisation des paramètres de classification : de la même façon, on va pouvoir tester des classificateurs avec des méthodologies plus lourdes.

### 2.2.1 Décodage directionnel

L'utilisation des micro-électrodes chez le singe ont permis de mettre en valeur le *directional tuning* (Georgopoulos et al., 1982, 1986, Georgopoulos and Carpenter, 2015), c'est-à-dire que le taux de décharges dépend de la direction et donc, un neurone pourra afficher une direction préférentielle dans laquelle il déchargera davantage. Le *directional tuning* a également été étudié avec le signal LFP chez le singe (Mehring et al., 2004, Rickert, 2005). Chez l'homme, l'utilisation du signal filtré basse fréquence en ECoG (Mehring et al., 2004, Ball et al., 2009) et l'amplitude prise dans différentes bandes de fréquences (Leuthardt et al., 2004, Ball et al., 2009) ont aussi permis de mettre en évidence des modulations spécifiques en fonction des directions permettant un décodage. Pour quatre directions, Ball et al. (2009) a obtenu des décodages supérieurs à 80% en ECoG, en combinant les électrodes du cortex moteur. Plus récemment, Gunduz et al. (2016) montre des décodages significatifs en utilisant l'activité gamma en ECoG que ce soit pour décoder l'exécution ou la préparation de mouvements effectués dans 8 directions (avec un maximum de 30% pour décoder la préparation et 50% pour l'exécution dans les quatre directions en combinant jusqu'à 30 électrodes ECoG).

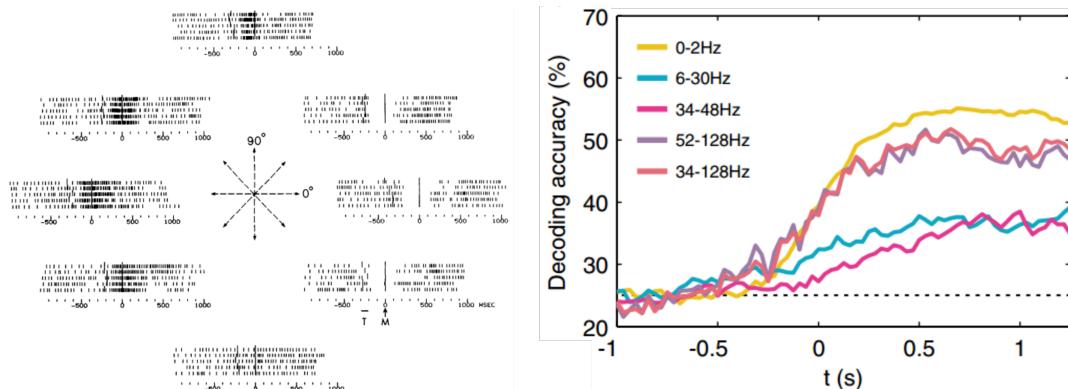


FIGURE 2.3 – (A gauche) Cinq essais montrant le taux de décharge d'un neurone moteur en fonction de directions de mouvements (Georgopoulos et al., 1982). On constate que ce neurone affiche une préférence pour les directions vers la gauche (A droite) Évolution temporelle du décodage de quatre directions utilisant l'amplitude dans différentes bandes de fréquences pour des électrodes prises dans le cortex moteur primaire (Ball et al., 2009)

Le décodage directionnel a également été exploré dans le cadre de données non-invasives, en EEG et MEG (Waldert et al., 2007, 2008). Dans cette étude, l'auteur montre des décodages jusqu'à 67% pour décoder quatre directions de mouvements. Il est intéressant de noter que ces décodages significatifs ont été obtenus en utilisant le signal filtré dans les basses fréquences et que la puissance dans les bandes bêta et gamma ne semblent pas décoder. Enfin, l'EEG et la MEG fournissent des résultats similaires et la combinaison des deux améliore peu les résultats dans cette étude. Hammon et al. (2008) ont également montré des décodages significatifs pour décoder la directionnalité que ce soit pendant l'exécution, ou la préparation motrice.

### 2.2.2 Prédiction continu de la cinétique du mouvement

Les études ci-dessus tentent de décoder les directions, c'est-à-dire d'essayer de retrouver en *offline* les directions effectuées par le sujets. D'autres articles tentent de décoder la position, la vitesse ou l'accélération du mouvement, notamment via l'utilisation du filtre de Kalman (Wu et al., 2002, 2003, 2004, 2006, Pistohl et al., 2008, Li et al., 2009)

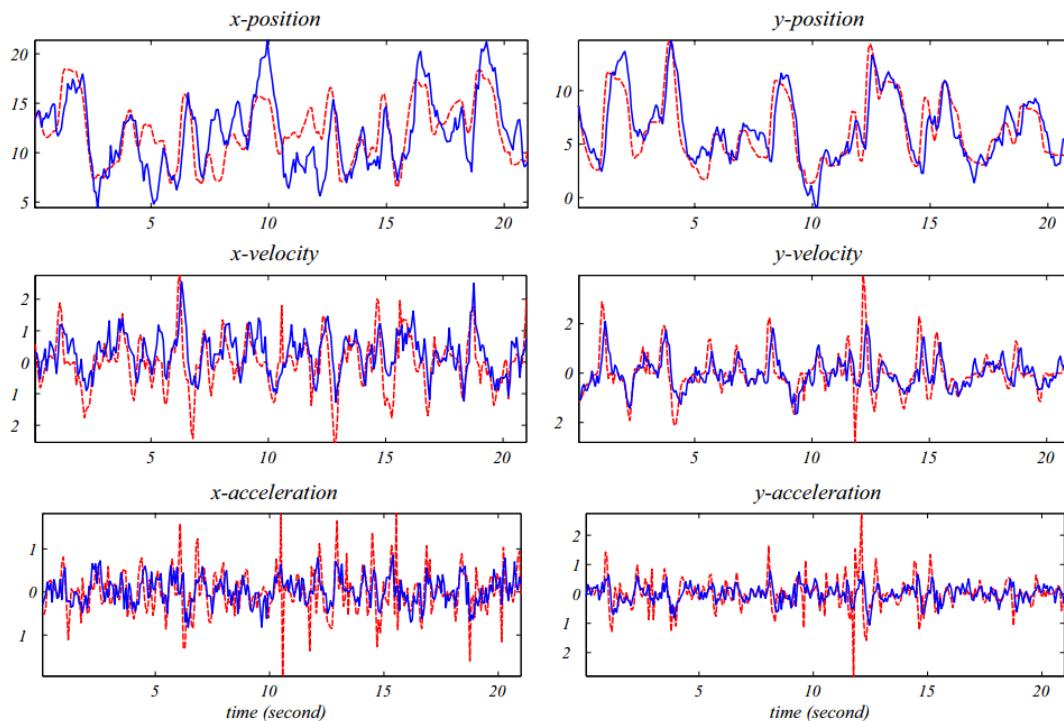


FIGURE 2.4 – Décodage continu de la cinétique d'un mouvement 2D (position/vitesse/accélération) (Wu et al., 2003). Le mouvement réellement effectué (en rouge) est reconstruit en utilisant l'activité spike et le filtre de Kalman (en bleu)

#### Conclusion du chapitre 2

Les Interface Cerveau-Machine peuvent donc être contrôlées par deux types de signaux cérébraux. D'une part, les *réponses évoquées* (SSEP, P300...), correspondant à des réponses cérébrales suite à un stimulus externe. L'avantage de ce type d'ICM est qu'elles ne nécessitent pas de période d'apprentissage. D'autre part, les *réponses spontanées* (SCP, rythmes sensorimoteurs), correspondant à une activité cérébrale que le sujet peut volontairement modifier. Pour apprendre à modifier l'activité de son cerveau, l'utilisateur pourra soit apprendre de façon autonome ou via les différentes modalités d'imagerie motrice (visuelle, kinesthésique...).



# OBJECTIFS DE LA THÈSE

3

Durant cette thèse, nous avons principalement utilisé les données intracrâniennes issues d'une tâche motrice (cf. 6) afin d'étudier les axes suivants :

**Utilisation des outils de *machine learning*** : les méthodes d'apprentissage machine ont été utilisées comme un outil de validation pour explorer les différences entre des états moteur.

**Comparatif d'états moteur** : dans un premier temps, nous avons chercher à raffiner la connaissance des substrats neuronaux propres à un état de repos, de préparation ou d'exécution motrice. Puis, dans un second temps, nous avons étudié le décodage directionnel durant ces phases de préparation ou d'exécution motrice.

**Exploration et amélioration des marqueurs de l'activité cérébrale** : pour comprendre ce qui caractérise ces états, nous avons extrait de l'activité neuronale une variété relativement large de marqueurs spectraux (principalement la puissance, la phase et le couplage phase-amplitude). De plus, nous avons fait varier de nombreux paramètres liés à ces attributs (taille et emplacement des fenêtres temporelles et fréquentielles, comparatif de différentes méthodologies propres à chaque *feature*...)

**Exploration des régions non-motrices** : les données intracrâniennes nous ont également permis d'explorer si des régions non-motrices peuvent discriminer certains états ou si leur association avec des régions motrices peut constituer un gain de performances.

**Optimisation des paramètres de *machine learning*** : de nombreux paramètres liés à la classification ont été pris en compte afin d'évaluer leur influence sur le décodage (choix et optimisation de l'algorithme de classification et de validation croisée, stratégie de *multi-features*, évaluation statistique...)

**Implémentation et mise à disposition d'un ensemble de méthodes** : enfin, l'essentiel des méthodes et des outils présentés et utilisés durant cette thèse ont été codé en Python puis mis à la disposition de tous sur un compte Github. Ces outils comprennent l'extraction de marqueurs, leur classification ainsi qu'un ensemble de fonction pour visualiser les résultats.

Tous ces différents paramètres ont permis d'avoir une compréhension assez fine des impacts méthodologiques mais ont aussi forcé chaque étude à être hautement dimensionnelle. Cette thèse permettra peut-être aux lecteurs, et je l'espère, de commencer son exploration avec une nombre plus restreint de dimensions.



# MÉTHODOLOGIE

Cette partie méthodologique sera divisée en deux grandes sous parties visant à présenter :

1. **L'extraction des features** : présentation des méthodes utilisées dans le cadre de l'extraction d'attributs issus de l'activité neuronale. De manière générale, nous avons étudiés des attributs spectraux comprenant :
  - Phase et puissance spectrale
  - Attributs de couplage
2. **Le machine learning** : présentation des principaux algorithmes testées dans le cadre du décodage de l'activité neuronale

## 4.1 EXTRACTION DES FEATURES

Comme nous l'avons décrit précédemment, l'objectif du décodage de l'activité neuronale est d'arriver à extraire des signaux cérébraux une information suffisamment pertinente pour pouvoir discriminer différents types de classes (exemple : mouvement vers la gauche Vs droite).

Tout les attributs testés dans le cadre de cette thèse sont des attributs spectraux, donc issus de bandes de fréquences. La plupart de ces outils partagent donc une partie méthodologique commune à savoir, le filtrage. De plus, la plupart sont extraits en utilisant la transformée d'Hilbert. Pour éviter une redondance à travers les attributs, nous allons tout d'abord introduire quelques pré-requis.

### 4.1.1 Pré-requis

#### 4.1.1.1 Filtrage

L'intégralité des filtrages dans cette thèse ont été effectués avec la fonction *eegfilt* (qui a ensuite été reproduite pour le passage à python). De plus, afin d'éviter tout phénomène de déphasage, la fonction *filtfilt* a été systématiquement utilisée afin que le filtre soit appliqué dans les deux sens. Si cette dernière fonctionnalité n'est pas forcément indispensable dans le cadre d'un calcul de puissance, elle est absolument nécessaire pour un calcul de couplage phase-amplitude .

L'ordre du filtre présenté au dessus dépend de la fréquence de filtrage. Il a systématiquement été calculé en utilisant la méthode décrite par [Bahramisharif et al. \(2013\)](#) :

$$FiltOrder = N_{cycle} \times f_s / f_{oi} \quad (4.1)$$

où  $f_s$  est la fréquence d'échantillonnage,  $f_{oi}$  est la fréquence d'intérêt et  $N_{cycle}$  est

un nombre de cycles définit par  $N_{cycle} = 3$  pour les oscillations lentes et  $N_{cycle} = 6$  pour les oscillations rapides.

#### 4.1.1.2 Transformée d'Hilbert

Transformée permettant de passer un signal temporel  $x(t)$  du domaine réel au domaine complexe. Le signal peut ensuite s'écrire  $x_H(t) = a(t)e^{j\phi(t)}$  où  $a(t)$  est l'amplitude et  $\phi(t)$ , la phase. Cette transformation est particulièrement exploitée car le module de  $x_H(t)$  permet de récupérer l'amplitude et la phase est obtenue en prenant l'angle de  $x_H(t)$ .

#### 4.1.1.3 Transformée en ondelettes

La transformée en ondelettes ([Tallon-Baudry et al., 1997](#), [Worrell et al., 2012](#)) permet de décomposer un signal dans le domaine temps-fréquence. La décomposition en ondelettes d'une fonction  $f$  est définie par :

$$f(a, b) = \int_{-\infty}^{\infty} f(x) \bar{\psi}_{a,b} dx \quad (4.2)$$

Où  $\psi$  est appelé ondelette mère dont la définition générale est donnée par  $\psi_{a,b} = \frac{1}{\sqrt{a}} \Psi(\frac{x-b}{a})$  où  $a$  est le facteur de dilatation et  $b$  le facteur de translation. Le choix de l'ondelette mère s'est porté sur l'ondelette de Morlet qui est très largement utilisée à travers la littérature et définie par :

$$w(t, f_0) = A e^{-t^2/2\sigma_t^2} e^{2i\pi f_0 t} \quad (4.3)$$

Où  $\sigma_f = 1/2\pi\sigma_t$  et  $A = (\sigma_t\sqrt{\pi})^{-1/2}$ . L'ondelette de Morlet est caractérisée par le ratio constant  $r = f_0/\sigma_f$  que nous avons fixé égale à 7 comme suggéré par [Tallon-Baudry et al. \(1997\)](#).

Cette décomposition peut être comparée à la transformée courte de Fourier qui décompose le signal en une somme de combinaisons linéaire de sinus et de cosinus mais part du principe qu'il existe une régularité dans le signal permettant une telle décomposition. La transformée en ondelettes résout plusieurs limitations :

- Elle permet d'obtenir l'énergie d'un signal dans le temps, ce qui permet une bien meilleure exploration des phénomènes.
- Le rapport constant  $r$  permet d'obtenir des ondelettes dont la résolution fréquentielle varie en fonction des fréquences et permet une meilleure coïncidence avec la définition des bandes physiologiques ([Bertrand et al., 1994](#))

Tout les attributs qui vont être maintenant présentés, utilisent les méthodes décrites ci-dessus.

#### 4.1.1.4 Évaluation statistique à base de permutations

Pour une distribution de permutations construite à partir de deux sous-ensembles  $A$  et  $B$  et comportant  $N$  observations et pour une valeur  $p$  prédéfinie, on pourra conclure que :

- $A > B$  si  $A$  est parmi les  $N - N \times p$  derniers échantillons ("One-tailed test upper tail")

- $A < B$  si  $A$  est parmi les  $N \times p$  premiers échantillons ("One-tailed test lower tail")
- $A \neg B$  si  $A$  est soit inférieur aux  $(N \times p)/2$  premiers échantillons soit supérieur aux  $(N - N \times p)/2$

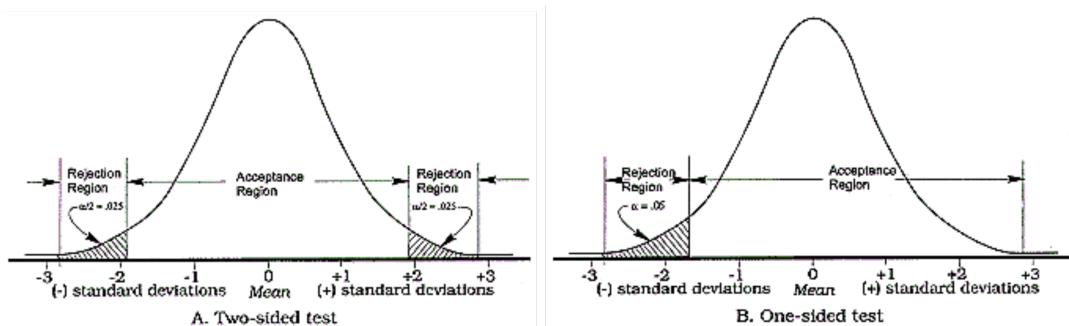


FIGURE 4.1 – Évaluation statistique à base de permutations, (A) "Two-tailed", (B) "One-tailed"

Grâce à cette méthode d'évaluation statistique, nous pourrons par exemple conclure si l'on a une augmentation, une diminution ou une différence statistique entre une valeur de puissance et la puissance contenue dans une période de baseline. Dernière précision, on comprend ainsi que pour obtenir une valeur  $p$  il faut que la taille de la distribution  $N$  soit au moins de  $1/p$ .

#### 4.1.1.5 Hyperplan

Un hyperplan est un espace de co-dimension 1. Donc, dans un espace  $3D$ , l'hyperplan est un plan (dimension  $2D + 1$ ). De manière générale, un espace de dimension  $N$  possède un hyperplan de dimension  $N - 1$

$$\dim_{\text{ESPACE}} = \dim_{\text{HYPERPLAN}} + 1 \quad (4.4)$$

#### 4.1.2 Puissance spectrale

##### 4.1.2.1 Méthodes explorées

Le calcul de la puissance spectrale a été approché par deux méthodologies et qui ont été utilisés à des fins différentes :

- La transformée d'Hilbert : souvent exploité dans le cadre du décodage ainsi que pour garder une uniformité entre les attributs de phase et couplage phase-amplitude basés eux aussi sur cette transformée.
- La transformée en ondelettes : principalement utilisée pour la visualisation des cartes temps-fréquence à cause de l'adaptation des ondelettes aux bandes physiologiques.

##### 4.1.2.2 Normalisation

On utilise la normalisation pour observer l'émergence d'un phénomène par rapport à une période définie comme baseline. A travers la littérature, quatre grands types de normalisation sont rencontrés :

1. Soustraction par la moyenne de la baseline
2. Division par la moyenne de la baseline
3. Soustraction puis division par la moyenne de la baseline

4. Z-score : soustraction de la moyenne puis division par la déviation de la baseline

La normalisation z-score est certainement la plus fréquemment rencontrée à travers la littérature. Le choix du type de normalisation dépend du type de données utilisées. Dans le cadre de nos données,  $\beta$ . était clairement la plus adaptée pour la visualisation. En revanche, dans le cadre de la classification, nous obtenions systématiquement de meilleurs résultats sans normalisation.

#### 4.1.2.3 Évaluation statistique

La fiabilité statistique de la puissance a été évaluée en comparant chaque valeur de puissance à la puissance contenue dans une période définie comme baseline. Pour ce faire, nous avons testé deux approches :

1. Permutations : les valeurs de puissance et de baseline sont aléatoirement mélangées à travers les essais. Puis, on normalise cette puissance. En répétant cet procédure  $N$  fois, on obtient une distribution qui peut ensuite être utilisée pour en déduire la valeur  $p$  de la véritable puissance (cf : *pré-requis*)
2. "Wilcoxon signed-rank test" : ordonne les distances entre les paires de puissances (vraie valeur, baseline) ([Demandt et al., 2012](#), [Rickert, 2005](#), [Waldert et al., 2008](#))

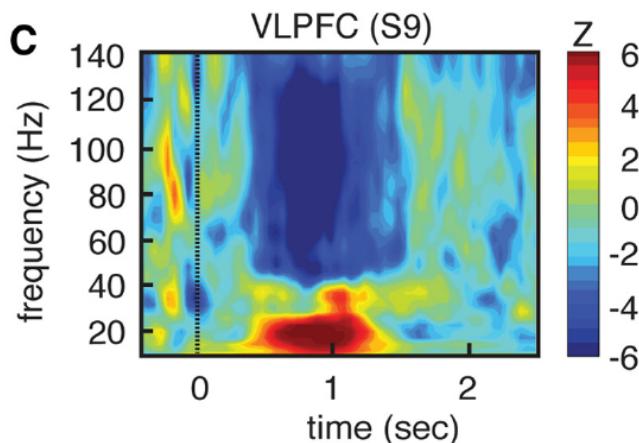


FIGURE 4.2 – Exemple de représentation temps-fréquence de puissance normalisées z-score ([Ossandon et al., 2011](#))

#### 4.1.3 Phase

L'extraction de la phase se fait de la même manière que pour le Couplage phase-amplitude , en prenant l'angle de la transformée d'Hilbert d'un signal filtré. La significativité peut être évaluée en utilisant le test de Rayleigh ([Jervis et al., 1983](#), [Tallon-Baudry et al., 1997](#)). Point de vue pratique, cela correspond à la fonction `circ_rtest` de la toolbox Matlab *CircStat* ([Berens and others, 2009](#))

#### 4.1.4 Phase-amplitude coupling

Le calcul du Phase-amplitude coupling ne se limite pas uniquement à la méthode. En réalité, pour obtenir une estimation fiable sur des données réelles, il est indispensable de suivre les trois étapes suivantes :

1. Estimation de la véritable valeur de PAC. Il existe plusieurs méthodes.
2. Calcul de "surrogates" : on va calculer des PAC déstructurés. Idem, il existe de nombreuses méthodes
3. Correction du véritable PAC par les "surrogates". Cette correction, qui est en fait une normalisation, aura pour but de soustraire à l'estimation du PAC de l'information considérée comme bruitée.

Les sous-parties suivantes présenteront de manières succinctes les principales méthodes rencontrées dans la littérature, ainsi que différents types de corrections applicables.

#### 4.1.4.1 Méthodologie du phase-amplitude coupling

Il existe une large variété de méthodes pour calculer le PAC, ce qui complique son exploration. Toutefois, il n'existe pas de consensus sur une méthode plus polyvalente qu'une autre, chacune possédant ses points forts et limitations. Pour aller un peu plus loin, et présenter quelques méthodes, il est nécessaire d'introduire quelques variables. Soit  $x(t)$ , une série temporelle de données de taille N. Pour cette série temporelle, on souhaite savoir si la phase extraite dans une bande de fréquence  $f_\phi = [f_{\phi_1}, f_{\phi_2}]$  est couplée avec l'amplitude contenue dans  $f_A = [f_{A_1}, f_{A_2}]$ . Pour cela, on va tout d'abord extraire  $x_\phi(t)$  et  $x_A(t)$  les signaux filtrés dans ces deux bandes. Enfin, la phase  $\phi(t)$  est obtenue en prenant l'angle de la transformée d'Hilbert de  $x_\phi(t)$  tandis que l'amplitude  $a(t)$  est obtenue en prenant le module de la transformée d'Hilbert de  $x_A(t)$ .

##### 1. Mean Vector Length-Modulation Index :

Cette méthode a été introduite par [Canolty et al. \(2006\)](#) et consiste à sommer, à travers le temps, le complexe formé de l'amplitude des hautes fréquences avec la phase des basses fréquences. L'équation est donnée par :

$$MVL = \left| \sum_{j=1}^N a(j) \times e^{j\phi(j)} \right| \quad (4.5)$$

##### 2. Kullback-Leibler divergence :

A l'origine, la divergence de Kullback-Leibler (KLD), qui est issue de la théorie de l'information, permet de mesurer les dissimilarités entre deux distributions de probabilités. Ainsi, pour pouvoir utiliser cette mesure dans le cadre du PAC, [Tort et al. \(2010\)](#) propose une solution élégante qui consiste à générer une distribution de densité probabilités de l'amplitude (DPA) en fonction des valeurs de phase et d'ensuite utiliser le KLD pour comparer cette distribution à la densité de probabilité d'une distribution uniforme (DPU). Plus la DPA s'éloigne de la DPU, plus le couplage entre l'amplitude et la phase est consistant.

Pour construire la DPA, l'astuce consiste à couper le cercle trigonométrique en N tranches (dans l'article il est proposé de couper en 18 tranches de  $20^\circ$ ). Puis, si on prend l'exemple de la tranche  $[0, 20^\circ]$ , on va chercher tout les instants temporels où la phase prend des valeurs comprises entre  $[0, 20^\circ]$  ( $t, \phi(t) \in [0, 20^\circ]$ ). On prend ensuite la moyenne de l'amplitude pour ces valeurs de  $t$  et on répète cette procédure pour chacune des tranches de phase.

On obtient ainsi la densité d'amplitudes en fonction des valeurs de phase. Il ne reste plus qu'à normaliser cette distribution par la somme des amplitudes à travers les tranches et on récupère une distribution de densité de probabilités. La figure 4.3 ([Tort et al., 2010](#)) présente un exemple de DPA en fonction de tranches de phase.

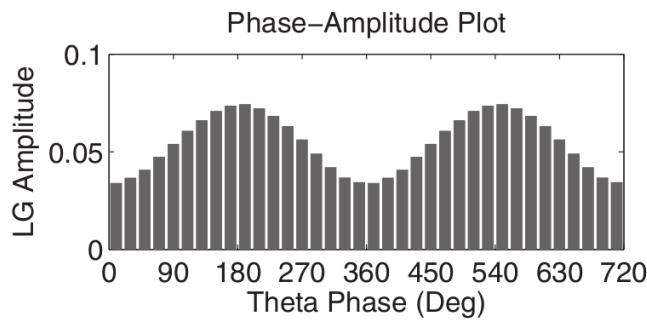


FIGURE 4.3 – *Densité de probabilité d'une distribution d'amplitudes en fonction de tranches de phases*

Le calcul de la divergence de Kullback-Leibler est ensuite appliqué pour mesurer les dissimilarités entre la DPA et la DPU et c'est cette mesure qui servira d'estimation du couplage phase-amplitude :

$$D_{KL}(P, Q) = \sum_{j=1}^N P(j) \times \log \frac{P(j)}{Q(j)} \quad (4.6)$$

où  $P(j)$  est la densité de probabilité de  $a(t)$  en fonction de  $\phi(t)$  et  $Q(j)$  est la densité de probabilité d'une distribution uniforme.

### 3. Height Ratio

La méthode du Height Ratio ([Lakatos, 2005](#)) est extrêmement proche du Kullback-Leibler divergence . En effet, l'amplitude sera binée de la même façon en fonction des tranches de phase. La mesure du PAC est ensuite donnée par :

$$hr = (f_{max} - f_{min}) / f_{max} \quad (4.7)$$

où  $f_{max}$  et  $f_{min}$  sont respectivement le maximum et le minimum de la densité de probabilité de l'amplitude en fonction des valeurs de phase.

### 4. Normalized Direct Phase-Amplitude Coupling

Le Normalized Direct Phase-Amplitude Coupling , qui n'est pas une des méthodes les plus fréquemment rencontrées, présente toutefois une avantage certain. En plus de fournir une estimation fiable du couplage phase-amplitude , [Ozkurt \(2012\)](#) démontre l'existence d'un seuil à partir duquel on peut considérer l'estimation du PAC comme étant statistiquement fiable. La beauté de cette méthode, c'est que ce seuil statistique, qui est une fonction de la valeur p désirée, ne dépend que de la taille de la série temporelle. Ce qui rend son utilisation particulièrement simple.

Pour estimer le PAC, une des hypothèses ayant permis d'aboutir à ce seuil

statistique est de devoir normaliser l'amplitude par un z-score dénotée  $\tilde{a}(t)$ . L'estimation du PAC est quasiment identique au MVL puisque c'est en réalité le carré de celle-ci. Enfin, pour une valeur p désirée, l'article introduit le seuil statistique :

$$x_{lim} = N \times [erf^{-1}(1 - p)]^2 \quad (4.8)$$

où  $erf^{-1}$  est la fonction d'erreur inverse. On déduira que l'estimation PAC est significative si et seulement si cette valeur est deux fois supérieure à ce seuil.

5. Autres méthodes : Tout les algorithmes présentés ci-dessus ont été testés, implémentés et comparés. En complément, voici une liste non exhaustive d'autres méthodes existantes :

- *Phase Locking Value (PLV)* ([Cohen, 2008, Penny et al., 2008](#)) : détournement du PLV proposé par [Lachaux et al. \(1999\)](#) qui mesure la synchronie de phase entre deux électrodes. Cette méthode va comparer la phase des basses fréquences avec la phase de l'amplitude des hautes-fréquences.
- *Generalized Linear Model (GLM)* ([Penny et al., 2008](#)) : outil décrit comme adapté aux données courtes et bruitées.
- *Generalized Morse Wavelets (GMW)* ([Nakhnikian et al., 2016](#)) : basée sur des ondelettes, semble particulièrement utile dans le cadre de l'exploration des données.
- *Oscillatory Triggered Coupling (OTC)* ([Dvorak and Fenton, 2014, Watrous et al., 2015](#)) : issue d'une détection de maximums des hautes fréquences.

#### 4.1.4.2 Correction du phase-amplitude coupling et évaluation statistique

Nous avons vu dans la section précédente différentes méthodes permettant de calculer un Couplage phase-amplitude . Toutefois, celui-ci peut être largement amélioré en faisant une estimation du PAC contenu dans le bruit des données. Une fois que cette estimation sera faite, on pourra retrancher ce PAC bruité à la valeur initiale. Tout comme il existe plusieurs méthodes de PAC, les équipes de recherche proposent à tour de rôle de nouvelles méthodes. Parmi elles, on peut citer :

- *Time-lag* : proposée par [Canolty et al. \(2006\)](#), on introduit un délai sur l'amplitude compris entre  $[f_s, N - f_s]$  où  $f_s$  est la fréquence d'échantillonnage et  $N$  est le nombre de points de la série temporelle
- *Shuffling des couples [phase,amplitude]* : ici, on mélange aléatoirement les essais de phase et d'amplitude ([Tort et al., 2010](#))
- *Swapping temporel d'amplitudes (ou de phase)* : on mélange aléatoirement les essais d'amplitude puis on recalcule le PAC avec la phase originale ([Bahramisharif et al., 2013, Lachaux et al., 1999, Penny et al., 2008, Yanagisawa et al., 2012b](#))

Ces trois méthodes produisent une distribution de *surrogates*. On pourra ensuite appliquer un z-score à la véritable estimation en utilisant la moyenne et la déviation de cette distribution. Enfin, l'évaluation statistique se fait également à partir de cette distribution (cf : *pré-requis*)

A ma connaissance, il n'existe pas de comparatif entre ces corrections et je n'ai jamais rencontré d'articles mentionnant que l'on ne puisse pas combiner les méthodes

de PAC avec les différentes corrections. En revanche, ce qui est relaté c'est que le *time-lag* nécessite des données longues dû à l'introduction de ce délai temporel.

#### 4.1.4.3 Comparatif des méthodes

[Penny et al. \(2008\)](#) ont comparé plusieurs méthodes dont le *MVL*, *PLV* et le *GLM* et [Tort et al. \(2010\)](#) ont complété cette étude avec d'autres méthodes (cf. A.3). Enfin, [Canolty and Knight \(2010\)](#) a fait une review qui comprend un descriptif très instructif.

#### 4.1.4.4 Représentation du phase-amplitude coupling

Comparée à la puissance, l'exploration du PAC peut s'avérer plus complexe dû à sa dimensionnalité plus grande. Il existe donc des outils et des méthodes destinées à simplifier cette exploration et à visualiser ces résultats.

Exemple concret, si on cherche à connaître les modulations de puissance contenue dans un signal, on peut représenter une carte temps-fréquence . Pour le PAC, idéalement on voudrait visualiser les phases, les amplitudes et le temps mais ces trois dimensions empêche une représentation simple. On peut donc avoir recours à différents types de représentations complémentaires :

- Puissance phase-locked : cette représentation permet de faire émerger l'existence d'un couplage, pour une phase donnée, et d'observer sa durée. Pour cela, on aligne les phases en détectant le pic le plus proche de l'instant temporel étudié. On calcule les cartes temps-fréquence que l'on va ensuite moyenner après les avoir recalées de la même façon que les phases (c'est-à-dire avec la même latence).
- Comodulogramme : pour une tranche temporelle définie, on représente les valeurs de PAC pour différentes valeurs de phase et d'amplitude

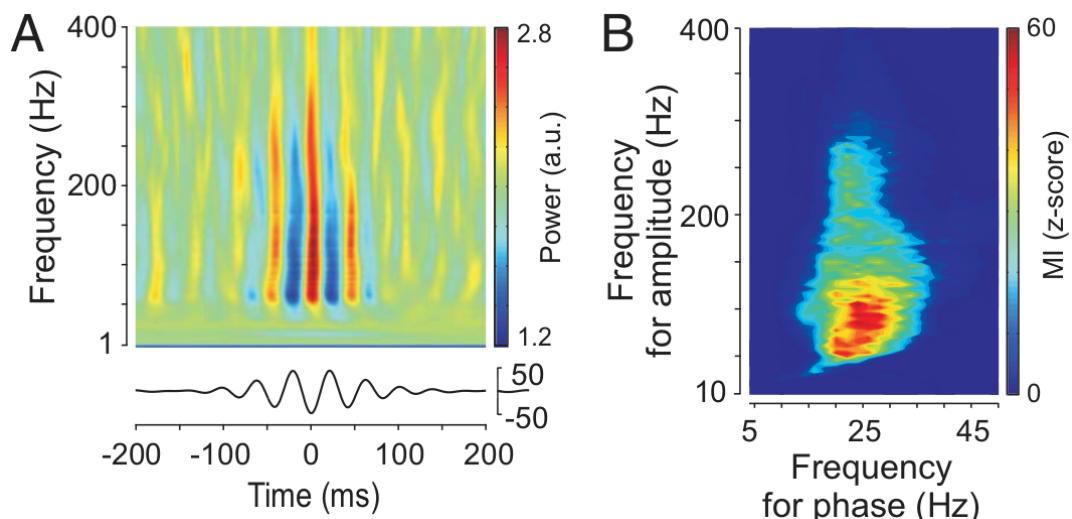


FIGURE 4.4 – (A) Exemple de cartes temps-fréquence phase locked sur le  $\beta$ , (B) Exemple de comodulogramme

La figure 4.4 (de [Hemptinne et al., 2013](#)) met en évidence que la représentation des cartes temps-fréquence phase-locked (A) est limitée d'une part, par la phase sur laquelle on choisit de recalier et d'autre part cette méthode est également limité par l'instant où l'on choisit de recalier. Pour la figure (B), le calcul du PAC se faisant à

travers la dimension temporelle, on a aucune idée de l'évolution du couplage dans le temps.

#### 4.1.4.5 Phase-amplitude coupling : résolution temporel ?

Voytek et al. (2013) introduit une méthode appelée *Event Related Phase-Amplitude Coupling* permettant de mesurer l'évolution temporelle du PAC. L'approche traditionnelle nécessite de connaître un nombre de cycles afin d'en déduire l'existence ou non du couplage, et donc perdre la dimension temps. L'article propose de calculer le PAC à travers les essais (ou répétitions). Pour un jeu de données de M essais de longueur N, on extrait respectivement les phases et les amplitudes  $\phi_M(t)$  et  $a_M(t)$  puis, pour chaque point temporel, on calcul la corrélation à travers les essais (corrélation linéaire-circulaire (Berens and others, 2009) qui se fait entre l'amplitude et des sinus/cosinus de la phase). Il en résulte une valeur de corrélation pour chaque instant et donc, de couplage.

Il est intéressant de noter que ce changement de positionnement dimensionnel est le même dans le cadre du calcul du *Phase-Locking Statistics* (PLS). Lachaux et al. (1999) propose de mesurer la synchronie de phase entre deux sites distants en calculant le PLS à travers les essais (dans ce cas, on conserve la dimension temporelle) ou alors à l'intérieur de chaque essais (Lachaux et al. (2000) - on perd la dimension temporelle mais on conserve la dimension essais ce qui permet d'envisager une classification avec le *single-trial PLS*)

#### 4.1.4.6 Phase préférentielle

La phase préférentielle correspondant à la phase où l'amplitude est maximum. De manière analogue au Kullback-Leibler divergence , l'amplitude est d'abord découpée en fonction de tranches de phases. La tranche de phase donnant une amplitude maximum est alors appelée *phase préférentielle* (PP). Cette mesure est particulièrement proche du Height Ratio , il serait intéressant d'étudier si l'un ou l'autre est porteuse d'une plus grande quantité d'informations.

Dans le cadre de cette thèse, nous avons entamé une exploration de l'utilité de la PP pour décoder des directions de mouvements, à la fois durant la préparation et l'exécution motrice. Nous avons obtenu des résultats faiblement significatifs et difficiles à remettre dans le contexte de la littérature. Pour cette raison, et par manque d'investigations supplémentaires, les résultats n'ont pas été inclus dans les papiers.

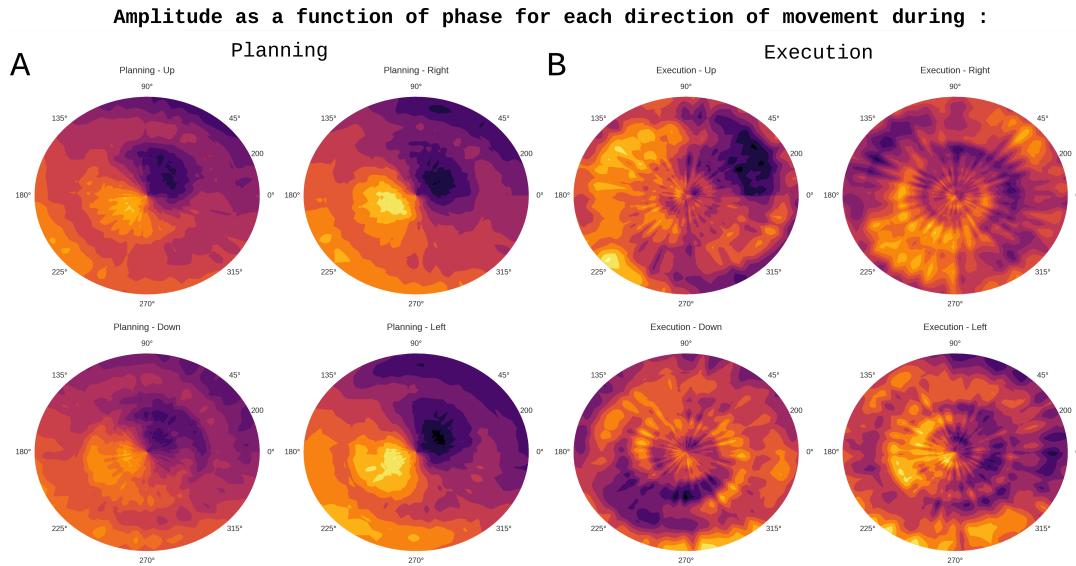


FIGURE 4.5 – Exemple de phase préférentielle - Chaque disque représente les modulations d'amplitude pour différentes valeurs de phase durant (A) une phase de préparation motrice et (B) d'exécution motrice pour quatre directions de mouvement (haut / bas / gauche / droite)

## 4.2 APPRENTISSAGE SUPERVISÉ

Le travail effectué durant cette thèse s'est exclusivement porté sur l'apprentissage supervisé. Celui-ci consiste à apprendre à la machine à reconnaître des événements qui ont été labellisé au préalable (cf. 4.2.1). A contrario, l'apprentissage non supervisé laisse la machine apprendre par elle-même. En pratique, l'apprentissage se fait sur des attributs. Par exemple, pour différencier des chats et des chiens, ou pourra utiliser l'angle formé par le sommet des oreilles. Les attributs doivent contenir une information pertinente permettant de différencier les classes. Enfin, les algorithmes de classification vont se servir de ces attributs pour définir une frontière entre les classes étudiées. A ce stade, il semble important de préciser que l'utilisation des outils d'apprentissage machine peut s'orienter (globalement) suivant deux axes :

1. Optimisation des attributs : on travail sur un raffinement des attributs afin que ceux-ci soient les plus performants possibles pour séparer les classes
2. Optimisation des paramètres de classification : on considère une base de données comme étant fixe, définitive, optimale et l'on va faire varier les différents paramètres liés à l'apprentissage machine (classificateurs, cross-validation...). C'est le cas des compétitions *BCI* où tout le monde travail sur une même base de données.

Bien sûr, ces deux axes peuvent être cumulés. Dans le cadre de cette thèse, le machine learning a été utilisé comme outil de validation d'hypothèses donc essentiellement porté sur l'optimisation des attributs. Le raffinement des paramètres de classification a également été étudié, mais, au final, il ne constitue pas la majeure partie de l'étude.

Un schéma classique d'analyse peut-être décrit par :

1. Labellisation des données
2. Constitution de données d'entraînement (*training*) et de test (*testing*)

3. Choix d'un classifieur puis entraînement de celui-ci sur les données *training*
  4. Test de ce classifieur entraîné sur les données *testing* et évaluation de la performance
  5. Évaluation statistique de cette acuité de décodage

#### 4.2.1 Labellisation et apprentissage

La labellisation c'est le fait d'associer à chaque événement l'appartenance à une classe ou à une condition. C'est par ce procédé que l'on va pouvoir apprendre ensuite au classifieur à identifier les classes. Par exemple, considérons *up* et *down* deux classes qui reflètent des mouvements de la main vers le haut ou vers le bas. On va donc construire un vecteur  $y_{direction}$  qui labellise chaque essais avec direction effectuée (ce vecteur peut aussi être booléen ou contenir des entiers. L'essentiel est que à chaque classe soit attribué une valeur qui lui est propre). Ce vecteur  $y$  est appelé *vecteur label*, qui vient labelliser chaque essais d'un vecteur d'attributs  $x$ .

$$y_{direction} = \begin{pmatrix} up \\ down \\ down \\ \vdots \\ up \end{pmatrix}, y_{bool} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ \vdots \\ 0 \end{pmatrix}, x = \begin{pmatrix} x_{trial_1} \\ x_{trial_2} \\ x_{trial_3} \\ \vdots \\ x_{trial_N} \end{pmatrix} \quad (4.9)$$

D'où le nom apprentissage supervisé. Finalement, l'apprentissage machine se fera grâce à ce vecteur label  $y$  et cette matrice d'attributs  $x$ . Ce qui nous amène directement aux notions de *training set* et de *testing set*.

x												
y	0	0	0	1	1	1	2	2	2	3	3	3

FIGURE 4.6 – Labellisation de données - pour un vecteur de données  $x$ , chacun des essais se voit attribuer une classe  $c_i \in \{0; 3\}$

#### 4.2.2 *Training, testing et validation-croisée*

Cette section est sans aucun doute la plus importante pour le machine learning puisque c'est elle qui assure la conformité méthodologique.

Un bon exemple pour comprendre cette partie est celui des contrôles de mathématiques. Avant l'examen, l'étudiant s'entraîne sur une série d'exercices. C'est la phase de *training*. D'ailleurs, plus il s'entraîne, plus ses chances de réussir à l'examen sont grandes. Le jour du contrôle, le professeur teste l'étudiant sur une série de nouveaux exercices en lien avec ce qu'il a étudié. C'est le *testing*. Ici, c'est un test parfait puisque l'étudiant est naïf sur le contenu de l'examen ce qui veut dire que l'on teste ses capacités mathématiques pures. Toutefois, il peut arriver durant la scolarité que l'on soit testé sur des exercices que l'on a déjà vu dans la phase de *training*. Dans ce cas, la moyenne des notes des étudiants est généralement beaucoup plus élevée puisque l'on ne teste plus des capacités mathématiques, mais la capacité à restituer un apprentissage.

#### 4.2.2.1 *Training set, testing set et naïveté*

Pour en revenir à la question du machine learning, on définit une partie des données pour entraîner la machine. Ensuite, on teste cette machine entraînée sur un

nouveau jeu de données de test. Il est essentiel d'avoir une séparation stricte entre des données définies comme *training* et des données de *testing* afin d'assurer la naïveté du classifieur. Même si cela peut paraître évident, nous verrons que ça n'est pas toujours aussi facile que ça.

Se pose maintenant la question de comment l'on choisit de couper les données en *training* et *testing*. Une méthode serait de prendre une partie des données de manière aléatoire, de la définir comme *training* et sur tester sur les données restantes. Toutefois, ce choix ne représenterait qu'une partie des données. Une méthode plus exhaustive et plus rigoureuse consiste à utiliser une validation-croisée (ou cross-validation).

#### 4.2.2.2 Validation-croisée

La validation-croisée (CV) est une procédure permettant de séparer les données en *training* et *testing*. Pour comprendre comment cela fonctionne, prenons un ensemble composé de  $N$  échantillons. Il existe plusieurs types de CV mais de manière générale, toutes dérivent du même principe qui est la cross-validation k-Fold ([Efron and Tibshirani, 1994](#), [Kohavi and others, 1995](#)). On coupe les  $N$  échantillons en  $k$  paquets de tailles égales (ou proches). Ensuite, le classifieur est entraîné sur  $k - 1$  paquets puis on le teste sur le paquet restant. Cette procédure est ensuite appliquée  $k$  fois afin que chaque paquet passe au *testing*. On dira que la cross-validation est *stratified* si la proportion de classes représentées au sein de chaque dossier est approximativement uniforme à travers les folds. On pourra aussi rencontrer le terme *shuffle* si il y a un mélange supplémentaire. Tout cela nous emmène à des CV k-fold, k-fold stratified, k-fold shuffle ou encore k-fold stratified shuffle.

Concernant le nombre de folds, on rencontre en général 3 valeurs à travers la littérature : 3-folds, 5-folds ou 10-folds ([Latinne et al., 2001](#), [Yanagisawa et al., 2009](#), [Besserve et al., 2007](#), [Waldert et al., 2008](#)). Un cas particulier, mais si le nombre de folds  $k = N$ , ça revient à entraîner la machine sur  $N - 1$  échantillons tester sur celui qui a été isolé et on répète cette procédure  $N$  fois. C'est ce que l'on appelle le *Leave-One-Out*. Toutefois cette dernière possède une grande variance et peut conduire à des estimations non fiables ([Efron and Tibshirani, 1994](#), [Kohavi and others, 1995](#)). Un autre cas particulier, est celui du *Leave-p-Subject-Out* ([Vidaurre et al., 2009](#), [Lajnef et al., 2015](#)) qui consiste à entraîner sur  $p$  sujets et tester sur les sujets restants. Cette procédure est particulièrement exigeante puisqu'elle nécessite d'avoir une certaine reproductibilité entre les sujets. Cette validation-croisée est fréquente avec des données EEG mais impossible à mettre en œuvre pour la sEEG à cause de l'implantation unique de chaque sujet.

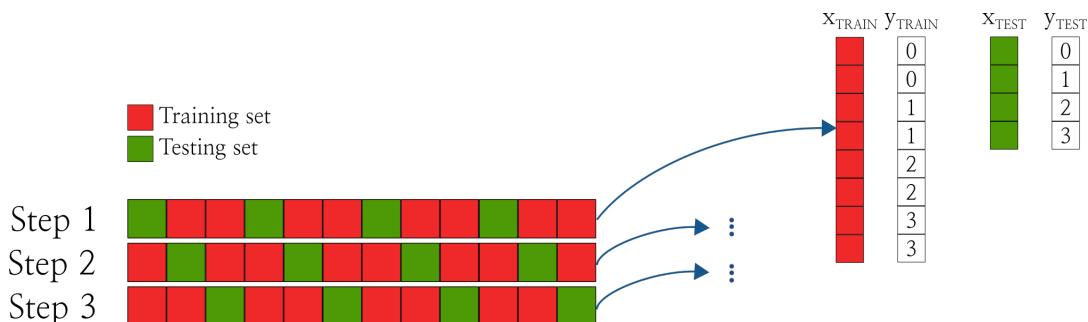


FIGURE 4.7 – Exemple d'une cross validation 3-folds

### 4.2.3 Classificateurs

#### 1. Linear Discriminant Analysis (LDA)

Le LDA ([Fisher, 1936](#)) est un classifieur linéaire. Pour un problème à deux classes, le LDA tente de trouver un hyperplan qui va maximiser la distance entre les classes tout en minimisant la variance inter-classes. Ce classifieur fait l'hypothèse que les données sont normalement distribuées avec la même co-variance. Un problème multi-classes pouvant être transformée en multiple bi-classes, le LDA tente de trouver un hyperplan séparant la classe du reste (*One-vs-All*)

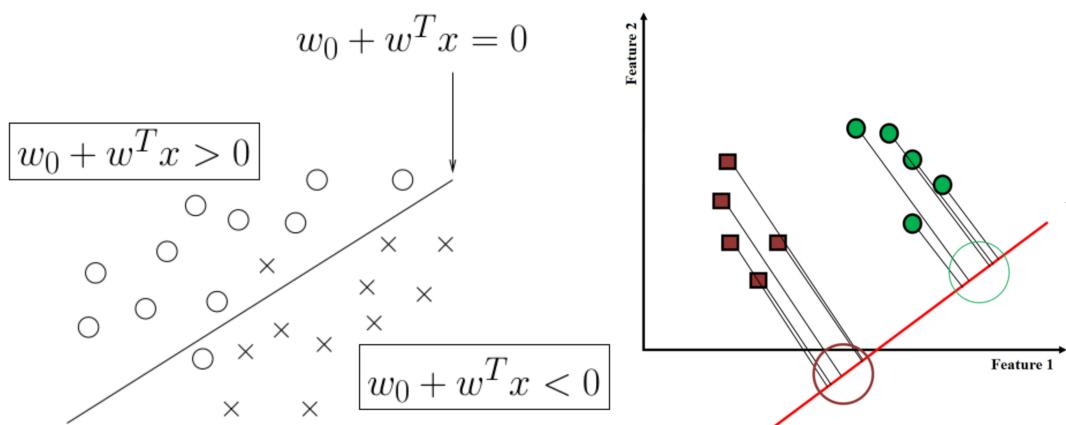


FIGURE 4.8 – Principe du Linear Discriminant Analysis ([Lotte et al., 2007](#), [Naseer and Hong, 2015](#))

#### 2. Support Vector Machine (SVM)

Le SVM ([Boser et al., 1992](#), [Cortes and Vapnik, 1995](#), [Vladimir and Vapnik, 1995](#)) utilise également un hyperplan pour séparer deux classes. Toutefois, cet hyperplan optimal est trouvé en maximisant les marges (ou distance) entre ce plan et les attributs les plus proches. Le SVM possède une particularité, il utilise un noyau qui peut permettre de résoudre les problèmes linéaire (*linear SVM*) mais également les problèmes non-linéaire en projetant les données dans un espace de dimension supérieure (*kernel trick*). Un noyau que l'on retrouve assez régulièrement est le *Radial Basis Function (RBF)* ([Burges, 1998](#)). Les problèmes multi-classes peuvent également être traités en *One-vs-All*. Pour une utilisation optimale du SVM, il est vivement conseillé d'utiliser la librairie C *libsvm* ainsi que de lire les pré-requis sur son utilisation et optimisation ([Chang and Lin, 2011](#)).

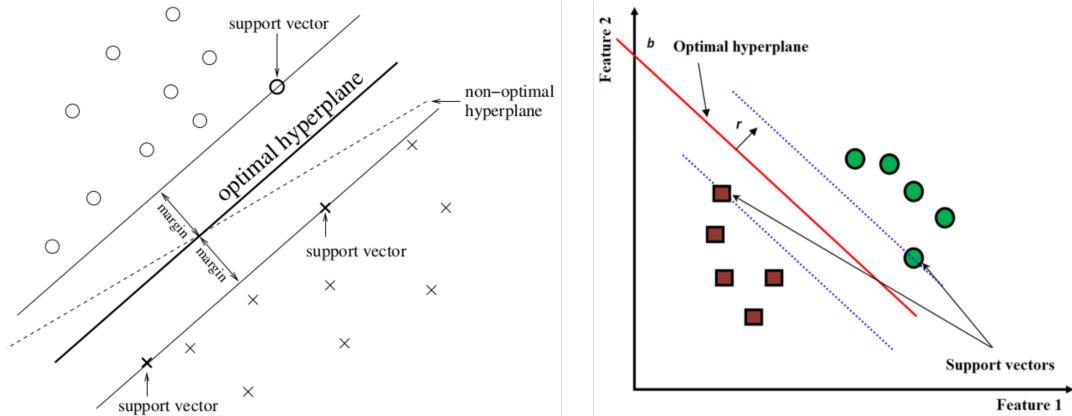


FIGURE 4.9 – Principe du Support Vector Machine ([Lotte et al., 2007](#), [Naseer and Hong, 2015](#))

### 3. k-Nearest Neighbor (KNN)

Pour un nouveau point de testing, le KNN ([Fix and Hodges Jr, 1951](#)) mesure la distance avec les  $k$  plus proches voisins et déduit la classe de ce point en fonction des classes de ces  $k$ -voisins (l'attribution de la classe se fait donc par vote)

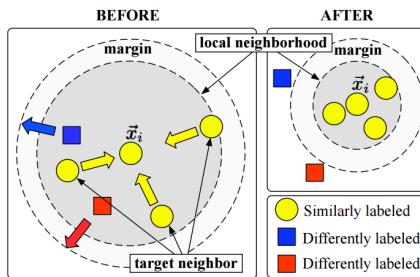


FIGURE 4.10 – Principe du k-Nearest Neighbor ([Weinberger et al., 2005](#))

### 4. Naive Bayes (NB)

Le NB ([Fukunaga, 1990](#)) est un classifieur probabiliste. Une des hypothèses du NB est que les données dans les classes doivent être normalement distribuées et indépendante.

La figure A.5 en annexe, issue de l'excellentissime librairie python scikit-learn dédiée au machine learning, illustre le comportement de chaque classifieur face à trois types de données. D'autres informations détaillées à propos des classificateurs peuvent être trouvées dans [Lotte et al. \(2007\)](#), [Wieland and Pittore \(2014\)](#), [Wu et al. \(2008\)](#)

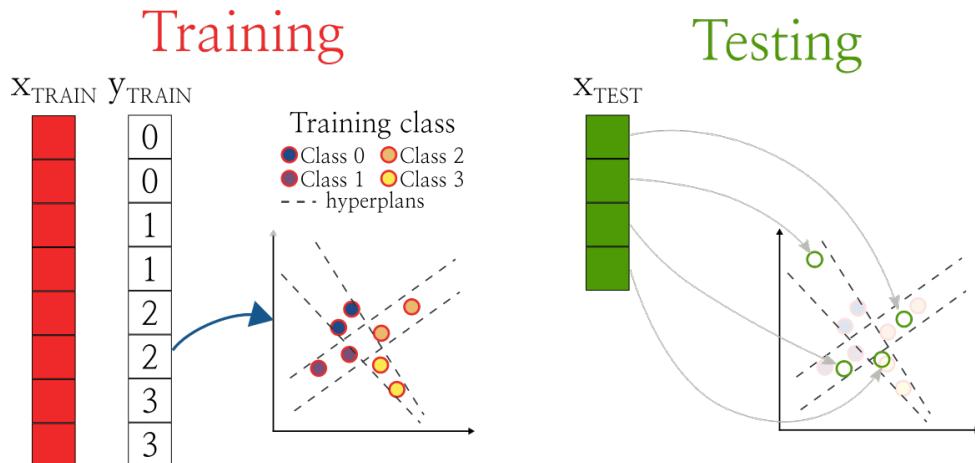


FIGURE 4.11 – Entraînement puis test d'un classifieur linéaire

#### 4.2.4 Évaluation de la performance de décodage

La question qui se pose maintenant, c'est comment évaluer la performance de décodage. Pour cela, on peut par exemple utiliser le *Decoding accuracy* ou le *roc*

##### 1. Decoding accuracy (DA)

L'utilisation (DA) est ce que l'on retrouve le plus fréquemment. Le calcul est simple, on compare les véritables labels avec les labels prédits par le classifieur. En faisant la somme des labels correctement prédits divisé par le nombre d'essais, on obtient un ratio qui correspond à l'acuité de décodage. Le plus souvent, ce ratio est ensuite exprimé en pourcentage. Le taux d'erreurs peut-être calculé en prenant  $1 - DA$ .

$y_{TRUE}$	$y_{PREDICTED}$
0	0
1	2
2	2
3	3
0	0
1	0
2	2
3	2
0	0
1	2
2	0
3	3

$8/12 \Rightarrow 75\%$

FIGURE 4.12 – Calcul de l'acuité de décodage

##### 2. Receiver operating characteristic (ROC)

Une autre méthode pour évaluer la performance de décodage est l'utilisation de l'aire sous la courbe (AUC) ROC ([Ling et al., 2003](#), [Huang and Ling, 2005](#), [Bradley, 1997](#)). Celle-ci prend en compte le nombre d'essais correctement et incorrectement classifiés et pourrait donc prendre davantage de valeur possible comparé au Decoding accuracy .

### 4.2.5 Seuil de chance et évaluation statistique de la performance de décodage

De manière théorique, le seuil de chance est donné par  $1/c$  où  $c$  est le nombre de classes. Par exemple, un problème à quatre classes donne un seuil de chance de 25%. Toutefois, ce seuil de chance est atteint pour un nombre de sample  $n$  infinis. En pratique, nous travaillons avec un nombre réduis de données, parfois même, avec très peu de sample. Dans ce cas, on peut obtenir des DA très élevés qui pourtant, ne sont pas pertinents. Les méthodes présentées ci-dessous ont pour but de trouver le seuil de chance associé à un jeu de donnée et de trouver pas la même occasion, la valeur  $p$ .

#### 1. Loi binomiale

En faisant l'hypothèse que l'erreur de classification suit une distribution binomiale cumulative, on peut utiliser la loi suivante pour en déduire la probabilité de prédire au moins  $z$  fois la classe  $c$  :

$$P(z) = \sum_{i=z}^n \binom{n}{i} \times \left(\frac{1}{c}\right)^i \times \left(\frac{c-1}{c}\right)^{n-1} \quad (4.10)$$

#### 2. Permutation

Les permutations présentent l'avantage d'être calculées à partir des données (*data driven*). [Ojala and Garriga \(2010\)](#) nous renseigne sur les différents types de permutations possibles dans le cadre du décodage :

- (a) *Full permutation* : les données sont mélangés
- (b) *Shuffle y* : le vecteur de label est mélangé. C'est la procédure la plus fréquemment rencontrée.
- (c) *Intra-class shuffle* : les données sont mélangées à travers la dimension *features* (colonne) et ce, à l'intérieur de chaque classe.

Autant les méthodes (a) et (b) nous renseigne véritablement sur la consistance d'un décodage par rapport aux données, autant la méthode (c) donne des informations un peu différentes. En effet, en cas de décodage non-significatif, on pourra soit conclure qu'il n'y a pas de consistance dans les attributs à l'intérieur des classes, soit que le classifieur est incapable d'utiliser cette l'interdépendance. [Ojala and Garriga \(2010\)](#) précise que dans ce cas, il n'est pas nécessaire d'utiliser un classifieur compliqué et qu'un classifieur simple devrait suffire.

Cette partie a fait l'objet d'une publication scientifique (cf. ?? - ([Combrisson and Jerbi, 2015](#))).

Un pipeline standard de classification est proposé en annexe (cf. [A.4](#)).

### 4.2.6 Du single au multi-features

Dans les sections précédentes, nous avons vu comment extraire des attributs de l'activité neuronale et comment les classifier. C'est ce que l'on appelle le *single feature* (SF), c'est-à-dire que l'on évalue la performance de chaque attribut séparément. Cette approche permet de constituer un set de features pertinents et répond à des questions neuro-scientifique. Cette démarche de SF a donc un but exploratoire.

La question que l'on peut maintenant se poser, c'est quelle performance de décodage puis-je obtenir si je combine ces attributs et dans quel cas est-ce utile? C'est le multi-attributs (ou *multi-features* (MF)). Tout d'abord, le MF est utilisé lorsqu'il y a

soit un désir soit un besoin de performances accrue. Par exemple, on utilisera le MF dans les compétitions de décodage ou tout simplement, pour une BCI où la performance est essentielle. Si l'on construit un système de bras robotisé piloté par activité neuronale, on comprend sans peine que celui-ci doit être le plus efficace possible et donc, le MF s'impose. Le dernier cas où l'on rencontre du MF, et ce n'est pas le cas le plus glorieux, c'est le cas où il y a un besoin de pallier à des résultats de SF assez faibles. La littérature expose des Decoding accuracy toujours plus hauts, des méthodes toujours plus complexes et donc, pour publier correctement un article, il faut avoir des résultats au-moins aussi perspicaces.

Le multi-features c'est donc l'utilisation de multiples attributs pour aboutir à une classification et ce, sans sélection particulière. Individuellement, les attributs d'un même set n'auront pas la même performance. Certains seront des bons marqueurs et d'autres, n'ajouteront pas ou peu d'information. Donc en combinant ces features, il est probable que l'acuité de décodage soit moins bonne que la performance en attribut unique. Pour cela, on pourra donc utiliser des algorithmes de sélections de marqueurs (*feature selection*). Le but de cette sélection est de trouver dans un set d'attributs, un sous-ensemble dont la performance groupée est meilleure que la performance individuelle.

Cette sélection est une procédure exigeante où le risque de sur-apprentissage est grand. C'est la raison pour laquelle cette sélection doit être mise à l'intérieur d'une cross-validation . Donc on définit un set de *training* et de *testing* grâce à la validation croisée, puis sur le *training* , on lance la *feature selection*. On aboutit à un sous-ensemble de marqueurs qui va servir à entraîner le classifieur. Ensuite, on sélectionne ce subset dans le *testing* et on test le classifieur avec ce subset. Toute ceci étant enfin répété pour chaque *fold* de la cross-validation . A la vue de cette procédure, deux problèmes émergent :

- La sélection d'attributs se faisant à l'intérieur des folds de la cross-validation , on peut très bien aboutir à des listes d'attributs différentes. Pour obtenir une information finale, on pourra donc parler des attributs les plus fréquemment choisis. Par exemple, si la sélection se fait dans un cross-validation 10-folds, on pourra dire que le feature 1 a été choisi 7/10, le feature 2, 3/10...
- En fonction de la sélection choisie et de la cross-validation , le pipeline complet peut être très (très) lourd et long.

Les mécanismes de *feature selection* peuvent être regroupés en deux grandes familles ([Guyon and Elisseeff, 2003](#), [Liu et al., 2008](#), [Das, 2001](#)) : les *Filter methods* et les *Wrapper methods*.

#### 4.2.6.1 Filter methods

Ces méthodes sont basées sur un critère et sont indépendantes du classifieur. Parmi elles, on retrouve des outils de corrélation, d'information mutuelle ou encore de statistiques. Ces derniers outils évaluent la contribution de chaque feature de manière indépendante sans tenir compte de la corrélation entre ces features. Pour résoudre ce problème, [Yu and Liu \(2004\)](#), [Ding and Peng \(2005\)](#) introduisent le *minimal-redundancy-maximal-relevance* qui en plus de trouver les features les plus pertinents, va permettre d'éliminer ceux qui sont redondants.

Pour terminer, ces méthodes sont effectivement indépendantes de l'algorithme de classification mais elles peuvent s'avérer optimales pour tel ou tel classifieur (ex :

l'utilisation du critère de Fisher pour filtrer les features est très performant lorsqu'il est ensuite associé au Linear Discriminant Analysis ([Duda et al., 2001](#))).

#### 4.2.6.2 Wrapper methods

Contrairement aux méthodes de filtrage, les *wrapper* utilisent le classifieur comme outil de sélection. Le premier inconvénient que l'on peut d'ors et déjà leur reprocher, c'est que le résultat final sera donc classifieur-dépendant, donc difficile pour la généralisation.

Parmi ces *Wrapper methods*, on peut citer :

1. Sélection exhaustive : on teste toutes les combinaisons de features possibles puis on sélectionne la meilleure. Procédure qui ne peut être faisable qu'en présence d'un jeu de données particulièrement restreint.
2. Sélection sur la statistique de décodage : on utilise le classifieur pour évaluer l'acuité de décodage de chaque feature séparément pour en déduire une valeur  $p$  (cf : [4.2.5](#)). Enfin, on sélectionne les features dont la valeur  $p$  est inférieur à un seuil désiré.
3. Sélection séquentielle : processus où l'on va ajouter/enlever des features de manière séquentielle jusqu'à atteindre un décodage optimal. Ce type de sélection se fait suivant deux directions :
  - (a) *Forward feature selection (FFS)* : la première étape consiste à évaluer la performance de chaque attributs. On sélectionne le meilleur que l'on va ensuite combiner en couple avec tout les features restant. On sélectionne le meilleur couple puis on teste les combinaisons des meilleures triplettes... On continu tant que la performance s'améliore. Si le DA d'une étape  $i$  est inférieur au DA de l'étape  $i - 1$ , on considère le nouveau subset de features à  $i - 1$ .

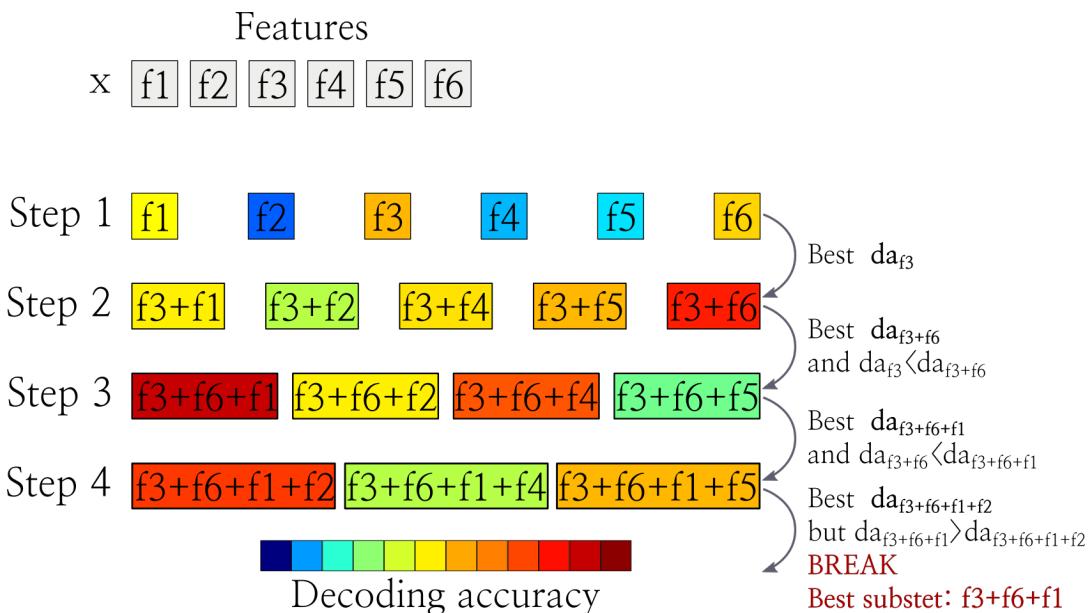


FIGURE 4.13 – Exemple d'une *Forward feature selection* appliquée sur six features

- (b) *Backward feature elimination (BFE)* : la philosophie est la même que pour un *forward*. On classifie d'abord les  $N$  features pris ensemble, puis on

enlève à tour de rôle chaque marqueur. On sélectionne le subset composé de  $N - 1$  features ayant fournit le meilleur résultat, puis on enlève de nouveau chaque feature... L'algorithme s'arrête de la même façon que le *forward*.

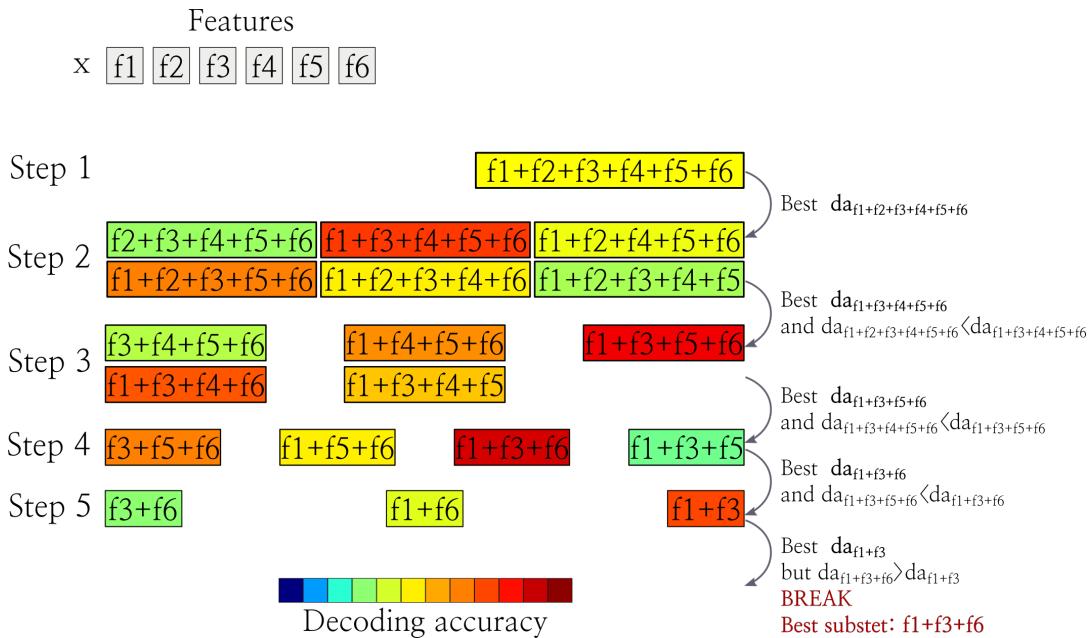


FIGURE 4.14 – Exemple d'une Backward feature elimination appliquée sur six features

De manière générale, il est rapporté que la FFS converge plus rapidement que la BFE ([Guyon and Elisseeff, 2003](#)). Toutefois, la FFS tombe plus facilement dans des minimums locaux et donc, mène à un décodage moins bon. En effet, la *forward* sélectionne pas-à-pas les meilleurs attributs, elle est donc moins ensembliste que la *backward*.

Les méthodes de filtrage demandent moins de ressources et représentent donc un premier choix pour les larges sets de données. En revanche, elles peuvent ne pas déceler les phénomènes de complémentarité entre features. Pour cette dernière raison, les méthodes de wrapper fournissent en général de meilleurs résultats ([Chai and Domeniconi, 2004](#)).

#### 4.2.7 Généralisation temporelle

L'introduction du *single-feature* faite plus haut était une présentation générique, c'est-à-dire que celle-ci est vraie quelque soit les features étudiés. On pourra donc classifier des attributs de puissance, de PAC, de phase, d'entropie... On peut également envisager l'étude un seul marqueur mais dans sa dimension temporelle. En effet, cela consiste à entraîner et tester un classifieur à différents instants temporels pour voir si le décodage varie dans le temps. Une des limitations de cette utilisation d'un classifieur est que, à chaque instant, celui-ci change. Donc on ne peut inférer aucune généralisation. Pour envisager une généralisation, il faut entraîner le classifieur à un instant puis le tester à travers toute la dimension temporelle restante. Dans ce cas, on pourra parler de généralisation mais reste encore le problème du

choix de l'instant temporel qui servira à entraîner le prédateur.

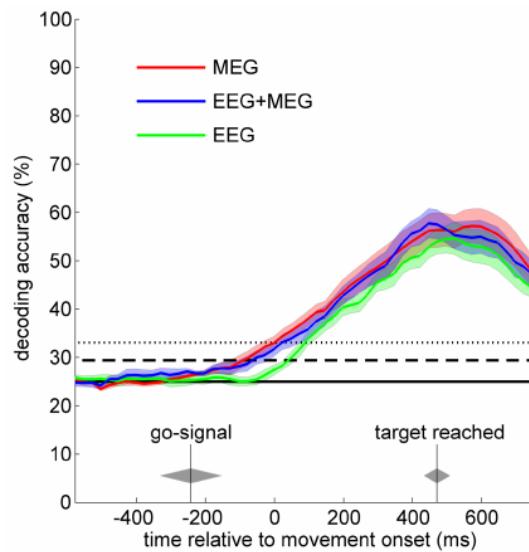


FIGURE 4.15 – Exemple de décodage temporel (Waldert et al., 2008). Ici, l'auteur décide 4-directions de mouvements de la main dans le temps. A chaque instant, un classifieur est créé, entraîné puis testé à ce même instant.

Pour répondre à cette question, King and Dehaene (2014) introduisent une idée particulièrement esthétique visant à généraliser le comportement d'un classifieur à travers le temps. Sans trop de surprise, ils ont nommé cette méthode la *généralisation temporelle*. Elle permet de répondre à deux limitations :

- Comment généraliser le comportement d'un classifieur lors d'une étude temporelle ?
- Comment choisir l'instant qui servira à entraîner le classifieur et quel impact ce choix aura-t-il sur le reste du décodage temporel ?

La méthodologie consiste à prendre un instant  $i$ , entraîner le classifieur et tester celui-ci sur tout les instants. Puis, on prend l'instant suivant  $i + 1$ , on entraîne un nouveau classifieur et on teste... Et on répète cette procédure pour tous les instants. On obtient ainsi une représentation 2D où, par convention, l'axe des ordonnées matérialise l'endroit où le classifieur a été entraîné (*Training time*) et l'axe des abscisses pour tester ce prédateur sur le reste de la dimension temporelle (*Generalization time*). La couleur permettra de signaler la performance de décodage.

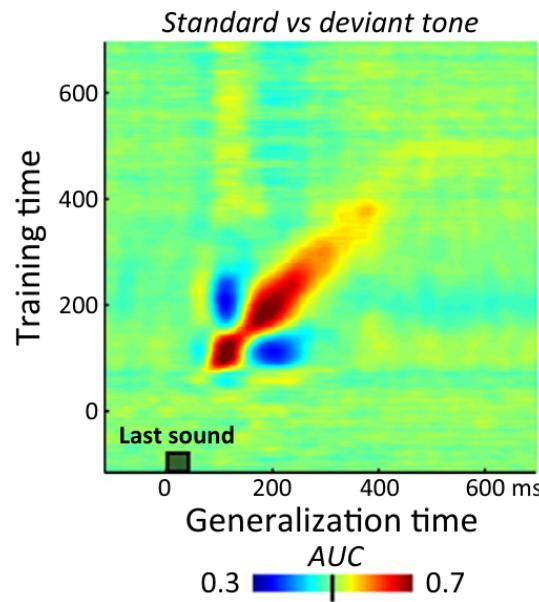


FIGURE 4.16 – Exemple de généralisation temporelle ([King and Dehaene, 2014](#))



# DÉVELOPPEMENTS INFORMATIQUES

A l'heure actuelle, il existe une vaste diversité de *toolboxs*/logiciels permettant d'analyser puis de visualiser des données neuro-scientifiques, que ce soit en *Matlab*, en *Python* ou tout autre langage. Parmi ces solutions, on pourrait citer *Brainstorm*, *FieldTrip*, *MNE python*, *Nipipe*, *ELAN*... Toutes sont développées depuis des années, par des équipes hautement qualifiées et expertes et jouissent d'une excellente réputation. Durant cette thèse, nous avons souhaité proposer des solutions informatiques pour les raisons suivantes :

**Maîtrise et compréhension des outils** : bien qu'il soit tout à fait envisageable d'analyser ligne par ligne ces *toolboxs*, le code peut parfois être assez dense et difficile à comprendre. Coder soi-même ses outils est une merveilleuse méthode pour les démystifier et surtout, pour les utiliser correctement c'est-à-dire connaître les avantages et les limites de chacun.

**Adaptation, amélioration et indépendance** : lorsque l'on choisit une *toolbox* on est limité aux possibilités et à la qualité d'implémentation de celle-ci. Il se peut que des besoins très spécifiques ne soient donc pas couverts (pour des données intracrânienne comme dans le cadre de cette thèse, il existe peu de solution). Pour ces raisons, le développement d'outils personnels permet une meilleure couverture des besoins spécifiques et assure une indépendance face aux limites d'une boîte à outils.

**Acquisition de compétences** : l'inconvénient majeur de l'implémentation d'outils est le temps, un temps qui est forcément pris sur autre chose. En revanche, c'est une somme de compétences non-négligeables.

**Identité, communication et communauté** : si les outils développés sont de qualité et forment un ensemble cohérent, une communauté d'utilisateurs peut se mettre en place ce qui peut contribuer à faire connaître une équipe ou un laboratoire.

## 5.1 CHOIX DU LANGAGE : PYTHON

Dans leur première version, les solutions informatiques ont été développées en *Matlab*. *Python* s'est imposé plus tard notamment grâce à son confort d'écriture et sa qualité syntaxique, l'abondance de documentations, d'utilisateurs et de modules. De plus, c'est une langue portable, pouvant être installé sur toute machine et tout système et surtout, *Open Source* distribuable à souhait. A noter que le langage *Julia* (<https://julialang.org/>) a également été testé. Ce langage se veut particulièrement prometteur puisqu'il promet une syntaxe élégante à l'instar de *Python* et des performances se rapprochant du *C*, devançant ainsi *Python* dans sa version non-optimisée. Il a toutefois été écarté, non à cause de ses performances mais parce que

c'est un langage encore récent et comportant un nombre plus réduit de modules que *Python* et une communauté plus petite.

## 5.2 PAQUETS DÉVELOPPÉS DURANT CETTE THÈSE

Trois paquets ont été développés pour proposer une solution cohérente de l'extraction des données à la visualisation des résultats :

**ipywksp** : workspace s'intégrant dans les notebooks *Jupyter*.

**brainpipe** : paquet permettant d'analyser des données (pré-traitements, extraction de features, classification et visualisation 2D)

**visbrain** : ensemble de modules destinés à des visualisations complexes et de hautes performances.

### 5.2.1 ipywks

Ce paquet est destiné aux utilisateurs venant de *Matlab* et souhaitant retrouver un workspace semblable. **ipywksp** mèle plusieurs langages (*Python*, *HTML* et *JavaScript*) et permet de visualiser le type, le contenu et la taille des variables, de les sauvegarder/charger et de les visualiser. Enfin, ce paquet

#### 5.2.1.1 Installation et utilisation

Dans un terminal, lancer :

```
git clone https://github.com/EtienneCmb/ipywksp.git
python setup.py install
```

Pour utiliser le workspace, lancer dans un notebook *Jupyter* :

```
# Chargement du module :
from ipywks import workspace

# Ouverture du workspace avec un thème noir et s'affichant automatiquement
# au survol de la souris :
workspace(theme="dark", autoHide=True)
```

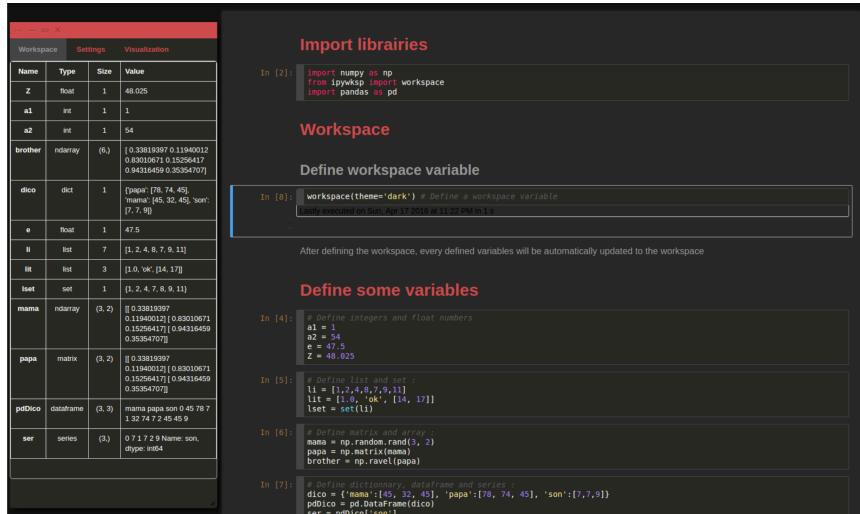


FIGURE 5.1 – *ipywksp* : Exemple de workspace pour *Jupyter*

### 5.2.2 Brainpipe

Ce paquet est destiné à l'analyse de données de tout type même si il est particulièrement adapté aux données intracrânienne. Il permet d'extraire un ensemble d'attributs, de les classifier, d'effectuer des analyses statistiques et de visualiser les résultats sur des graphes simples. Tout les résultats obtenus durant cette thèse ont été obtenu avec ce module et donc, toutes les méthodes y sont implémentées.

#### 5.2.2.1 Fonctionnalités

**5.2.2.1.1 Study :** Ce sous-module permet de gérer plusieurs études et plusieurs jeux de données, de gérer un très grands nombres de fichiers, de créer une arborescence de dossiers propre, une meilleure gestion des chemins d'accès ce qui est un atout majeur pour des collaborations.

```
# Importation des librairies :
import numpy as np
from brainpipe.system import study

# Création de deux études :
st = study('MEG')
st.add('~/Python/')
st = study('EEG')
st.add('~/Python/')

# Création et sauvegarde d'une variable dans le sous-dossier database :
x = np.random.rand(1000)
st.save('database', 'test.npy', x)
```

**5.2.2.1.2 Pré-traitements :** Ensemble d'outils pour pré-traiter les données, c'est-à-dire des outils de filtrage performants, la bipolarisation et la recherche des structures anatomiques associées à des coordonnées MNI/Talairach.

```
# Chargement des librairies :
from brainpipe.preprocessing import bipolarization, xyz2phy
from brainpipe.system import study

# Chargement de l'étude en cours :
st = study('CenterOut')

# Chargement des données à pré-traiter :
data, channels, xyz = st.load('database', 'centerout_data.npz')
# Où :
# - data : les données intracrâniennes
# - channels : nom des channels monopolaires
# - xyz : les coordonnées MNI des channels

# Bipolarisation et recherche des structures anatomiques :
data_bip, channels_bip, xyz_bip = bipolarization(data, channels, xyz=xyz)
phy = xyz2phy().get(xyz_bip, channels_bip)
```

**5.2.2.1.3 Attributs :** Brainpipe intègre une collection relativement importante de features calculables :

- Signal filtré
- Amplitude
- Puissance (hilbert, wavelet ou PSD)

- Phase Amplitude Coupling (nombreuses méthodologies / possibilité de générer des signaux synthétiques couplés)
- Phase-Locking Factor (PLF)
- Cartes temps-frequencies
- Phase-Locked Power (puissance alignées sur un cue)
- Event-Related Phase Amplitude Coupling (ERPAC)
- Phase préférentielle
- Phase Locking Value (PLV, soit à travers le temps, soit à travers les trials)
- Entropie spectrale

A noter que certains attributs intègrent un fenêtrage et le calcul dans des bandes de fréquences. De plus, tous ont été implémentés de façon matricielle et peuvent être calculés en parallèle pour un temps de calcul le plus réduit possible. Enfin, tous comprennent de nombreuses configuration possibles, intègrent le calcul de significativité et les outils de visualisation.

```
# Chargement des librairies :
from brainpipe.feature import *
import numpy as np
import matplotlib.pyplot as plt

sf = 1024 # Fréquence d'échantillonage

# On génère des données contenant un couplage entre 10 et 100hz :
data = cfcRndSignals(sf=sf, fPha=10, fAmp=100, ndatasets=10, noise=2, chi
                      =.5) [0].T
npts = data.shape[0]

# On génère des vecteurs phase et amplitude :
pVec, aVec, pha, amp = cfcVec()

# Calcul du PAC :
pacO = pac(sf, npts, pha_f=pha, amp_f=amp, Id='133')
xPac = pacO.get(data, data, matricial=True)[0]
xPac = np.squeeze(xPac) # Suppression des dimensions inutiles

# Plot du PAC avec les fonctions intégrées :
fig = plt.figure()
pacO.plot2D(fig, xPac.mean(-1), vmin=0, vmax=16, xvec=pVec, yvec=aVec,
            xlabel='Fréquence de phase (hz)',
            ylabel='Fréquence amplitude (hz)', title='Exemple de PAC',
            cmap='viridis', cblabel='Couplage PAC')
plt.show()
```

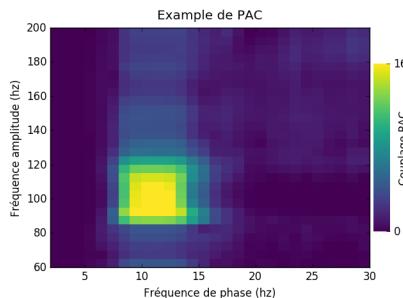


FIGURE 5.2 – Exemple de calcul PAC avec brainpipe

**5.2.2.1.4 Classification :** L'essentiel de la classification est assuré par *scikit-learn* (Pedregosa et al., 2011). Toutefois, brainpipe offre certaines fonctionnalités d'ordre pratiques qui, de manière non-exhaustive, peuvent être résumées à :

- Possibilité de définir une cross-validation et différents classifieurs et surtout, de pouvoir comparer leur performances de manière plus compactes.
- Adaptation de la classification aux données neuro-scientifiques, notamment en offrant un calcul en parallèle plus efficace car mieux adapté à nos petites données (en comparaison aux énormes banques de données d'images). De plus, de nombreuses études utilisent des classification particulière telles que le *Leave-p-Subject-Out*, présente dans brainpipe tout comme les cross-validation de type *10-times*....
- Généralisation temporelle (King and Dehaene, 2014)
- Calcul de la significativité des décodages plus synthétique (loi binomiale ou permutations)

Pour les neuro-scientifiques, brainpipe est un bon point d'entrée au monde de la classification puisqu'il permet de rapidement classifier nos données en un minimum de lignes et de manière lisible. Toutefois, pour une utilisation plus fine, un programme en *scikit-learn* pure reste moins limitatif.

**5.2.2.1.5 Statistiques :** En plus des statistiques calculés pour chaque attribut et pour la classification, brainpipe met à disposition un ensemble d'outils d'analyses statistiques. Calcul et gestion de permutations, correction multiple (Bonferroni, False Discovery Rate, Maximum statistic) ainsi que des outils pour les données circulaires (comme des données de phase).

**5.2.2.1.6 Visualisation :** Enfin, des fonctions pour visualiser des données ont également été ajoutées. Celles-ci permettent de créer des graphes 2D esthétiques et hautement configurables (plot de lignes avec déviation, ajout de valeur  $p$ , de lignes verticales/-horizontales, plot d'image, de contours...).

```
# Chargement des librairies :
from brainpipe.visual import BorderPlot
import numpy as np

# Définition d'un vecteur temps et de 10 sinusoïdales:
sf = 1024
t = np.mgrid[0:10, 0:1000] / sf
x = np.sin(2*np.pi*5*t) + .5 * np.random.rand(*t.shape)

# Plot du signal et de sa déviation :
BorderPlot(t[0, :], x.T, kind='std', xlabel='Temps', ylabel='Amplitude',
            title='Exemple de visualisation')
plt.show()
```

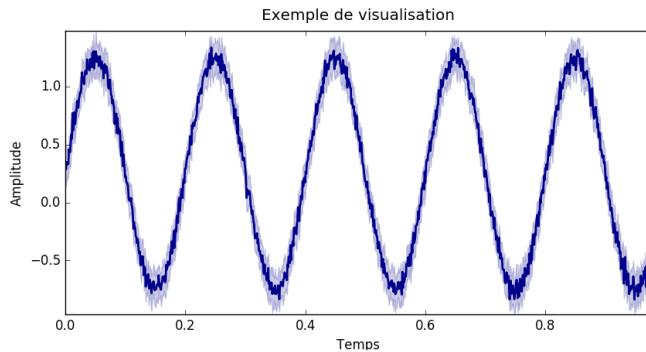


FIGURE 5.3 – Exemple de plot d'un signal et de sa déviation

### 5.2.2.2 Installation et documentation

Pour installer brainpipe, lancer dans un terminal :

```
git clone https://github.com/EtienneCmb/brainpipe.git
python setup.py install
```

Pour finir, une documentation complète est disponible en ligne <https://etiennecmb.github.io/brainpipe>

### 5.2.3 Visbrain

Visbrain est un paquet destiné à la visualisation de données neuro-scientifiques. Sa particularité réside dans le fait qu'il se base sur Vispy [Campagnola et al. \(2013\)](#) qui lui même utilise *OpenGL*. Les calculs sont envoyés sur la carte graphique ce qui, en conséquence, offre de très hautes performances en terme de fluidité et de temps de calcul. De plus, les interactions entre l'utilisateur et les différents modules se font via des interfaces graphiques (*Graphical User Interface, GUI*) construites à partir de *PyQt*.

#### 5.2.3.1 Présentation des modules

##### 5.2.3.1.1 Brain : *Brain* est une GUI avec un cerveau MNI dans lequel il est possible d'insérer des objets :

- **Sources** : dispositions de sources matérialisées par des sphères de couleur
- **Connectivité** : possibilité d'afficher des liens de connectivité entre ces sources
- **Structures** : ajout de structures 3D internes soit basées sur les aires de Brodmann soit sur l'AAL (*Automated Anatomical Labeling*)
- **Autres** : tout autre objet à trois dimensions peut-être rajouté par l'utilisateur.

Il n'y a aucune limite sur le nombre d'objets pouvant être ajoutés et ils peuvent tous être contrôlés indépendamment (couleur, transparence, taille, forme...). De plus, certains de ces objets peuvent interagir ensemble. Par exemple, l'activité des sources peuvent être projetées sur la surface du cerveau ou sur des structures internes.

Dernier point important, toutes les interactions possibles depuis l'interface graphique (et par raccourcis) sont également possibles en ligne de commande (Voir les [User functions](#) dans la documentation). Cette fonctionnalité est particulièrement utile pour produire un grand nombre de figures puisque tout peut être automatisé.

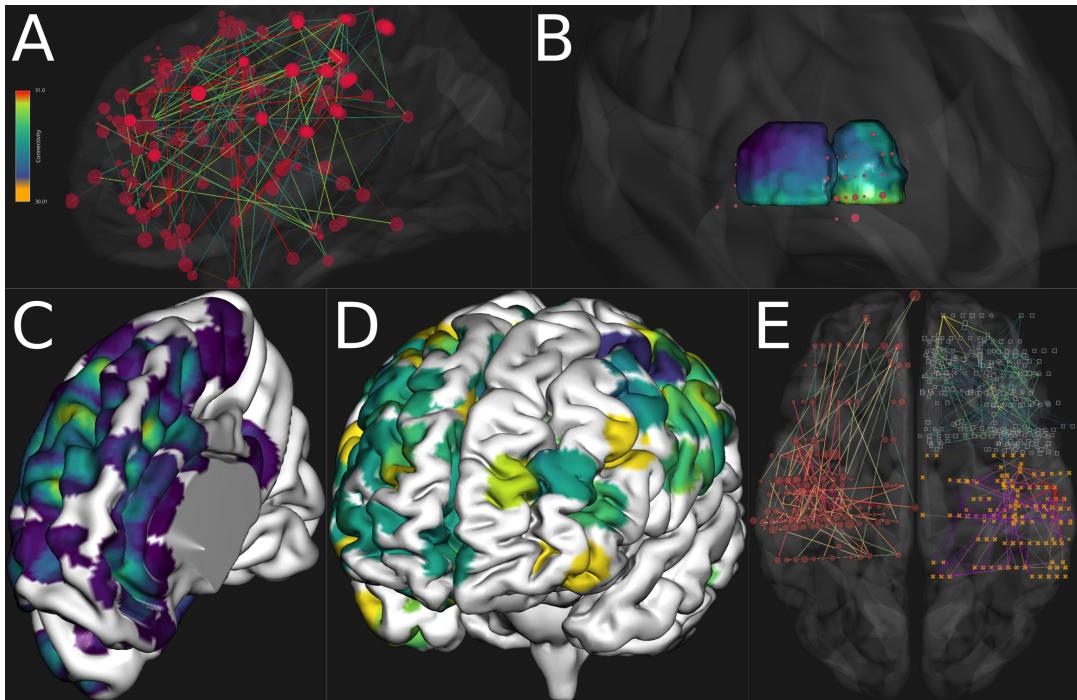


FIGURE 5.4 – Exemples des principales fonctionnalités de **Brain**, (A) Sites intracrâniens et connectivité, (B) Sources MEG et projection de leur puissance bêta sur le thalamus, (C) Nombre de sources contribuant à chaque point de l'hémisphère droit, (D) Projection de l'activité corticale de plusieurs sources intracrâniennes, (E) Exemple de scène complexe mêlant différents objets possédant chacun leur configuration

**5.2.3.1.2 Sleep :** *Sleep* est un module particulièrement performant pour visualiser, analyser et éditer des données de sommeil. Il a été développé en collaboration avec [Raphael Vallat](#).

Parmi les fonctionnalités principales, on peut citer :

- Chargement de fichiers \*.edf, \*.eeg (Brainvision, ELAN) et \*.trc
- Visualisation temporelle des données polysomnographiques (avec possibilité d'afficher/cacher les channels, contrôle des unités temporelles, de la taille de fenêtre, de l'amplitude...), en spectrogramme ou sous forme topographique
- Chargement/visualisation/édition/sauvegarde de l'hypnogramme
- Implémentation de détection de spindles, K-complexes, slow waves, rapid eye movements (REM), contraction musculaire ou encore de pic. Chaque détection peut être lancée sur les channels souhaités et des repères visuels sont ajoutés à l'hypnogramme
- Outils de traitement de signal (suppression des composantes linéaire et de moyenne, bipolarisation/re-référencement, outil de filtrage pour afficher le signal filtré, l'amplitude, la puissance ou encore la phase)
- Nombreux raccourcis pour une interaction la plus efficace possible.
- Une fonction d'*Auto scoring* est actuellement en cours de développement.

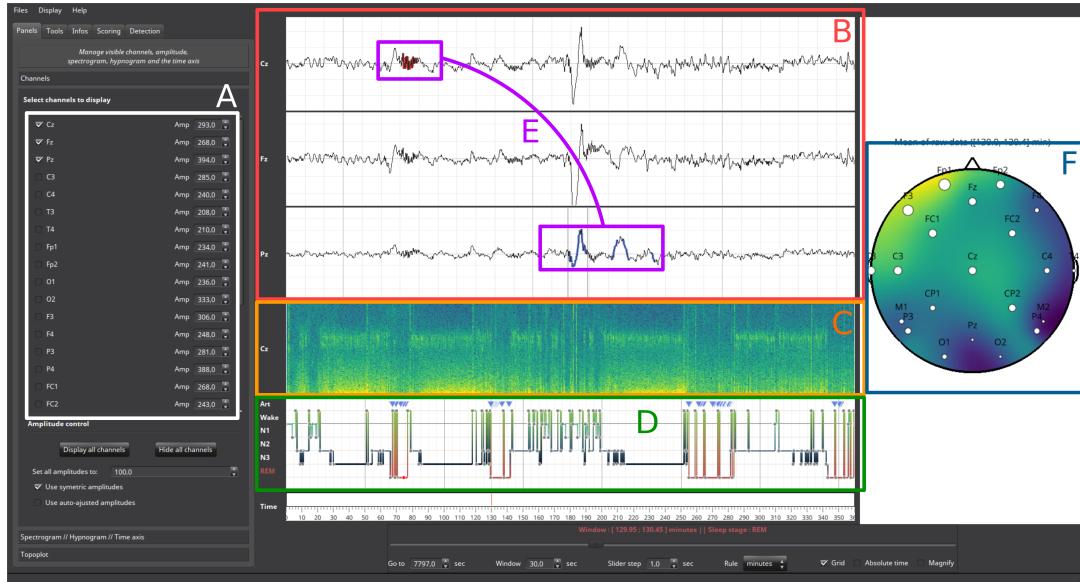


FIGURE 5.5 – Exemples des principales fonctionnalités de *Sleep*, (A) Possibilité de choisir les channels à afficher et contrôle indépendant des amplitudes, (B) Représentation temporelle des données polysomnographique, (C) Spectrogramme d'un channel, (D) Visualisation de l'hypnogramme et possibilité de l'édition, (E) Détections de spindle et de REM sur deux channels, (F) Exemple de représentation topographique (topoplot)

**5.2.3.1.3 Ndviz** Le module *Ndviz* a été conçu pour fouiller et explorer des données multi-dimensionnelles. Un des soucis majeurs des étudiants qui ne sont pas familiers avec la programmation est de se faire une image de ce que signifie une matrice et surtout, arriver à gérer les dimensions. Par exemple, des données organisées en  $(n_{channels}, n_{points}, n_{essais})$  offrent un certain nombre de visualisation possible à travers les dimensions : essais par essais par channel, la moyenne des essais par channel voir la moyenne à travers certains channels et essais... De plus, pour des données que l'on ne connaît pas, il peut être difficile de rechercher des artefacts, des activités épileptiques... *Ndviz* essaye de répondre à ses différentes problématiques en offrant différentes fonctionnalités :

- Dans tout *Ndviz* il est possible de sélectionner les dimensions à inspecter ce qui permet de se familiariser avec les matrices.
- Possibilité de visualiser plusieurs milliers de signaux disposés en grille en même temps. Cette fonctionnalité est issue d'un exemple de *Vipy* originalelement codé par [Cyrille Rossant](#). Par exemple, pour des données organisées en  $(n_{channels}, n_{points}, n_{essais})$ , il serait possible d'afficher une grille de  $n_{channels}$  lignes et  $n_{essais}$  colonnes et où sur chaque point de cette grille serait disposé un signal de  $n_{points}$  temporels. Cette fonctionnalité permet donc de visualiser des données comportant au maximum trois dimensions.
- De plus, en sélectionnant trois dimensions on peut aussi visualiser les données sous forme d'image (avec la couleur en guise de troisième dimension) ce qui pourrait par exemple être utile pour inspecter un grand nombre de carte temps-fréquence. Enfin, en sélectionnant deux dimensions, les données peuvent être représentées sous forme linéaire, en nuage de points, en histogramme ou spectrogramme.

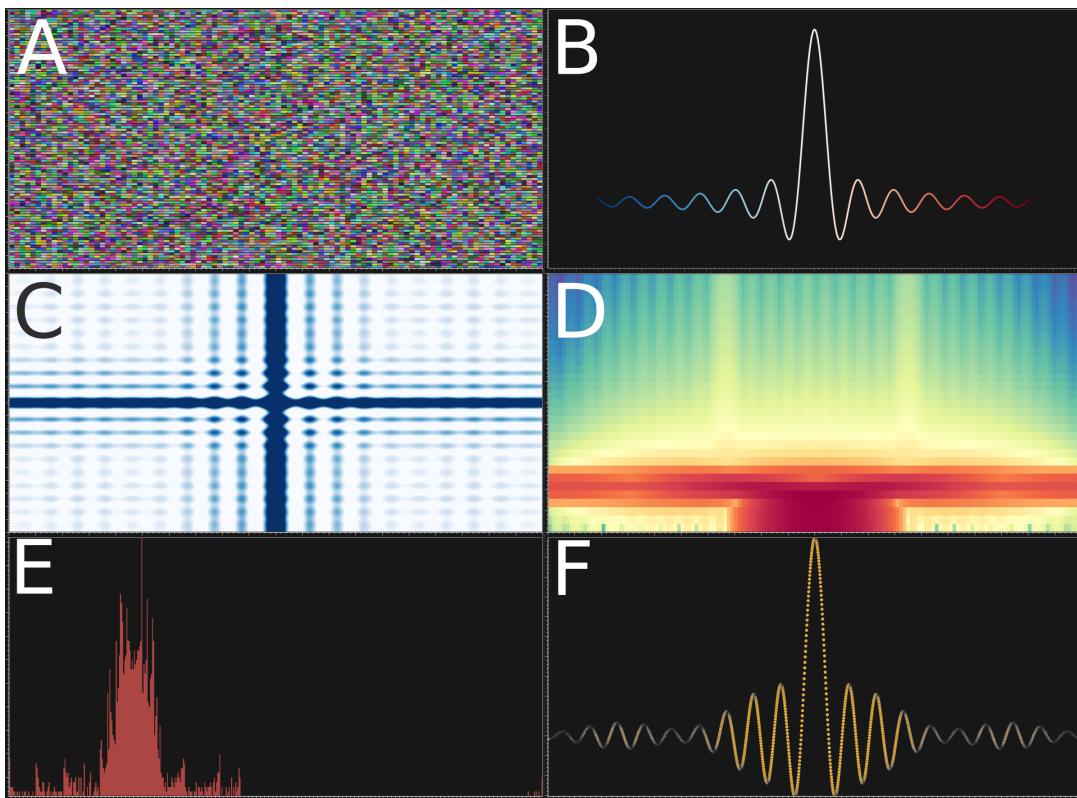


FIGURE 5.6 – Exemples des principales fonctionnalités de *Ndviz*, (A) Visualisation de 40000 signaux disposés dans une grille de (200, 200). Chaque signal fait plusieurs milliers de points et il est possible de zoomer sur chaque signal, (B) Représentation linéaire d'un signal, (C) Exemple de représentation sous forme d'image, (D-E) Calcul du spectrogramme et d'un histogramme d'un signal, (F) Représentation en nuage de points. Cette dernière représentations pourrait être utilisée pour inspecter des features

**5.2.3.1.4 Figure** Ce dernier module est le plus simple et certainement le plus utile. Il permet de faire des mises en page complexes de figures qui peuvent ensuite être exportées en haute définition et prête à être intégrée dans un papier. Il peut charger des images, les couper, les disposer en grille, ajouter des colorbars (soit pour chaque figure soit des colorbar communes à plus images), contrôler la couleur de l'arrière plan, ajouter des titres à tous les axes...

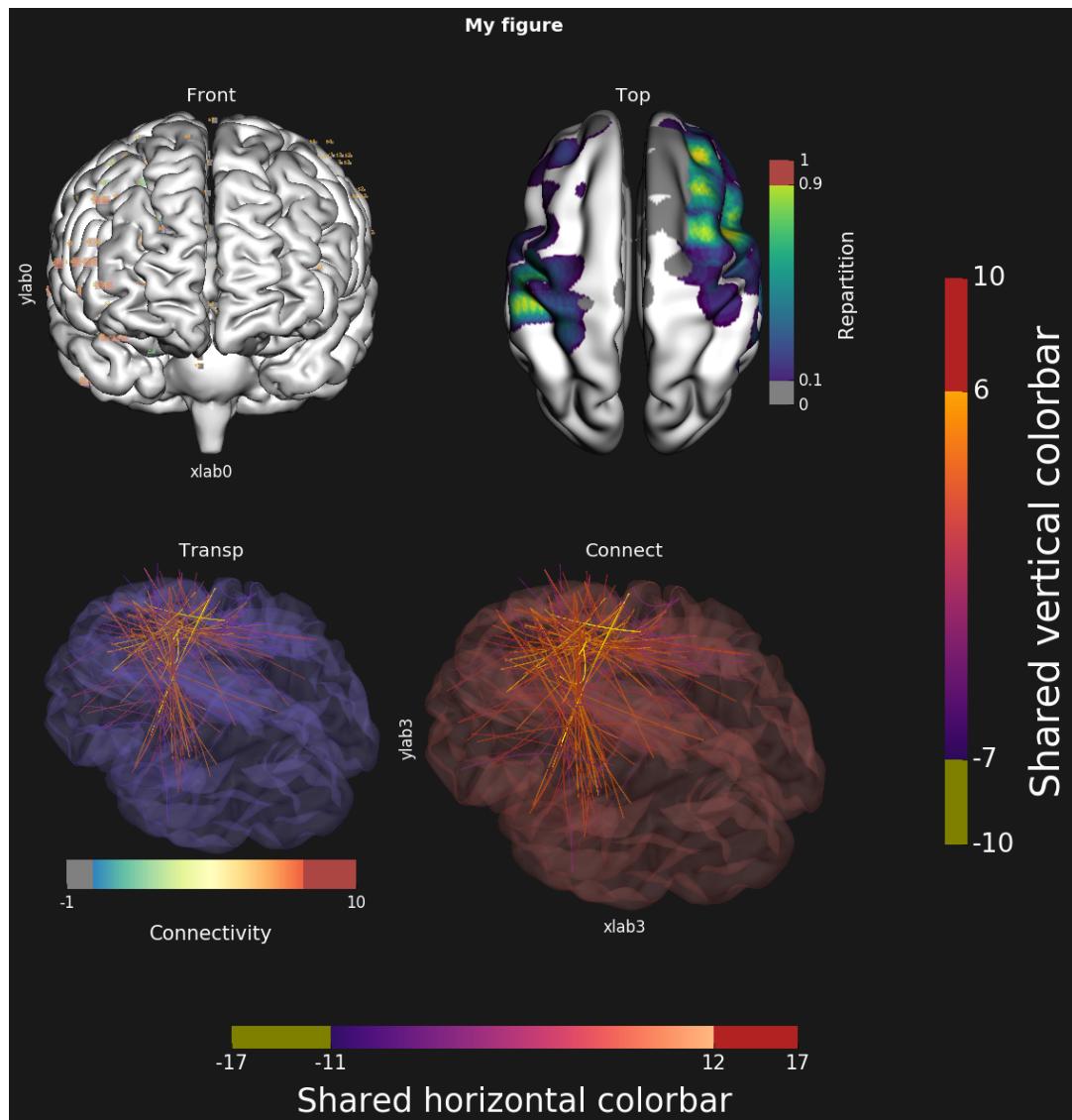


FIGURE 5.7 – Exemple de mise en page avec le module **Figure**

### 5.2.3.2 Installation et documentation

La procédure d’installation est plus complexe car elle possède plus de dépendances. Elle a donc été décrite plus largement dans la documentation <http://etiennecmb.github.io/visbrain/>. A noter que cette documentation décrit et illustre les fonctionnalités de chaque module et des exemples complets sont également mis à disposition <https://github.com/EtienneCmb/visbrain/tree/master/examples>

# DONNÉES EXPÉIMENTALES

Durant cette thèse, l'exploration s'est faite chez l'Homme par le biais, principalement, de données de type Stéréoélectroencéphalographie (SEEG). Ces données rares de très grande qualité (cf. 6.1.2) ont été acquise avant le début de la thèse, ce qui a permis de rentrer dans le vif du sujet très rapidement, après une période d'acclimatation aux différents traitements, propres à ce type d'enregistrement. D'autres types tel que l'EEG, la MEG ou les micro-électrodes ont également été approchés mais de manière ponctuelle, comme ce fût le cas dans l'Étude 1 (cf. ??) ou dans différentes collaborations. Toutefois, étant donné que le temps consacré à ces données ne représente qu'une faible portion du travail total, nous allons ici nous concentrer uniquement sur l'intra.

Pour commencer, nous verrons ce que l'analyse de la Stéréoélectroencéphalographie a de particulier (richesse des données, qualité, les traitements associés, les avantages et les limitations). Enfin, nous verrons concrètement les enregistrements qui ont été utilisés dans le cadre de cette thèse.

## 6.1 DONNÉES INTRACRÂNIENNES

### 6.1.1 Acquisition

La première question que l'on est en droit de se poser, c'est comment est-il possible de travailler, chez l'Homme, avec des enregistrements qui nécessitent une implantation invasive, c'est-à-dire dans le cortex? Certaines personnes présentent des formes agressives d'épilepsies, pouvant s'avérer pharmacorésistantes. En fonction de la localisation du foyer épileptogène, les méfaits engendrés par les décharges épileptiques peuvent être variés. Dans ce cas, il est nécessaire de localiser ce foyer avec, de préférence, des techniques non-invasives telles que l'EEG ou la MEG. Mais si ces dernières ne permettent pas une localisation précise le patient sera implanté avec des macro-électrodes comme la SEEG pour tenter de localiser puis d'enlever ce foyer par intervention chirurgicale. Cette implantation a un second objectif, déterminer quel est le rôle fonctionnel de la structure lésée (rôle moteur, langage, vision...). C'est dans ce contexte que les chercheurs proposent au patient de participer à une étude scientifique.

### 6.1.2 Avantages et limitations

Le paragraphe précédent met en exergue la rareté de ces données. De plus, ce type d'acquisition enregistre l'activité cérébrale d'une population relativement restreinte de neurones. En conséquence, on peut espérer que ce petit groupe s'active

dans des processus précis et ainsi, étudier des phénomènes fins. Enfin, le rapport signal sur bruit (RSB) de la SEEG est excellent, ce qui doit permettre l'étude de processus, même en essais unique là où d'autres types d'enregistrements auront besoin d'une large banque d'essais avant de pouvoir constater l'émergence d'un phénomène.

La SEEG présente toutefois quelques limitations que l'on peut nuancer. Le problème majeur est certainement la généralisation d'un phénomène ou la reproductibilité à travers les sujets. La pathologie est propre à chaque patient, donc son implantation aussi. Ce qui signifie qu'il n'y a aucune chance que plusieurs sujets présentent rigoureusement la même implantation. Pour contourner cette limitation, on pourra utiliser :

- Des régions d'intérêt (ROI) : on va regrouper les électrodes des différents sujets par "proximité" en faisant l'hypothèse que celles-ci s'activent de façon similaire face à un processus. Ces ROI pourront être par exemple les gyrus ou les aires de Brodmann. Bien sûr, ce que l'on gagne en généralisation, on le perd en précision.
- Projection corticale : autour de chaque électrode, on définit une sphère d'intérêt (généralement 10 millimètres de rayon) puis on prend l'intersection de ces sphères avec la surface du cerveau. Cette technique permet une visualisation des activités proches de la surface à travers les sujets mais on perd de la lisibilité sur ce qui se passe en profondeur.

Une utilisation combinée des ROI et de la projection corticale permet de palier, au moins partiellement, au problème de reproductibilité inter-sujets.

Autre limitation, on ne dispose que d'une couverture partielle puisque le neurochirurgien implante une quantité limitée d'électrodes. Ce dernier point est traité en augmentant le nombre de sujets. Enfin, la dernière limitation que l'on soulèvera ici, concerne le fait de travailler sur un cerveau "malade" empêchant donc une généralisation à des sujets sains. On limite ce problème par un ensemble de prétraitements ([Jerbi et al., 2009b](#)) décrits dans le prochain paragraphe.

Un dernier point que l'on peut argumenter à la fois comme avantage ou limitation, c'est de ne pas pouvoir contrôler l'implantation pour étudier un phénomène précis. Par exemple, si l'on analyse l'encodage moteur, on s'attendrait à concentrer les efforts sur le cortex moteur primaire ou pré-moteur. Or l'implantation SEEG peut très bien contenir du frontal, du pariétal, du temporal. Là où finalement on peut considérer ça comme un avantage, c'est que l'on a accès à un ensemble de structures jugées non-primordiales mais dont l'ajout pourrait permettre la compréhension d'un processus de manière plus globale.

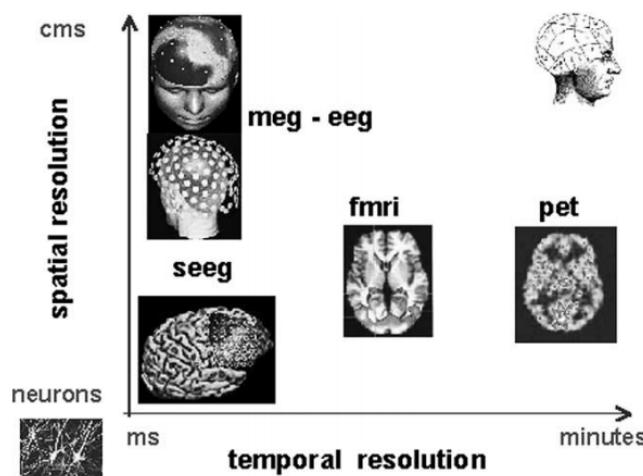


FIGURE 6.1 – Comparatif de résolution spatiale et temporelle pour différentes techniques d'imagerie (Lachaux et al., 2003). La SEEG offre à la fois une résolution spatiale équivalente à la PET ou fMRI et une résolution temporelle proche de celle de la MEG et de l'EEG ce qui en fait une technique de choix, sans compromis de résolution.

### 6.1.3 Inspection visuelle

BrainTV (Lachaux et al., 2007, Jerbi et al., 2009a)

### 6.1.4 Prétraitements

Le premier prétraitement appliqué a été une réjection des sites bruités ou présentant une activité pathologique, c'est-à-dire des décharges épileptiques. C'est par cette inspection manuelle que l'on augmente le potentiel de généralisation aux sujets sains.

Autre prétraitement, les données peuvent être bipolarisées comme c'est le cas dans de nombreuses études (Bastin et al., 2016, Ossandon et al., 2011, Jerbi et al., 2009b). La bipolarisation part du principe que deux sites proches enregistrent des activités neuronales différentes mais que toute source de bruit, ou influence de sources lointaines, se retrouvera sur ces deux sites. La technique de bipolarisation consiste donc à soustraire les activités neuronales de sites proches ce qui a pour effet de supprimer la partie commune, le bruit. Par exemple prenons une électrode contenant les sites k9, k10, k11 et k12. Après bipolarisation, on considérera les sites matérialisés par k10 – k9, k11 – k10 et k12 – k11. Les bénéfices de la bipolarisation peuvent être résumés par :

1. Limitation des influences des sources lointaines et de la tension secteur (50hz)
2. Augmentation de la spécificité qui, pour un site bipolarisé, est estimée à 3mm (Kahane et al., 2006, Lachaux et al., 2003, Jerbi et al., 2009b)

## 6.2 DONNÉES D'ÉTUDE

Trois jeux de données intracrâniaux ont été exploré :

1. *Center-out* : étude de l'encodage et du décodage des actions et des intentions motrices lors de mouvements de main
2. *Occulo* : étude des intentions et décision de mouvements oculaires
3. *Emotions* : étude de l'encodage des émotions

### 6.2.1 Données *Center-out*

C'est le jeu de données qui a été le plus largement exploité. En effet, celui-ci a servi à étudier l'encodage (cf. ??) et le décodage (cf. ??) des actions motrices chez l'homme.

#### 6.2.1.1 Descriptif des données

Six sujets (six femmes), implantés au département de l'épilepsie de l'hôpital de Grenoble ont donné leur consentement écrit pour passer l'expérience, sous la supervision du personnel médical. Le tableau 6.2.1.1 résume les détails clinique des différents sujets.

	Dominance	Age	Genre	Zone épileptique
P <sub>1</sub>	D	19	F	Frontal (RH)
P <sub>2</sub>	D	23	F	Frontal (LH)
P <sub>3</sub>	D	18	F	Frontal (RH)
P <sub>4</sub>	D	18	F	Frontal (RH)
P <sub>5</sub>	D	31	F	Insula (RH)
P <sub>6</sub>	D	24	F	Frontal (LH)
Moyenne : 22.17 ± 4.6				

FIGURE 6.2 – Détails cliniques des sujets ayant participé à la tâche *Center-out*

#### 6.2.1.2 Matériel d'acquisition

De 12 à 15 multi électrodes ont été implantées dans différentes structures. Chaque multi électrode possède entre 10 et 15 sites mesurant 0.8mm et séparés de 1.5mm. La localisation anatomique des électrodes s'est faite en utilisant le schéma d'implantation (exemple en annexe A.6) et l'atlas proportionnel de Talairach et Tournoux (Talairach and Tournoux, 1993). La visualisation de la pré-implantation s'est faite par *IRM – 3D* et un *CT – scan* a été utilisé pour la post-implantation. Enfin, un *IRM* a également servi pour visualiser les électrodes implantées dans la matière blanche. Les coordonnées Talairach ont été déduites du *CT – scan* puis ont été transformées en MNI afin de pouvoir les superposer dans un cerveau standard (cf. figure ci-dessous).

Le système Micromed a été utilisé pour visionner l'acquisition de l'activité neuronale. Une électrode prise dans la matière blanche a été prise comme référence et un filtrage *passband* entre [0.1, 200hz] a également été effectué *online*. La fréquence d'échantillonnage est de 1024hz.

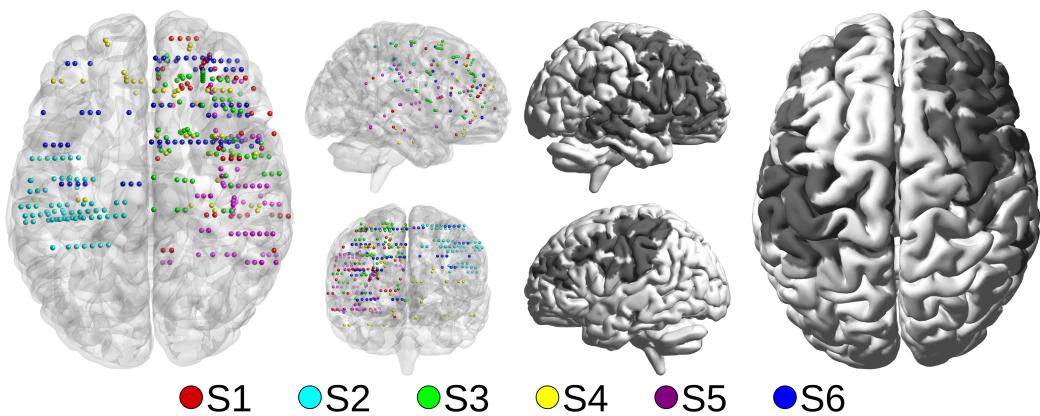


FIGURE 6.3 – Implantation intracrâniale et couverture corticale de six sujets épileptiques ayant passé la tâche Center-out

#### 6.2.1.3 Descriptif de la tâche

La tâche est composée de trois phases :

1. Phase de repos : on demande au sujet de rester immobile pendant une durée de une seconde
2. Phase de préparation motrice (**CUE 1**) : une direction est imposée à l'écran (haut/bas/gauche/droite). On demande au sujet de se préparer pendant 1.5s à bouger la souris dans la direction imposée.
3. Phase d'exécution motrice (**CUE 2**) : le sujet exécute le mouvement en bougeant la souris du centre à l'extrémité de l'écran indiquée (environ 1.5s) puis de revient vers le centre.

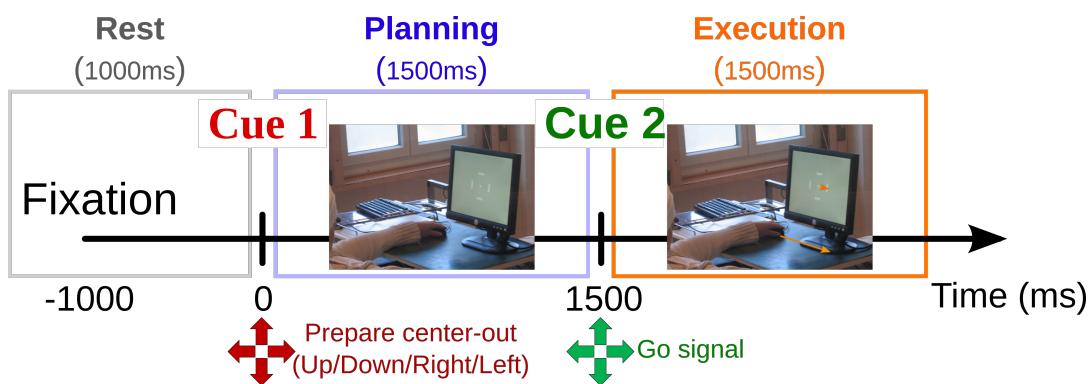


FIGURE 6.4 – Descriptif de la tâche Center-out

Cette tâche à conduit à deux études essentielles :

- L'étude de l'encodage des intentions motrices (cf. ??) : on étudiera le décodage du repos vs préparation, repos vs exécution et de la préparation vs exécution
- L'étude du décodage des directions de mouvement (cf. ??) que ce soit pendant la préparation ou pendant l'exécution motrice.

### 6.2.2 Autres données

- Occulo : Données occulo mais un seul sujet donc pas super cool. De plus, les résultats sont un peu vieux et mériteraient de se pencher dessus une fois pour toute.
- Emotions : Ce serait pas mal que j'ai quelques résultats sur les émotions, histoire de montrer aussi un peu la diversité des données utilisées.

## 6.3 DELAYED TASK : PROTOCOLE EXPÉRIMENTAL

okok

# DISCUSSION GÉNÉRALE

Enfin : la conclusion générale!!!

Au cours de ce mémoire, nous avons développé un modèle ...

## PERSPECTIVES

Dans la continuité directe de notre travail de thèse, nous pouvons ...

# A ANNEXES

## SOMMAIRE

A.1	CARTES DES INTERFACES CERVEAU-MACHINE ( <a href="#">GRAIMANN ET AL., 2009</a> )	77
A.2	JEUX DE DONNÉES EN LIBRE ACCÈS ( <i>BCI competition</i> ) . . . . .	78
A.3	COMPARATIF DE MÉTHODES PAC ( <a href="#">TORT ET AL., 2010</a> ) . . . . .	79
A.4	PIPELINE STANDARD DE CLASSIFICATION . . . . .	80
A.5	COMPARATIF DE CLASSIFIEURS ( <a href="#">PEDREGOSA ET AL., 2011</a> ) . . . . .	82
A.6	EXEMPLE DE SCHÉMA D'IMPLANTATION . . . . .	84
A.7	DOCUMENTATION DE BRAINPIPE . . . . .	85

## A.1 CARTES DES INTERFACES CERVEAU-MACHINE (GRAIMANN ET AL., 2009)

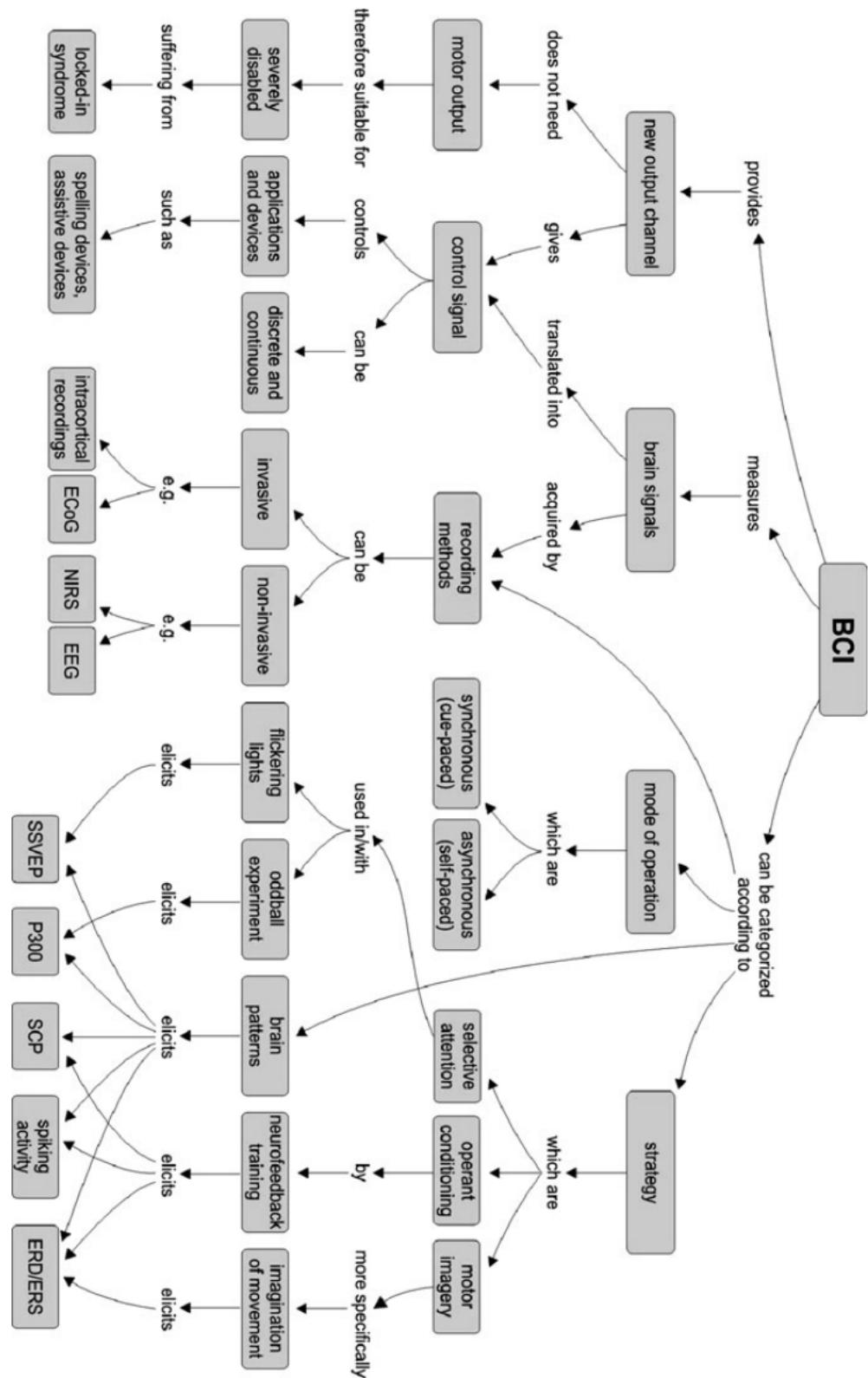


FIGURE A.1 – Cartes des Interfaces Cerveau-Machine (Graimann et al., 2009)

## A.2 JEUX DE DONNÉES EN LIBRE ACCÈS (*BCI competition*)

BCI Competition II			
Set	N-Classes	Channel	Challenge
Ia/Ib	2	6-EEG	Decide whether the subject tried to produce cortical negativity or cortical positivity
IIa	4	64-EEG	Provide the intended target of the feedback test trials
IIb	36	64-EEG	Estimate to which letter of a 6-by-6 matrix with successively intensified rows resp. columns the subject was paying attention to
III	2	3-EEG	Provide a continuous output that could be used for a BCI- feedback
IV	2	28-EEG	Predict the laterality of upcoming finger movements (left vs. right hand) 130 ms before key-press
BCI Competition III			
I	2	64-ECoG	Cued motor imagery (left pinky, tongue) from one subject
II	36	64-EEG	Estimate to which letter of a 6-by-6 matrix with successively intensified rows resp. columns the subject was paying attention to
IIIa	4	60-EEG	Cued motor imagery with 4 classes (left hand, right hand, foot, tongue) from 3 subjects. Measure : kappa-coefficient
IIIb	2	2-EEG	Cued motor imagery with online feedback with 2 classes (left hand, right hand). Measure : mutual information
IVa/IVb/IVc	2	118-EEG	Cued motor imagery with 2 classes (right hand, foot) from 5 subjects
V	3	32-EEG	Cued mental imagery with 3 classes (left hand, right hand, word association) from 3 subjects
BCI Competition IV			
I	2	64-EEG	Motor imagery (2 classes of left hand, right hand, foot)
IIa	4	22-EEG	Cued motor imagery (left hand, right hand, feet, tongue)
IIb	2	3-EEG	Cued motor imagery (left hand, right hand)
III	4	10-MEG	Decoding directions of finger/hand/wrist movements
IV	5	64-ECoG	Discrimination of movements of individual fingers

FIGURE A.2 – Jeux de données en libre accès (*BCI competition*)

### A.3 COMPARATIF DE MÉTHODES PAC ([TORT ET AL., 2010](#))

TABLE 1. *Summary of characteristics of the phase-amplitude coupling measures studied*

Phase-Amplitude Coupling Measure	Tolerance to Noise	Amplitude Independent	Sensitivity to Multimodality	Sensitivity to Modulation Width
Modulation index	Good	Yes	Good	Good
Heights ratio	Good	Yes	No discrimination	No
Mean vector length	Good	No	Restricted	Reasonable
Amplitude PSD	Low	No	Restricted	Good
Phase-locking value	Low	No*	Restricted	Low
Correlation measure	Low	No*	Restricted	Low
GLM measure	Low	No*	Restricted	Low
Coherence value	Low	No*	Restricted	Low

\* Under the presence of noise.

FIGURE A.3 – Comparatif de méthodes PAC ([Tort et al., 2010](#))

## A.4 PIPELINE STANDARD DE CLASSIFICATION

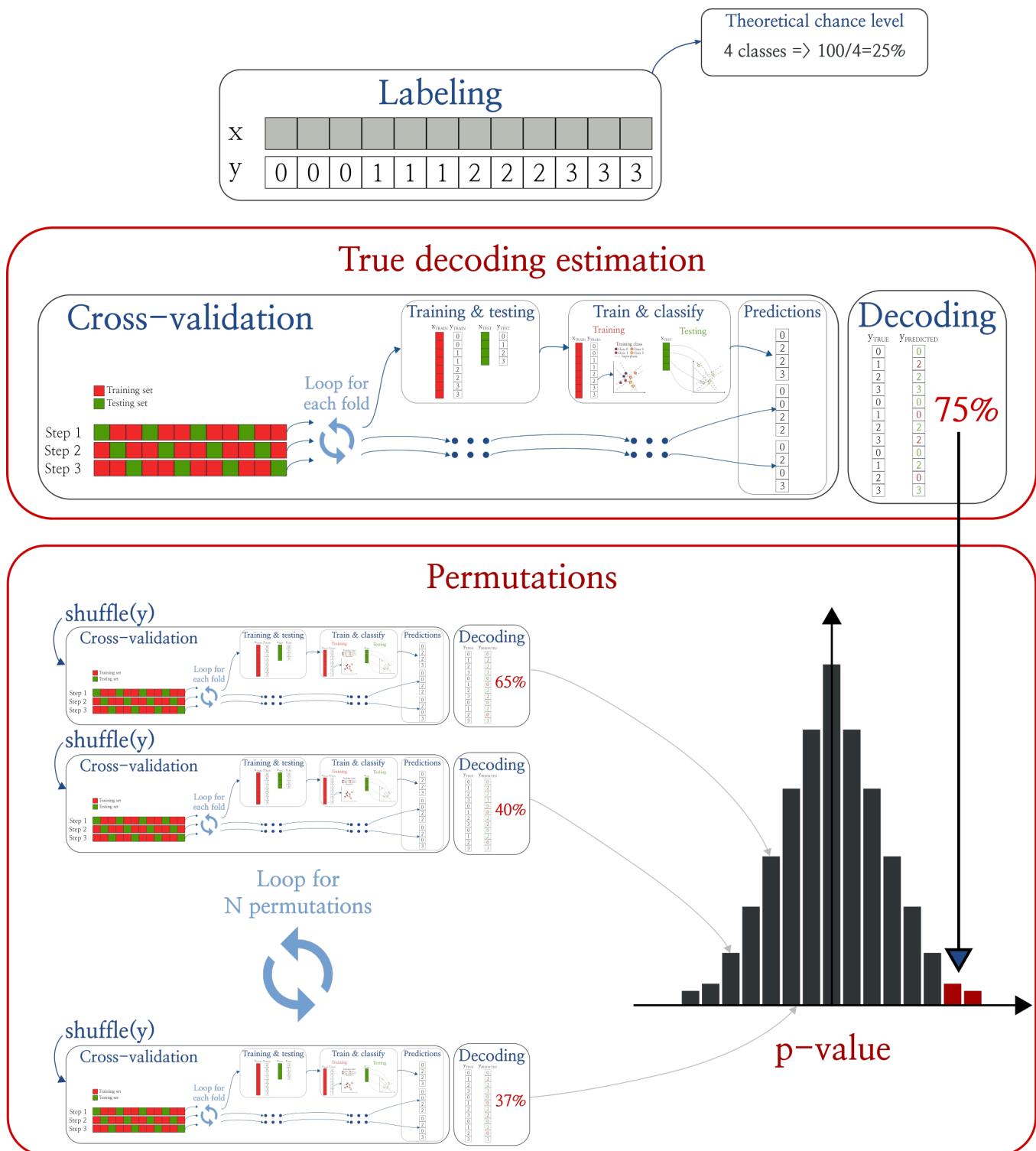


FIGURE A.4 – Pipeline standard de classification



## A.5 COMPARATIF DE CLASSIFIERS (PEDREGOSA ET AL., 2011)

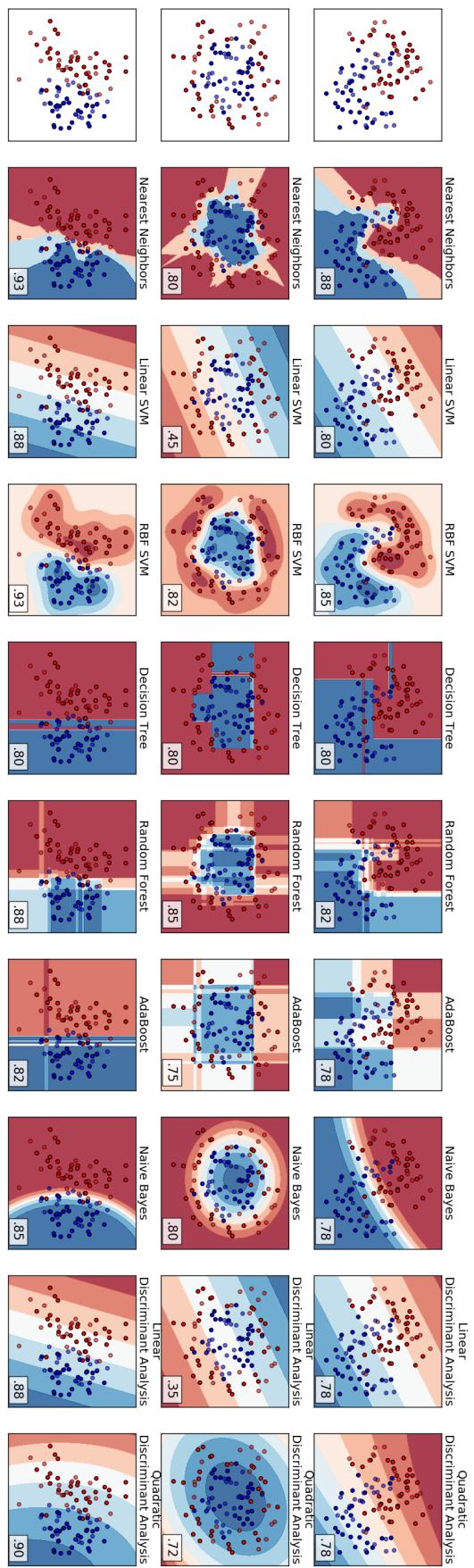
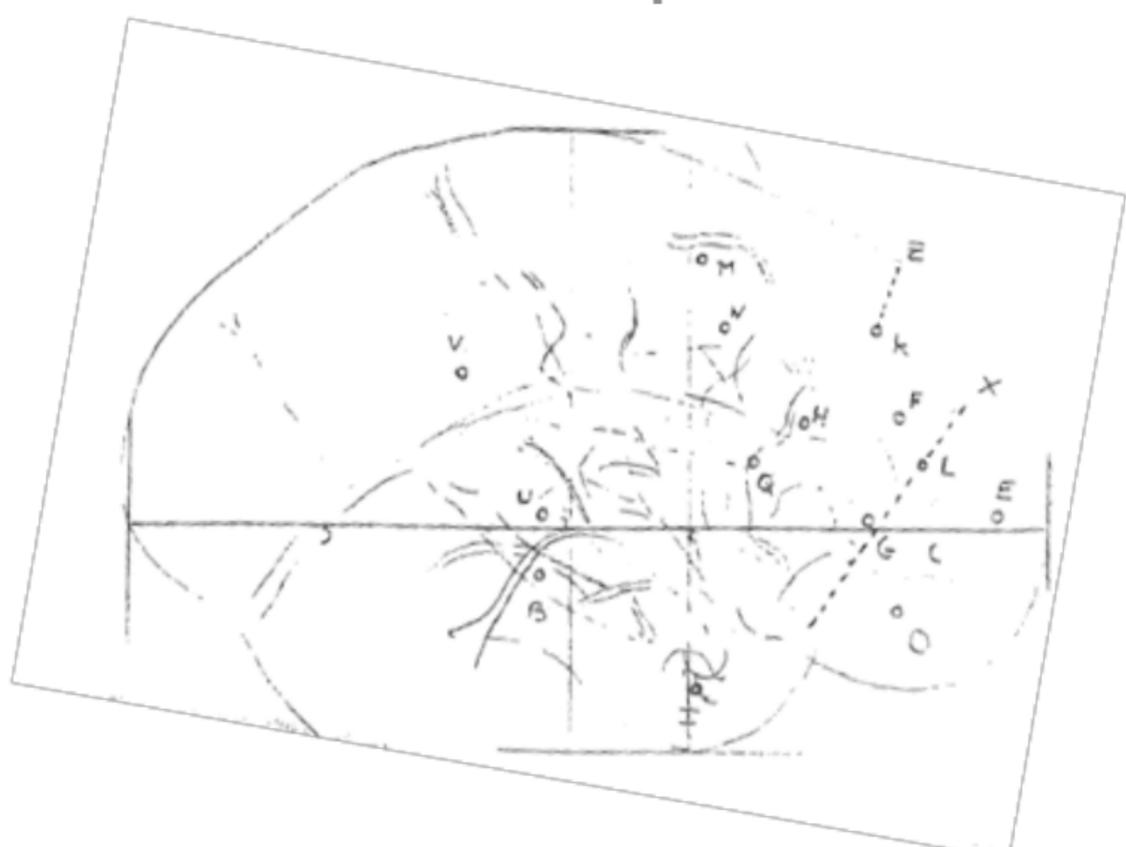
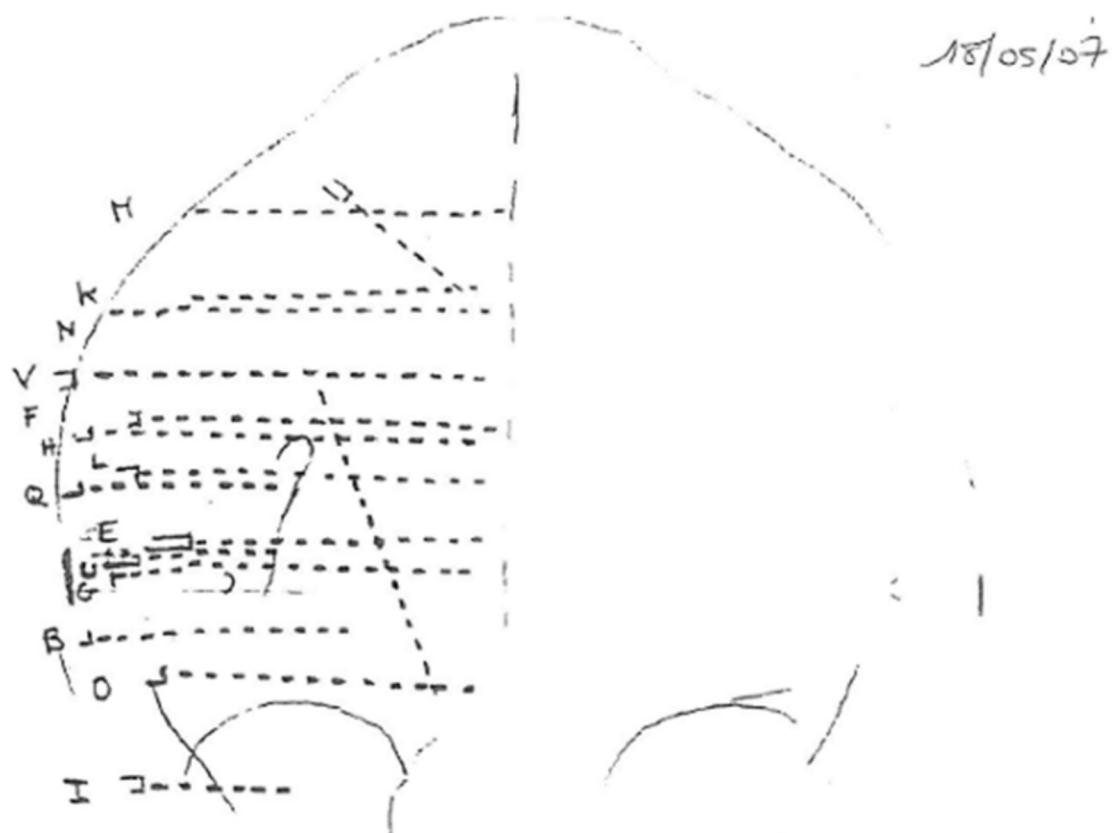


FIGURE A.5 – Comparatif de classifieurs (Pedregosa et al., 2011)



## A.6 EXEMPLE DE SCHÉMA D'IMPLANTATION



## A.7 DOCUMENTATION DE BRAINPIPE

---

# **brainpipe Documentation**

***Release 0.1.6***

**Etienne Combrisson**

Oct 09, 2016



## PRE PROCESSING

<b>1 Requirement</b>	<b>3</b>
<b>2 Installation</b>	<b>5</b>
<b>3 What's new</b>	<b>7</b>
3.1 v0.1.6 . . . . .	7
3.2 v0.1.5 . . . . .	7
3.3 v0.1.4 . . . . .	7
3.4 v0.1.0 . . . . .	8
<b>4 Organization</b>	<b>9</b>
4.1 Bipolarization . . . . .	9
4.2 Physiology . . . . .	10
4.3 Presentation . . . . .	11
4.4 Filtering based . . . . .	11
4.4.1 Filtered signal . . . . .	12
4.4.2 Amplitude . . . . .	13
4.4.3 Power . . . . .	15
4.4.4 Time-Frequency . . . . .	17
4.4.5 Phase . . . . .	20
4.4.6 Phase-Locking Factor . . . . .	21
4.5 Coupling features . . . . .	23
4.5.1 Phase-Amplitude Coupling . . . . .	23
4.5.2 Prefered-phase . . . . .	25
4.5.3 Phase-locked power . . . . .	27
4.5.4 Event Related Phase-Amplitude Coupling . . . . .	28
4.5.5 Phase-Locking Value . . . . .	29
4.6 PSD based features . . . . .	30
4.6.1 Power Spectrum Density . . . . .	30
4.6.2 Power PSD . . . . .	31
4.6.3 Spectral entropy . . . . .	31
4.7 Tools . . . . .	32
4.7.1 Physiological bands . . . . .	32
4.7.2 Cross-frequency vector . . . . .	32
4.7.3 Simulation . . . . .	33
4.8 Presentation . . . . .	33
4.9 Define a classifier . . . . .	34
4.10 Define a cross-validation . . . . .	35
4.11 Classify . . . . .	35
4.12 Leave p-subjects out . . . . .	37
4.13 Generalization . . . . .	38

4.14	Multi-features . . . . .	39
4.15	Binomial . . . . .	43
4.16	Permutations . . . . .	43
4.16.1	Evaluation . . . . .	43
4.16.2	Generate . . . . .	44
4.17	Multiple-comparisons . . . . .	45
4.17.1	Bonferroni . . . . .	45
4.17.2	False Discovery Rate (FDR) . . . . .	45
4.17.3	Maximum statistic . . . . .	46
4.18	Circular statistics toolbox . . . . .	46
4.19	1-D graphics . . . . .	47
4.19.1	Border plot . . . . .	47
4.19.2	p-value plot . . . . .	48
4.19.3	Continuous color . . . . .	48
4.20	1-D or 2-D graphics . . . . .	49
4.20.1	Add lines . . . . .	49
4.20.2	tilerplot . . . . .	50
4.21	Tools . . . . .	52
4.22	Tools . . . . .	52
4.22.1	Bpstudy . . . . .	52
4.22.2	Pandas complements . . . . .	54
4.22.3	Arrays . . . . .	55
4.22.4	File management . . . . .	55
Save file . . . . .	55	
Load file . . . . .	55	
4.22.5	Others . . . . .	55
4.23	References . . . . .	56
4.23.1	Pre-processing . . . . .	56
4.23.2	Features . . . . .	56
4.23.3	Classification . . . . .	56
4.23.4	Statistics . . . . .	57
4.23.5	Visualization . . . . .	57
4.23.6	Tools . . . . .	57

Brainpipe is a python toolbox dedicated for neuronal signals analysis and machine-learning. The aim is to provide a variety of tools to extract informations from neural activities (features) and use machine-learning to validate hypothesis. The machine-learning used the excellent [scikit-learn](#) library. Brainpipe can also perform parallel computing and try to optimize RAM usage for our ‘large’ datasets. If you want to have a quick overview of what you can actually do, checkout the [References](#).

It’s evolving every day! So if you have problems, bugs or if you want to collaborate and add your own tools, contact me at [e.combrisson@gmail.com](mailto:e.combrisson@gmail.com)



---

**CHAPTER  
ONE**

---

**REQUIREMENT**

brainpipe is developed on Python 3, so the compatibility with python 2 is not guaranteed! (not tested yet)

Please, check if you have this toolbox installed and already up-to-date:

- matplotlib (visualization)
- scikit-learn (machine learning)
- joblib (parallel computing)
- scipy
- numpy



---

**CHAPTER  
TWO**

---

## **INSTALLATION**

For instance, the easiest way of installing brainpipe is to use github ([brainpipe](#) ).

Go to your python site-package folder (ex: anaconda3/lib/python3.5/site-packages) and in a terminal run

```
git clone git@github.com:EtienneCmb/brainpipe.git
```



---

**CHAPTER  
THREE**

---

**WHAT'S NEW**

### **3.1 v0.1.6**

- Fix scikit-learn v0.18 compatibility
- Add multi-features pipelines

### **3.2 v0.1.5**

- classify: fit\_stat() has been removed, there is only fit() now. confusion\_matrix() has been renamed to cm()
- LeavePSubjectsOut: new leave p-subjects out cross validation
- classification: embedded visualization (daplot, cmplot), statistics (classify.stat.), dataframe for quick settings summarize (classify.info.) with excel exportation
- New folder: ipython notebook examples
- tools: unique ordered function

### **3.3 v0.1.4**

- PAC: new surrogates method (shuffle amplitude time-series) + phase synchro
- New features (phase-locked power (coupling), ERPAC (coupling), pfdphase (coupling), Phase-Locking Value (coupling), Phase-locking Factor (PLF in spectral) and PSD based features)
- New tools for physiological bands definition
- New plotting function (addPval, continuouscol)
- Start to make the python adaptation of circstat Matlab toolbox
- Add contour to plot2D() and some other parameters
- New doc ! Checkout the *References*
- Bug fix: pac phase shuffling method, coupling time vector, statistical evaluation of permutations

## 3.4 v0.1.0

- **Statistics:**

- Permutations: array optimized permutation module
- p-values on permutations can be compute on 1 tail (upper and lower part) or two tails
- metric:
- Multiple comparison: maximum statistique, Bonferroni, FDR

- **Features:**

- sigfilt//amplitude//power//TF: wilcoxon/Kruskal-Wallis/permuations stat test (comparison with baseline)
- PAC: new Normalized Direct PAC method (Ozkurt, 2012)

- **Visualization:**

- tilerplot() with plot1D() and plot2D() with automatic control of subplot

- **Tools:**

- Array: ndsplit and ndjoin method which doesn't depend on odd/even size (but return list)
- squarefreq() generate a square frequency vector

---

## CHAPTER FOUR

---

## ORGANIZATION

---

### Todo

Add trials rejection (MATLAB code adaptation)

---

```
from brainpipe.preprocessing import *
```

## 4.1 Bipolarization

```
preprocessing.bipolarization(data, channel, dim=0, xyz=None, sep='.', unbip=None, rmchan=None, keepchan='all', rmspace=True, rmalone=True)
```

Bipolarize data

Args:

**data: array** Data to bipolarize

**channel: list** List of channels name

Kwargs:

**dim: integer, optional, [def: 0]** Specify where is the channel dimension of data

**xyz: array, optional, [def: None]** Electrode coordinates. Must be a n\_channel x 3

**sep: string, optional, [def: '.']** Separator to simplify electrode names by removing undesired name after the sep. For example, if channel = ['h1.025', 'h2.578'] and sep='.', the final name will be 'h2-h1'.

**unbip: list, optional, [def: None]** Channel that don't need a bipolarization but to keep. This list can either be the index or the name of the channel.

**rmchan: list, optional, [def: None]** Channel to remove. This list can either be the index or the name of the channel.

**keepchan: list, optional, [def: 'all']** Channel to keep. This list can either be the index or the name of the channel.

**rmspace: bool, optional, [def: True]** Remove undesired space in channel names.

**rmalone: bool, optional, [def: True]** Remove electrodes that cannot be bipolarized.

Returns:

**data\_b: array** Bipolarized data.

**channel\_b: list** List of the bipolarized channels name.

**xyz\_b:** array Array of the new xyz coordinates.

**Example :**

```
>>> x = 47
>>> f = np.array(47, 54, 85)
```

## 4.2 Physiology

**class preprocessing .xyz2phy (nearest=True, r=5, rm\_unfound=False)**

Transform coordinates to physiological informations.

**Args:**

**nearest: bool, optional, [def: True]** If no physiological is found, use this parameter to force to search in sphere of interest.

**r: integer, optional, [def: 5]** Find physiological informations inside a sphere of interest with a radius of r.

**rm\_unfound: bool, optional, [def: False]** Remove un-found structures

**search (df, \*keep)**

Search in a pandas dataframe.

**Args:**

**df: pandas dataframe** The dataframe to filter

**keep: tuple/list** Control the informations to search. See the syntax definition

**Return:** List of row index for informations found.

**keep (df, \*keep, \*, keep\_idx=True)**

Filter a pandas dataframe and keep only interresting rows.

**Args:**

**df: pandas dataframe** The dataframe to filter

**keep: tuple/list** Control the informations to keep. See the syntax definition

**keep\_idx: bool, optional [def [True]]** Add a column to df to check what are the rows that has been kept.

**Return:** A pandas Dataframe with only the informations to keep.

**remove (df, \*rm, \*, rm\_idx=True)**

Filter a pandas dataframe and remove only interresting rows.

**Args:**

**df: pandas dataframe** The dataframe to be filter

**rm: tuple/list** Control the informations to remove. See the syntax definition

**rm\_idx: bool, optional [def [True]]** Add a column to df to check what are the rows that has been removed.

**Return:** A pandas Dataframe without the removed informations.

**get (xyz, channel=[])**

Get physiological informations from (x,y,z) coordinates.

**Args:**

**xyz: array** Array of coordinates. The shape of xyz must be (n\_electrodes x 3).

**channel: list, optional, [def: []]** List of channels name.

**Return:** A pandas DataFrame with physiological informations;

Import features:

```
from brainpipe.features import *
```

---

**Todo**

Coherence // permutation entropy // wavelet filtering (numpy.wlt)

---

## 4.3 Presentation

In order to classify different conditions, you can extract from your neural signals, a large variety of features. The aim of a feature is to verify if it contains the main information you want to classify. For example, let's say you search if an electrode is accurate to differentiate resting state from motor behavior. You can for example extract beta or gamma power from this electrode and classify your resting state versus motor using power features. Here's the list of all the implemented features:

- *Filtered signal*
- *Amplitude*
- *Power*
- *Time-Frequency*
- *Phase*
- *Phase-Locking Factor*
- *Phase-Amplitude Coupling*
- *Preferred-phase*
- *Event Related Phase-Amplitude Coupling*
- *Phase-locked power*
- *Phase-Locking Value*
- *Power Spectrum Density*
- *Power PSD*
- *Spectral entropy*

## 4.4 Filtering based

Those following features use filtering method to extract informations in specific frequency bands

#### 4.4.1 Filtered signal

```
class feature.sigfilt(sf, npts, f=[60, 200], baseline=None, norm=None, window=None, width=None,
                      step=None, split=None, time=None, **kwargs)
```

Extract the filtered signal. This class is optimized for 3D arrays.

Args:

**sf: int** Sampling frequency

**npts: int** Number of points of the time serie

**f: tuple/list** List containing the couple of frequency bands to extract spectral informations. Alternatively, f can be define with the form f=(fstart, fend, fwidth, fstep) where fstart and fend are the starting and endind frequencies, fwidth and fstep are the width and step of each band.

```
>>> # Specify each band:  
>>> f = [ [15, 35], [25, 45], [35, 55], [45, 60], [55, 75] ]  
>>> # Second option:  
>>> f = (15, 75, 20, 10)
```

**window: tuple/list/None, optional [def: None]** List/tuple: [100,1500] List of list/tuple: [(100,500),(200,4000)] None and the width and step parameters will be considered

**width: int, optional [def: None]** width of a single window.

**step: int, optional [def: None]** Each window will be spaced by the “step” value.

**time: list/array, optional [def: None]** Define a specific time vector

**norm: int, optional [def: None]**

**Number to choose the normalization method**

- 0: No normalisation
- 1: Substraction
- 2: Division
- 3: Subtract then divide
- 4: Z-score

**baseline: tuple/list of int [def: None]** Define a window to normalize the power

**split: int or list of int, optional [def: None]** Split the frequency band f in “split” band width.  
If f is a list which contain couple of frequencies, split is a list too.

**kwargs:** supplementar arguments for filtering

**filtname: string, optional [def: ‘fir1’]**

**Name of the filter. Possible values are:**

- ‘fir1’: Window-based FIR filter design
- ‘butter’: butterworth filter
- ‘bessel’: bessel filter

**cycle: int, optional [def: 3]** Number of cycle to use for the filter. This parameter is only available for the ‘fir1’ method

**order: int, optional [def: 3]** Order of the ‘butter’ or ‘bessel’ filter

**axis: int, optional [def: 0]** Filter accross the dimension ‘axis’

**get** (*x, statmeth=None, tail=2, n\_perm=200, metric='m\_center', maxstat=False, n\_jobs=-1*)  
Get the spectral feature of the signal *x*.

**Args:**

**x: array** Data with a shape of (n\_electrodes x n\_pts x n\_trials)

**Kargs:**

**statmeth: string, optional, [def: None]** Method to evaluate the statistical signifiancy. To get p-values, the program will compare real values with a defined baseline. As a consequence, the ‘norm’ and ‘baseline’ parameter should not be None.

- ‘permutation’: randomly shuffle real data with baseline. Control the number of permutations with the *n\_perm* parameter. For example, if *n\_perm* = 1000, this mean that minimum p-valueswill be 0.001.
- ‘wilcoxon’: Wilcoxon signed-rank test
- ‘kruskal’: Kruskal-Wallis H-test

**n\_perm: integer, optional, [def: 200]** Number of permutations for assessing statistical signifiancy.

**tail: int, optional, [def: 2]** For the permutation method, get p-values from one or two tails of the distribution. Use -1 for testing A<B, 1 for A>B and 2 for A~B.

**metric: string/function type, optional, [def: ‘m\_center’]** Use diffrent metrics to normalize data and permutations by the defined baseline. Use:

- None: compare directly values without transformation
- ‘m\_center’: (A-B)/mean(B) transformation
- ‘m\_zscore’: (A-B)/std(B) transformation
- ‘m\_minus’: (A-B) transformation
- function: user defined function [def myfcn(A, B): return array\_like]

**maxstat: bool, optional, [def: False]** Correct p-values with maximum statistique. If maxstat is True, the correction will be applied only trhough frequencies.

**n\_jobs: integer, optional, [def: -1]** Control the number of jobs to extract features. If *n\_jobs* = -1, all the jobs are used.

**Return:**

**xF: array** The un/normalized feature of *x*, with a shape of (n\_frequency x n\_electrodes x n\_window x n\_trials)

**pvalues: array** p-values with a shape of (n\_frequency x n\_electrodes x n\_window)

## 4.4.2 Amplitude

```
class feature.amplitude(sf, npts, f=[60, 200], baseline=None, norm=None, method='hilbert1', window=None, width=None, step=None, split=None, time=None, **kwargs)
```

Extract the amplitude of the signal. This class is optimized for 3D arrays.

**Args:**

**sf: int** Sampling frequency

**npts: int** Number of points of the time serie

**f: tuple/list** List containing the couple of frequency bands to extract spectral informations. Alternatively, f can be define with the form f=(fstart, fend, fwidth, fstep) where fstart and fend are the starting and endind frequencies, fwidth and fstep are the width and step of each band.

```
>>> # Specify each band:  
>>> f = [ [15, 35], [25, 45], [35, 55], [45, 60], [55, 75] ]  
>>> # Second option:  
>>> f = (15, 75, 20, 10)
```

**window: tuple/list/None, optional [def: None]** List/tuple: [100,1500] List of list/tuple: [(100,500),(200,4000)] None and the width and step parameters will be considered

**width: int, optional [def: None]** width of a single window.

**step: int, optional [def: None]** Each window will be spaced by the “step” value.

**time: list/array, optional [def: None]** Define a specific time vector

**norm: int, optional [def: None]**

#### Number to choose the normalization method

- 0: No normalisation
- 1: Subtraction
- 2: Division
- 3: Subtract then divide
- 4: Z-score

**baseline: tuple/list of int [def: None]** Define a window to normalize the power

**split: int or list of int, optional [def: None]** Split the frequency band f in “split” band width.  
If f is a list which contain couple of frequencies, split is a list too.

**method: string**

#### Method to transform the signal. Possible values are:

- ‘hilbert’: apply a hilbert transform to each column
- ‘hilbert1’: hilbert transform to a whole matrix
- ‘hilbert2’: 2D hilbert transform
- ‘wavelet’: wavelet transform

**kwargs: supplementar arguments for filtering**

**filname: string, optional [def: ‘fir1’]**

#### Name of the filter. Possible values are:

- ‘fir1’: Window-based FIR filter design
- ‘butter’: butterworth filter
- ‘bessel’: bessel filter

**cycle: int, optional [def: 3]** Number of cycle to use for the filter. This parameter is only available for the ‘fir1’ method

**order: int, optional [def: 3]** Order of the ‘butter’ or ‘bessel’ filter

---

**axis: int, optional [def: 0]** Filter accross the dimension ‘axis’

**get** (*x, statmeth=None, tail=2, n\_perm=200, metric='m\_center', maxstat=False, n\_jobs=-1*)  
Get the spectral feature of the signal *x*.

**Args:**

**x: array** Data with a shape of (n\_electrodes x n\_pts x n\_trials)

**Kargs:**

**statmeth: string, optional, [def: None]** Method to evaluate the statistical signifiancy. To get p-values, the program will compare real values with a defined baseline. As a consequence, the ‘norm’ and ‘baseline’ parameter should not be None.

- ‘permutation’: randomly shuffle real data with baseline. Control the number of permutations with the *n\_perm* parameter. For example, if *n\_perm* = 1000, this mean that minimum p-valueswill be 0.001.
- ‘wilcoxon’: Wilcoxon signed-rank test
- ‘kruskal’: Kruskal-Wallis H-test

**n\_perm: integer, optional, [def: 200]** Number of permutations for assessing statistical signifiancy.

**tail: int, optional, [def: 2]** For the permutation method, get p-values from one or two tails of the distribution. Use -1 for testing A<B, 1 for A>B and 2 for A~B.

**metric: string/function type, optional, [def: ‘m\_center’]** Use diffrent metrics to normalize data and permutations by the defined baseline. Use:

- None: compare directly values without transformation
- ‘m\_center’: (A-B)/mean(B) transformation
- ‘m\_zscore’: (A-B)/std(B) transformation
- ‘m\_minus’: (A-B) transformation
- function: user defined function [def myfcn(A, B): return array\_like]

**maxstat: bool, optional, [def: False]** Correct p-values with maximum statistique. If maxstat is True, the correction will be applied only trhough frequencies.

**n\_jobs: integer, optional, [def: -1]** Control the number of jobs to extract features. If *n\_jobs* = -1, all the jobs are used.

**Return:**

**xF: array** The un/normalized feature of *x*, with a shape of (n\_frequency x n\_electrodes x n\_window x n\_trials)

**pvalues: array** p-values with a shape of (n\_frequency x n\_electrodes x n\_window)

### 4.4.3 Power

```
class feature.power(sf, npts, f=[60, 200], baseline=None, norm=None, method='hilbertI', window=None, width=None, step=None, split=None, time=None, **kwargs)
```

Extract the power of the signal. This class is optimized for 3D arrays.

**Args:**

**sf: int** Sampling frequency

**npts: int** Number of points of the time serie

**f: tuple/list** List containing the couple of frequency bands to extract spectral informations. Alternatively, f can be define with the form f=(fstart, fend, fwidth, fstep) where fstart and fend are the starting and endind frequencies, fwidth and fstep are the width and step of each band.

```
>>> # Specify each band:  
>>> f = [ [15, 35], [25, 45], [35, 55], [45, 60], [55, 75] ]  
>>> # Second option:  
>>> f = (15, 75, 20, 10)
```

**window: tuple/list/None, optional [def: None]** List/tuple: [100,1500] List of list/tuple: [(100,500),(200,4000)] None and the width and step parameters will be considered

**width: int, optional [def: None]** width of a single window.

**step: int, optional [def: None]** Each window will be spaced by the “step” value.

**time: list/array, optional [def: None]** Define a specific time vector

**norm: int, optional [def: None]**

#### Number to choose the normalization method

- 0: No normalisation
- 1: Subtraction
- 2: Division
- 3: Subtract then divide
- 4: Z-score

**baseline: tuple/list of int [def: None]** Define a window to normalize the power

**split: int or list of int, optional [def: None]** Split the frequency band f in “split” band width.  
If f is a list which contain couple of frequencies, split is a list too.

**method: string**

#### Method to transform the signal. Possible values are:

- ‘hilbert’: apply a hilbert transform to each column
- ‘hilbert1’: hilbert transform to a whole matrix
- ‘hilbert2’: 2D hilbert transform
- ‘wavelet’: wavelet transform

**kwargs: supplementar arguments for filtering**

**filname: string, optional [def: ‘fir1’]**

#### Name of the filter. Possible values are:

- ‘fir1’: Window-based FIR filter design
- ‘butter’: butterworth filter
- ‘bessel’: bessel filter

**cycle: int, optional [def: 3]** Number of cycle to use for the filter. This parameter is only available for the ‘fir1’ method

**order: int, optional [def: 3]** Order of the ‘butter’ or ‘bessel’ filter

**axis: int, optional [def: 0]** Filter accross the dimension ‘axis’

**get** (*x, statmeth=None, tail=2, n\_perm=200, metric='m\_center', maxstat=False, n\_jobs=-1*)  
Get the spectral feature of the signal *x*.

**Args:**

**x: array** Data with a shape of (n\_electrodes x n\_pts x n\_trials)

**Kargs:**

**statmeth: string, optional, [def: None]** Method to evaluate the statistical signifiancy. To get p-values, the program will compare real values with a defined baseline. As a consequence, the ‘norm’ and ‘baseline’ parameter should not be None.

- ‘permutation’: randomly shuffle real data with baseline. Control the number of permutations with the n\_perm parameter. For example, if n\_perm = 1000, this mean that minimum p-valueswill be 0.001.
- ‘wilcoxon’: Wilcoxon signed-rank test
- ‘kruskal’: Kruskal-Wallis H-test

**n\_perm: integer, optional, [def: 200]** Number of permutations for assessing statistical signifiancy.

**tail: int, optional, [def: 2]** For the permutation method, get p-values from one or two tails of the distribution. Use -1 for testing A<B, 1 for A>B and 2 for A~B.

**metric: string/function type, optional, [def: ‘m\_center’]** Use diffrent metrics to normalize data and permutations by the defined baseline. Use:

- None: compare directly values without transformation
- ‘m\_center’: (A-B)/mean(B) transformation
- ‘m\_zscore’: (A-B)/std(B) transformation
- ‘m\_minus’: (A-B) transformation
- function: user defined function [def myfcn(A, B): return array\_like]

**maxstat: bool, optional, [def: False]** Correct p-values with maximum statistique. If maxstat is True, the correction will be applied only trhough frequencies.

**n\_jobs: integer, optional, [def: -1]** Control the number of jobs to extract features. If n\_jobs = -1, all the jobs are used.

**Return:**

**xF: array** The un/normalized feature of *x*, with a shape of (n\_frequency x n\_electrodes x n\_window x n\_trials)

**pvalues: array** p-values with a shape of (n\_frequency x n\_electrodes x n\_window)

#### 4.4.4 Time-Frequency

```
class feature.TF(sf, npts, f=(2, 200, 10, 5), baseline=None, norm=None, method='hilbert1', window=None, width=None, step=None, time=None, **kwargs)
```

Extract the time-frequency map of the signal. This class is optimized for 3D arrays.

**Args:**

**sf: int** Sampling frequency

**npts: int** Number of points of the time serie

**f: tuple/list** List containing the couple of frequency bands to extract spectral informations. Alternatively, f can be define with the form f=(fstart, fend, fwidth, fstep) where fstart and fend are the starting and endind frequencies, fwidth and fstep are the width and step of each band.

```
>>> # Specify each band:  
>>> f = [ [15, 35], [25, 45], [35, 55], [45, 60], [55, 75] ]  
>>> # Second option:  
>>> f = (15, 75, 20, 10)
```

**window: tuple/list/None, optional [def: None]** List/tuple: [100,1500] List of list/tuple: [(100,500),(200,4000)] None and the width and step parameters will be considered

**width: int, optional [def: None]** width of a single window.

**step: int, optional [def: None]** Each window will be spaced by the “step” value.

**time: list/array, optional [def: None]** Define a specific time vector

**norm: int, optional [def: None]**

#### Number to choose the normalization method

- 0: No normalisation
- 1: Subtraction
- 2: Division
- 3: Subtract then divide
- 4: Z-score

**baseline: tuple/list of int [def: None]** Define a window to normalize the power

**split: int or list of int, optional [def: None]** Split the frequency band f in “split” band width.  
If f is a list which contain couple of frequencies, split is a list too.

**method: string**

#### Method to transform the signal. Possible values are:

- ‘hilbert’: apply a hilbert transform to each column
- ‘hilbert1’: hilbert transform to a whole matrix
- ‘hilbert2’: 2D hilbert transform
- ‘wavelet’: wavelet transform

**kwargs: supplementar arguments for filtering**

**filname: string, optional [def: ‘fir1’]**

#### Name of the filter. Possible values are:

- ‘fir1’: Window-based FIR filter design
- ‘butter’: butterworth filter
- ‘bessel’: bessel filter

**cycle: int, optional [def: 3]** Number of cycle to use for the filter. This parameter is only available for the ‘fir1’ method

**order: int, optional [def: 3]** Order of the ‘butter’ or ‘bessel’ filter

**axis: int, optional [def: 0]** Filter accross the dimension ‘axis’

**get** ( $x$ ,  $statmeth=None$ ,  $tail=2$ ,  $n\_perm=200$ ,  $metric='m\_center'$ ,  $maxstat=False$ ,  $n\_jobs=-1$ )  
Get the spectral feature of the signal  $x$ .

**Args:**

**x: array** Data with a shape of (n\_electrodes x n\_pts x n\_trials)

**Kargs:**

**statmeth: string, optional, [def: None]** Method to evaluate the statistical signifiancy. To get p-values, the program will compare real values with a defined baseline. As a consequence, the ‘norm’ and ‘baseline’ parameter should not be None.

- ‘permutation’: randomly shuffle real data with baseline. Control the number of permutations with the  $n\_perm$  parameter. For example, if  $n\_perm = 1000$ , this mean that minimum p-valueswill be 0.001.
- ‘wilcoxon’: Wilcoxon signed-rank test
- ‘kruskal’: Kruskal-Wallis H-test

**n\_perm: integer, optional, [def: 200]** Number of permutations for assessing statistical signifiancy.

**tail: int, optional, [def: 2]** For the permutation method, get p-values from one or two tails of the distribution. Use -1 for testing  $A < B$ , 1 for  $A > B$  and 2 for  $A \sim B$ .

**metric: string/function type, optional, [def: ‘m\_center’]** Use diffrent metrics to normalize data and permutations by the defined baseline. Use:

- None: compare directly values without transformation
- ‘m\_center’:  $(A-B)/\text{mean}(B)$  transformation
- ‘m\_zscore’:  $(A-B)/\text{std}(B)$  transformation
- ‘m\_minus’:  $(A-B)$  transformation
- function: user defined function [def myfcn(A, B): return array\_like]

**maxstat: bool, optional, [def: False]** Correct p-values with maximum statistique. If maxstat is True, the correction will be applied only trhough frequencies.

**n\_jobs: integer, optional, [def: -1]** Control the number of jobs to extract features. If  $n\_jobs = -1$ , all the jobs are used.

**Return:**

**xF: array** The un/normalized feature of  $x$ , with a shape of (n\_frequency x n\_electrodes x n\_window x n\_trials)

**pvalues: array** p-values with a shape of (n\_frequency x n\_electrodes x n\_window)

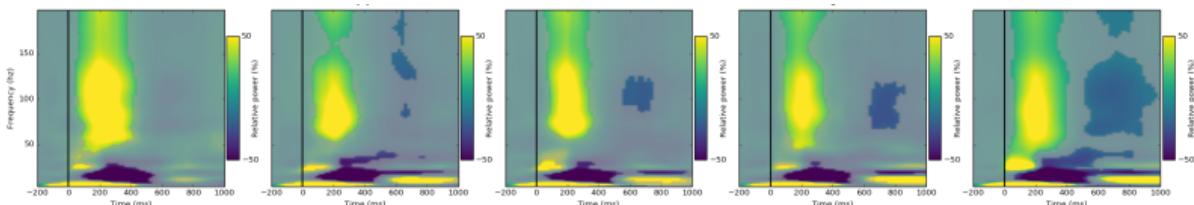


Fig. 4.1: Example of a Time-frequency map

#### 4.4.5 Phase

```
class feature.phase(sf, npts, f=[2, 4], method='hilbert', window=None, width=None, step=None,
                    time=None, **kwargs)
```

Extract the phase of a signal. This class is optimized for 3D arrays.

Args:

**sf: int** Sampling frequency

**npts: int** Number of points of the time serie

**f: tuple/list** List containing the couple of frequency bands to extract spectral informations. Alternatively, f can be define with the form f=(fstart, fend, fwidth, fstep) where fstart and fend are the starting and endind frequencies, fwidth and fstep are the width and step of each band.

```
>>> # Specify each band:  
>>> f = [ [15, 35], [25, 45], [35, 55], [45, 60], [55, 75] ]  
>>> # Second option:  
>>> f = (15, 75, 20, 10)
```

**window: tuple/list/None, optional [def: None]** List/tuple: [100,1500] List of list/tuple: [(100,500),(200,4000)] None and the width and step parameters will be considered

**width: int, optional [def: None]** width of a single window.

**step: int, optional [def: None]** Each window will be spaced by the “step” value.

**time: list/array, optional [def: None]** Define a specific time vector

**method: string**

**Method to transform the signal. Possible values are:**

- ‘hilbert’: apply a hilbert transform to each column
- ‘hilbert1’: hilbert transform to a whole matrix
- ‘hilbert2’: 2D hilbert transform

**kwargs:** supplementar arguments for filtering

**filtname: string, optional [def: ‘fir1’]**

**Name of the filter. Possible values are:**

- ‘fir1’: Window-based FIR filter design
- ‘butter’: butterworth filter
- ‘bessel’: bessel filter

**cycle: int, optional [def: 3]** Number of cycle to use for the filter. This parameter is only available for the ‘fir1’ method

**order: int, optional [def: 3]** Order of the ‘butter’ or ‘bessel’ filter

**axis: int, optional [def: 0]** Filter accross the dimension ‘axis’

**get(x, getstat=True, n\_jobs=-1)**

Get the spectral phase of the signal x.

Args:

**x: array** Data with a shape of (n\_electrodes x n\_pts x n\_trials)

Kargs:

**getstat: bool, optional, [def: True]** Set it to True if p-values should be computed. Statistical p-values are computed using Rayleigh test.

**n\_jobs: integer, optional, [def: -1]** Control the number of jobs to extract features. If n\_jobs = -1, all the jobs are used.

#### Return:

**xF: array** The phase of x, with a shape of (n\_frequency x n\_electrodes x n\_window x n\_trials)

**pvalues: array** p-values with a shape of (n\_frequency x n\_electrodes x n\_window)

**get** (x, getstat=True, n\_jobs=-1)

Get the spectral phase of the signal x.

#### Args:

**x: array** Data with a shape of (n\_electrodes x n\_pts x n\_trials)

#### Kargs:

**getstat: bool, optional, [def: True]** Set it to True if p-values should be computed. Statistical p-values are computed using Rayleigh test.

**n\_jobs: integer, optional, [def: -1]** Control the number of jobs to extract features. If n\_jobs = -1, all the jobs are used.

#### Return:

**xF: array** The phase of x, with a shape of (n\_frequency x n\_electrodes x n\_window x n\_trials)

**pvalues: array** p-values with a shape of (n\_frequency x n\_electrodes x n\_window)

## 4.4.6 Phase-Locking Factor

```
class feature.PLF(sf, npts, f=[2, 4], method='hilbert', window=None, width=None, step=None,
                  time=None, **kwargs)
```

Extract the phase-locking factor of a signal. This class is optimized for 3D arrays.

#### Args:

**sf: int** Sampling frequency

**npts: int** Number of points of the time serie

**f: tuple/list** List containing the couple of frequency bands to extract spectral informations. Alternatively, f can be define with the form f=(fstart, fend, fwidth, fstep) where fstart and fend are the starting and endind frequencies, fwidth and fstep are the width and step of each band.

```
>>> # Specify each band:
>>> f = [ [15, 35], [25, 45], [35, 55], [45, 60], [55, 75] ]
>>> # Second option:
>>> f = (15, 75, 20, 10)
```

**window: tuple/list/None, optional [def: None]** List/tuple: [100,1500] List of list/tuple: [(100,500),(200,4000)] None and the width and step parameters will be considered

**width: int, optional [def: None]** width of a single window.

**step: int, optional [def: None]** Each window will be spaced by the “step” value.

**time: list/array, optional [def: None]** Define a specific time vector

**method: string**

**Method to transform the signal. Possible values are:**

- ‘hilbert’: apply a hilbert transform to each column
- ‘hilbert1’: hilbert transform to a whole matrix
- ‘hilbert2’: 2D hilbert transform

kwargs: supplementar arguments for filtering

**filtnname: string, optional [def: ‘fir1’]**

**Name of the filter. Possible values are:**

- ‘fir1’: Window-based FIR filter design
- ‘butter’: butterworth filter
- ‘bessel’: bessel filter

**cycle: int, optional [def: 3]** Number of cycle to use for the filter. This parameter is only available for the ‘fir1’ method

**order: int, optional [def: 3]** Order of the ‘butter’ or ‘bessel’ filter

**axis: int, optional [def: 0]** Filter accross the dimension ‘axis’

**get** (*x*, *getstat=True*, *n\_jobs=-1*)

Get the phase-locking factor of the signal *x*.

**Args:**

**x: array** Data with a shape of (n\_electrodes x n\_pts x n\_trials)

**Kargs:**

**getstat: bool, optional, [def: True]** Set it to True if p-values should be computed. Statistical p-values are computed using Rayleigh test.

**n\_jobs: integer, optional, [def: -1]** Control the number of jobs to extract features. If n\_jobs = -1, all the jobs are used.

**Return:**

**plf: array** The PLF of *x*, with a shape of (n\_frequency x n\_electrodes x n\_window)

**pvalues: array** p-values with a shape of (n\_frequency x n\_electrodes x n\_window)

**get** (*x*, *getstat=True*, *n\_jobs=-1*)

Get the phase-locking factor of the signal *x*.

**Args:**

**x: array** Data with a shape of (n\_electrodes x n\_pts x n\_trials)

**Kargs:**

**getstat: bool, optional, [def: True]** Set it to True if p-values should be computed. Statistical p-values are computed using Rayleigh test.

**n\_jobs: integer, optional, [def: -1]** Control the number of jobs to extract features. If n\_jobs = -1, all the jobs are used.

**Return:**

**plf: array** The PLF of *x*, with a shape of (n\_frequency x n\_electrodes x n\_window)

**pvalues: array** p-values with a shape of (n\_frequency x n\_electrodes x n\_window)

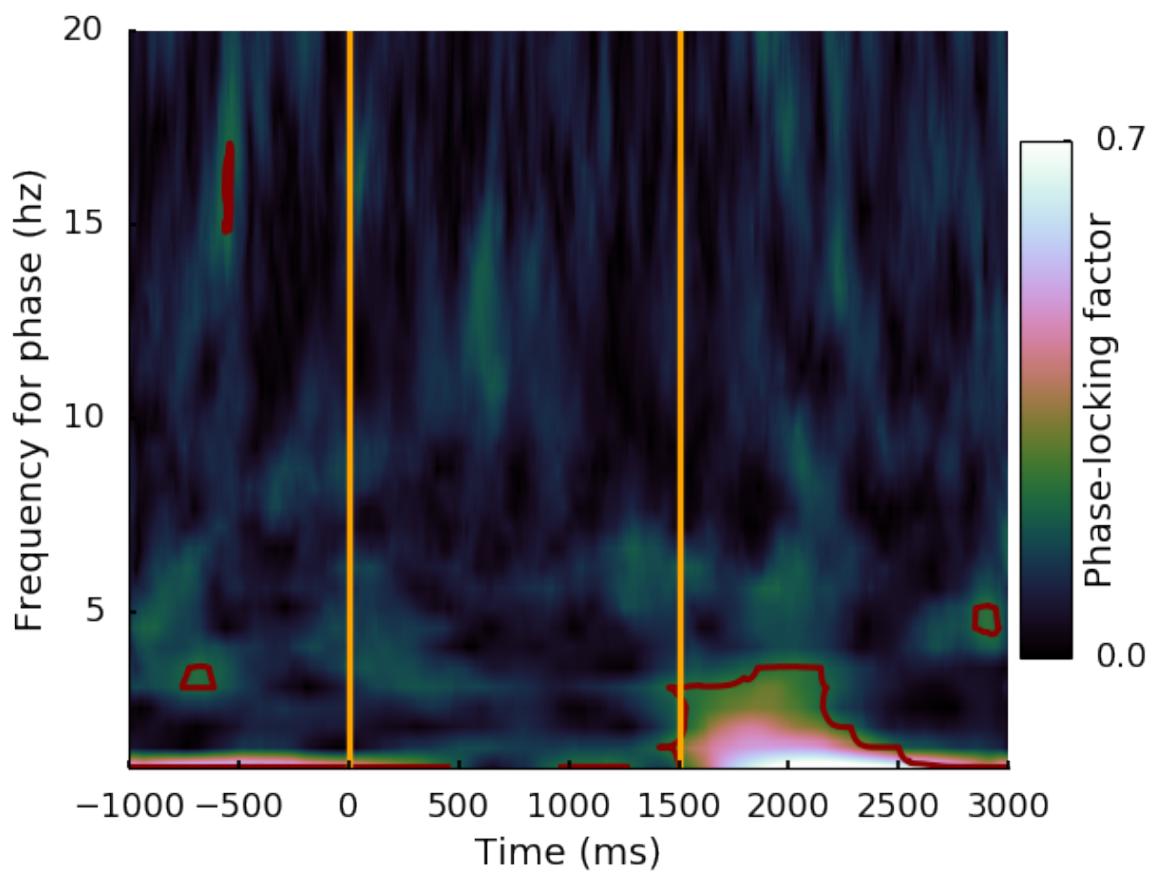


Fig. 4.2: Example of a Phase-Locking Factor map

## 4.5 Coupling features

Those following features use coupling (either distant or locals coupling)

### 4.5.1 Phase-Amplitude Coupling

```
class feature.pac(sf, npts, Id='113', pha_f=[2, 4], pha_meth='hilbert', pha_cycle=3, amp_f=[60,
    200], amp_meth='hilbert', amp_cycle=6, nbins=18, window=None, width=None,
    step=None, time=None, **kwargs)
```

Compute the phase-amplitude coupling (pac) either in local or distant coupling. PAC require three things:

- Main method to compute it
- Surrogates to correct the true pac estimation
- A normalization method to correct pas by surrogates

Contributor: Juan LP Soto.

#### Args:

**sf: int** Sampling frequency  
**npts: int** Number of points of the time serie

#### Kargs:

**Id: string, optional, [def: ‘113’]** The Id correspond to the way of computing pac. Id is composed of three digits [ex : Id=‘210’]

- First digit: refer to the pac method:
  - ‘1’: Mean Vector Length <sup>1</sup>
  - ‘2’: Kullback-Leibler Divergence <sup>2</sup>
  - ‘3’: Heights Ratio
  - ‘4’: Phase synchrony (or adapted PLV) <sup>5</sup>
  - ‘5’: ndPAC <sup>3</sup>
- Second digit: refer to the method for computing surrogates:
  - ‘0’: No surrogates
  - ‘1’: Swap trials phase/amplitude <sup>2</sup>
  - ‘2’: Swap trials amplitude <sup>4</sup>
  - ‘3’: Shuffle phase time-series
  - ‘4’: Shuffle amplitude time-series
  - ‘5’: Time lag <sup>1</sup> [NOT IMPLEMENTED YET]
  - ‘6’: Circular shifting [NOT IMPLEMENTED YET]
- Third digit: refer to the normalization method for correction:

<sup>1</sup> Canolty et al, 2006

<sup>2</sup> Tort et al, 2010

<sup>5</sup> Penny et al, 2008

<sup>3</sup> Ozkurt et al, 2012

<sup>4</sup> Bahramisharif et al, 2013

- ‘0’: No normalization
- ‘1’: Subtract the mean of surrogates
- ‘2’: Divide by the mean of surrogates
- ‘3’: Subtract then divide by the mean of surrogates
- ‘4’: Z-score

So, if Id=‘143’, this mean that pac will be evaluate using the Modulation Index (‘1’), then surrogates are computing by randomly shuffle amplitude values (‘4’) and finally, the true pac value will be normalized by subtracting then dividing by the mean of surrogates.

**pha\_f: tuple/list, optional, [def: [2,4]]** List containing the couple of frequency bands for the phase. Example: f=[ [2,4], [5,7], [60,250] ]

**pha\_meth: string, optional, [def: ‘hilbert’]** Method for the phase extraction.

**pha\_cycle: integer, optional, [def: 3]** Number of cycles for filtering the phase.

**amp\_f: tuple/list, optional, [def: [60,200]]** List containing the couple of frequency bands for the amplitude. Each couple can be either a list or a tuple.

**amp\_meth: string, optional, [def: ‘hilbert’]** Method for the amplitude extraction.

**amp\_cycle: integer, optional, [def: 6]** Number of cycles for filtering the amplitude.

**nbins: integer, optional, [def: 18]** Some pac method (like Kullback-Leibler Distance or Heights Ratio) need a binarization of the phase. nbins control the number of bins.

**window: tuple/list/None, optional [def: None]** List/tuple: [100,1500] List of list/tuple: [(100,500),(200,4000)] Width and step parameters will be ignored.

**width: int, optional [def: None]** width of a single window.

**step: int, optional [def: None]** Each window will be spaced by the “step” value.

**time: list/array, optional [def: None]** Define a specific time vector

**filtname: string, optional [def: ‘fir1’]**

**Name of the filter. Possible values are:**

- ‘fir1’: Window-based FIR filter design
- ‘butter’: butterworth filter
- ‘bessel’: bessel filter

**cycle: int, optional [def: 3]** Number of cycle to use for the filter. This parameter is only available for the ‘fir1’ method

**order: int, optional [def: 3]** Order of the ‘butter’ or ‘bessel’ filter

**axis: int, optional [def: 0]** Filter accross the dimension ‘axis’

**get (xpha, xamp, n\_perm=200, p=0.05, matricial=False, n\_jobs=-1)**

Get the normalized cfc mesure between an xpha and xamp signals.

**Args:**

**xpha: array** Signal for phase. The shape of xpha should be : (n\_electrodes x n\_pts x n\_trials)

**xamp: array** Signal for amplitude. The shape of xamp should be : (n\_electrodes x n\_pts x n\_trials)

**Kargs:**

**n\_perm: integer, optional, [def: 200]** Number of permutations for normalizing the cfc mesure.

**p: float, optional, [def: 0.05]** p-value for the statistical method of Ozkurt 2012.

**matricial: bool, optional, [def: False]** Some methods can work in matricial computation. This can lead to a 10x or 30x time faster. But, please, monitor your RAM usage because this parameter can use a lot of RAM. So, turn this parameter in case of small computation.

**n\_jobs: integer, optional, [def: -1]** Control the number of jobs for parallel computing. Use 1, 2, .. depending of your number of cores. -1 for all the cores.

If the same signal is used (example : xpha=x and xamp=x), this mean the program compute a local cfc.

**Returns:**

**ncfc: array** The cfc mesure of size : (n\_amplitude x n\_phase x n\_electrodes x n\_windows x n\_trials)

**pvalue: array** The associated p-values of size : (n\_amplitude x n\_phase x n\_electrodes x n\_windows)

**Method : Modulation Index (Canolty, 2006)**

**Surrogates : Swap phase/amplitude through trials, (Tort, 2010)**

**Normalization : Subtract then divide by the mean of surrogates**

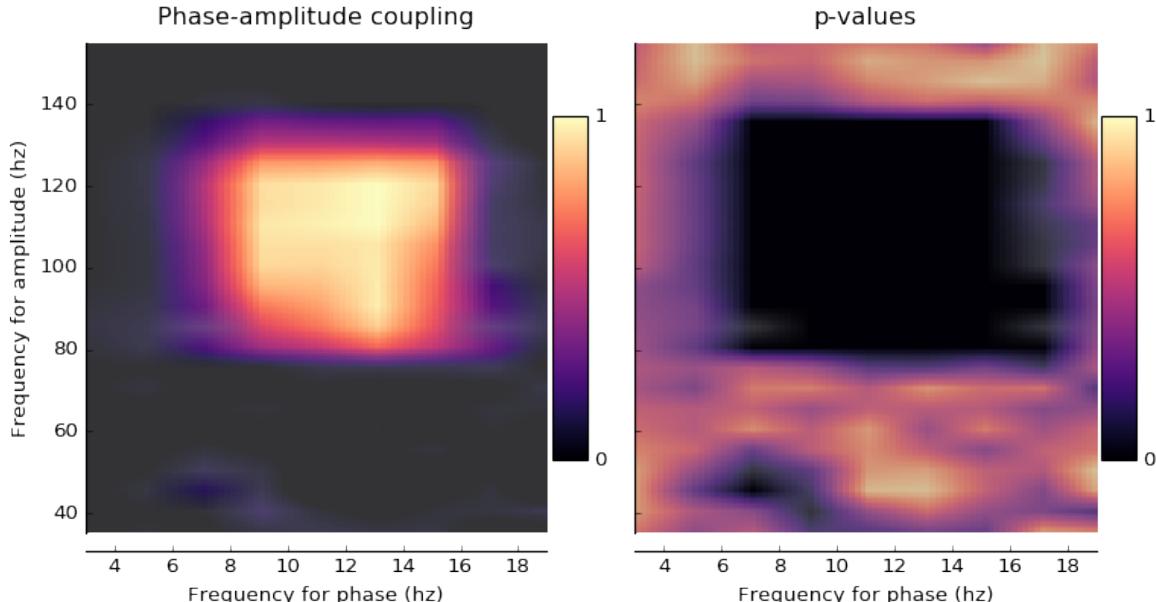


Fig. 4.3: Example of PAC maps for a synthetic coupled signal

## 4.5.2 Preferred-phase

```
class feature.pfdphase(sf, npts, nbins=18, pha_f=[2, 4], pha_meth='hilbert', pha_cycle=3,
                        amp_f=[60, 200], amp_meth='hilbert', amp_cycle=6, window=None,
                        width=None, step=None, time=None, **kwargs)
```

Get the preferred phase of a phase-amplitude coupling

**Args:**

**sf: int** Sampling frequency

**npts: int** Number of points of the time serie

**Kargs:**

**nbins: integer, optional, [def: 18]** Number of bins to binarize the amplitude.

**pha\_f: tuple/list, optional, [def: [2,4]]** List containing the couple of frequency bands for the phase. Example: f=[ [2,4], [5,7], [60,250] ]

**pha\_meth: string, optional, [def: ‘hilbert’]** Method for the phase extraction.

**pha\_cycle: integer, optional, [def: 3]** Number of cycles for filtering the phase.

**amp\_f: tuple/list, optional, [def: [60,200]]** List containing the couple of frequency bands for the amplitude. Each couple can be either a list or a tuple.

**amp\_meth: string, optional, [def: ‘hilbert’]** Method for the amplitude extraction.

**amp\_cycle: integer, optional, [def: 6]** Number of cycles for filtering the amplitude.

**window: tuple/list/None, optional [def: None]** List/tuple: [100,1500] List of list/tuple: [(100,500),(200,4000)] Width and step parameters will be ignored.

**width: int, optional [def: None]** width of a single window.

**step: int, optional [def: None]** Each window will be spaced by the “step” value.

**time: list/array, optional [def: None]** Define a specific time vector

**get** (*xpha*, *xamp*, *n\_jobs=-1*)  
Get the preferred phase

**Args:**

**xpha: array** Signal for phase. The shape of xpha should be : (n\_electrodes x n\_pts x n\_trials)

**xamp: array** Signal for amplitude. The shape of xamp should be : (n\_electrodes x n\_pts x n\_trials)

**Kargs:**

**n\_jobs: integer, optional, [def: -1]** Control the number of jobs for parallel computing. Use 1, 2, .. depending of your number of cores. -1 for all the cores.

If the same signal is used (example : xpha=x and xamp=x), this mean the program compute a local cfc.

**Returns:**

**pfp: array** The preferred phase extracted from the mean of trials of size : (n\_amplitude x n\_phase x n\_electrodes x n\_windows)

**prf: array** The preferred phase extracted from each trial of size : (n\_amplitude x n\_phase x n\_electrodes x n\_windows x n\_trials)

**ambin: array** The binarized amplitude of size : (n\_amplitude x n\_phase x n\_electrodes x n\_windows x n\_bins x n\_trials)

**pvalue: array** The associated p-values of size : (n\_amplitude x n\_phase x n\_electrodes x n\_windows)

### 4.5.3 Phase-locked power

```
class feature.PhaseLockedPower(sf, npts, f=(2, 200, 10, 5), pha=[8, 13], time=None, baseline=None,
                                norm=None, **powArgs)
```

Extract phase-locked power and visualize shifted time-frequency map according to phase peak.

**Args:**

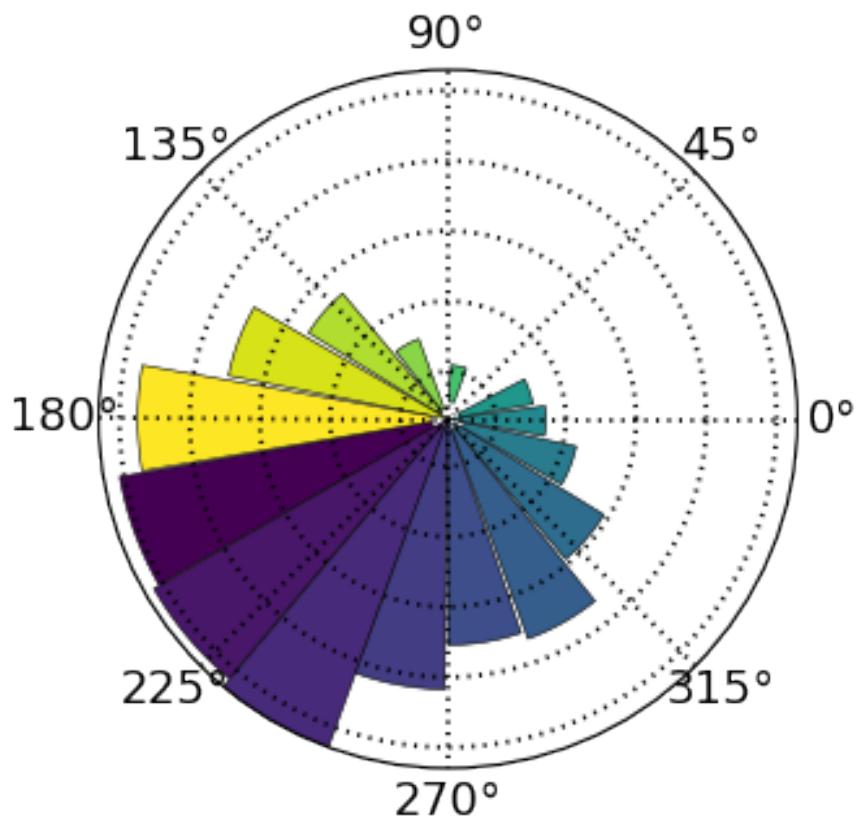


Fig. 4.4: Example of preferred phase for an alpha-gamma coupling in BA6

**sf: int** Sampling frequency

**npts: int** Number of points of the time serie

#### Kargs:

**f: tuple/list, optional, [def: (2, 200, 10, 5)]** The frequency vector (fstart, fend, fwidth, fstep)

**pha: tuple/list, optional, [def: [8, 13]]** Frequency for phase.

**time: array/list, optional, [def: None]** The time vector to use

**baseline: tuple/list, optional, [def: None]** Location of baseline (in sample)

**norm: integer, optional, [def: None]**

#### Normalize method

- 0: No normalisation
- 1: Substraction
- 2: Division
- 3: Subtract then divide
- 4: Z-score

**powArgs:** any supplementar arguments are directly passed to the power function.

#### get (x, cue)

Get power phase locked

#### Args:

**x: array** Data of shape (npt, ntrials)

**cue: integer** Cue to align time-frequency maps.

**Returns:** xpow, xpha, xsig: repectively realigned power, phase and filtered signal

**tflockedplot** (xpow, sig, cmap='viridis', vmin=None, vmax=None, ylim=None, alpha=0.3, kind='std', vColor='r', sigcolor='slateblue', fignum=0)  
Plot realigned time-frequency maps.

**Args:** xpow, sig: output of the get() method. sig can either be the phase or the filtered signal.

#### Kargs:

**cmap: string, optional, [def: 'viridis']** The colormap to use

**vmin, vmax: int/float, otpional, [def: None, None]** Limits of the colorbar

**ylim: tuple/list, optional, [def: None]** Limit for the plot of the signal

**alpha: float, optional, [def: 0.3]** Transparency of deviation/sem

**kind: string, optional, [def: 'std']** Choose between 'std' or 'sem' to either display standard deviation or standard error on the mean for the signal plot

**vColor: string, optional, [def: 'r']** Color of the vertical line which materialized the choosen cue

**sigcolor: string, optional, [def: 'slateblue']** Color of the signal

**fignum: integer, optional, [def: 0]** Number of the figure

**Returns:** figure, axes1 (TF plot), axes2 (signal plot), axes3 (colorbar)

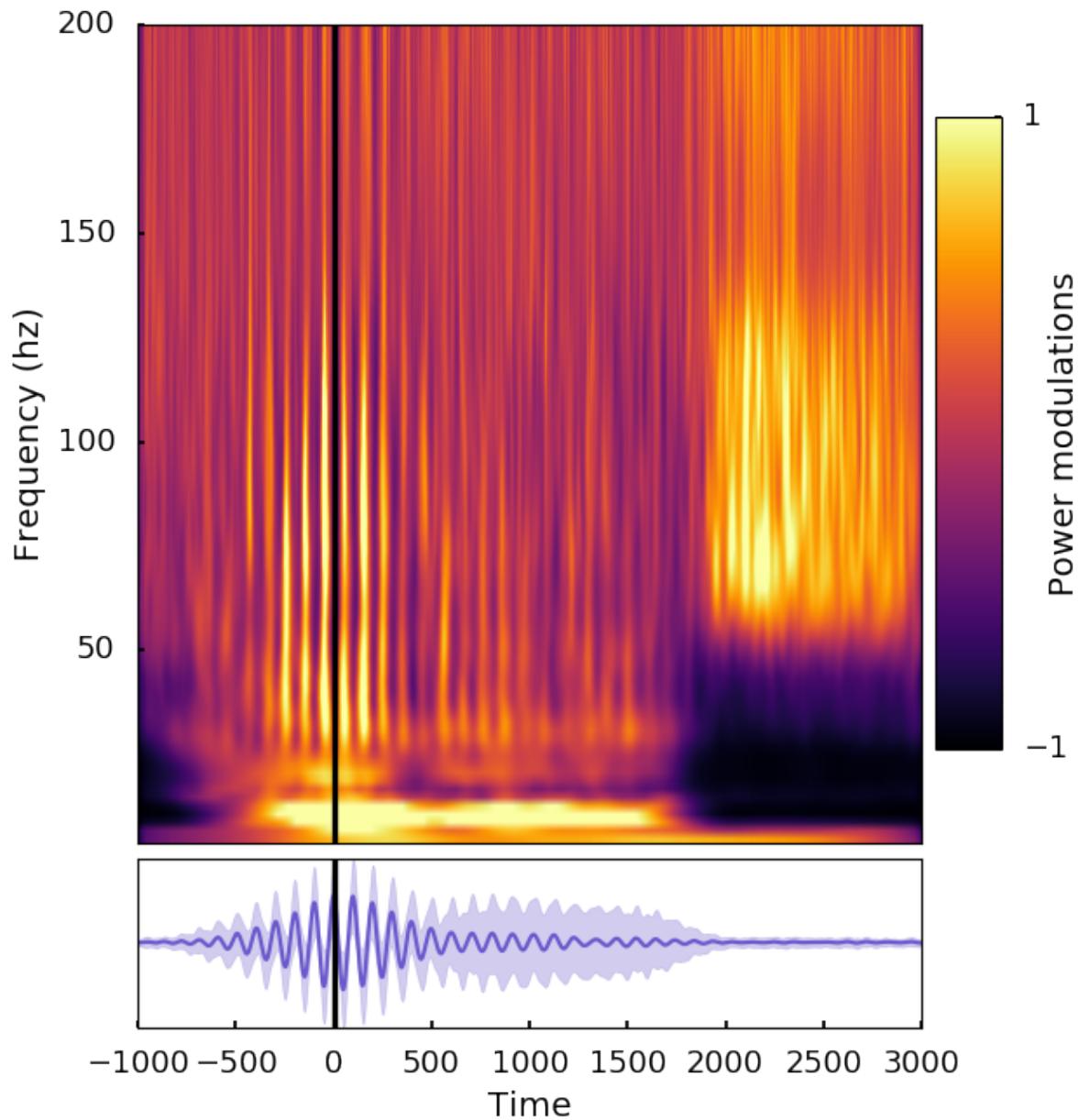


Fig. 4.5: Event Related Phase-Amplitude

#### 4.5.4 Event Related Phase-Amplitude Coupling

```
class feature.erpac(sf, npts, pha_f=[2, 4], pha_meth='hilbert', pha_cycle=3, amp_f=[60, 200],
                    amp_meth='hilbert', amp_cycle=6, window=None, step=None, width=None,
                    time=None, **kwargs)
```

Compute Event Related Phase-Amplitude coupling. See <sup>6</sup>

**Args:**

**sf: int** Sampling frequency

**npts: int** Number of points of the time serie

**Kargs:**

**pha\_f: tuple/list, optional, [def: [2,4]]** List containing the couple of frequency bands for the phase. Example: f=[ [2,4], [5,7], [60,250] ]

**pha\_meth: string, optional, [def: 'hilbert']** Method for the phase extraction.

**pha\_cycle: integer, optional, [def: 3]** Number of cycles for filtering the phase.

**amp\_f: tuple/list, optional, [def: [60,200]]** List containing the couple of frequency bands for the amplitude. Each couple can be either a list or a tuple.

**amp\_meth: string, optional, [def: 'hilbert']** Method for the amplitude extraction.

**amp\_cycle: integer, optional, [def: 6]** Number of cycles for filtering the amplitude.

**window: tuple/list/None, optional [def: None]** List/tuple: [100,1500] List of list/tuple: [(100,500),(200,4000)] Width and step parameters will be ignored.

**width: int, optional [def: None]** width of a single window.

**step: int, optional [def: None]** Each window will be spaced by the “step” value.

**time: list/array, optional [def: None]** Define a specific time vector

**get(xpha, xamp, n\_perm=200, n\_jobs=-1)**

Get the erpac mesure between an xpha and xamp signals.

**Args:**

**xpha: array** Signal for phase. The shape of xpha should be : (n\_electrodes x n\_pts x n\_trials)

**xamp: array** Signal for amplitude. The shape of xamp should be : (n\_electrodes x n\_pts x n\_trials)

**Kargs:**

**n\_perm: integer, optional, [def: 200]** Number of permutations for normalizing the cfc mesure.

**n\_jobs: integer, optional, [def: -1]** Control the number of jobs for parallel computing. Use 1, 2, .. depending of your number of cores. -1 for all the cores.

If the same signal is used (example : xpha=x and xamp=x), this mean the program compute a local erpac.

**Returns:**

**xerpac: array** The erpac mesure of size : (n\_amplitude x n\_phase x n\_electrodes x n\_windows)

**pvalue: array** The associated p-values of size : (n\_amplitude x n\_phase x n\_electrodes x n\_windows)

---

<sup>6</sup> Voytek et al, 2013

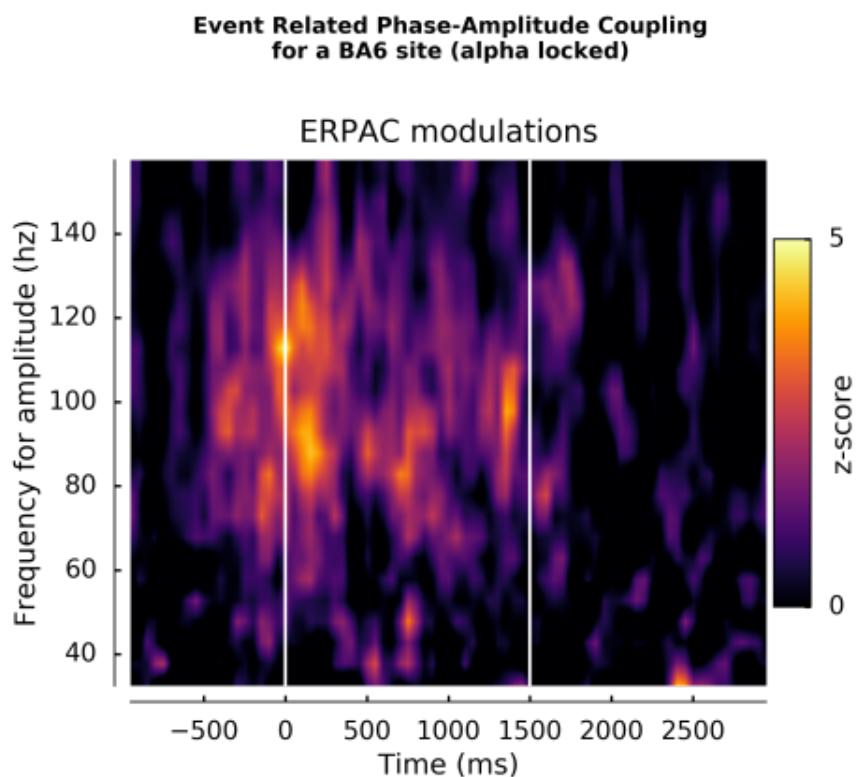


Fig. 4.6: Event Related Phase-Amplitude Coupling

## 4.5.5 Phase-Locking Value

**class feature.PLV(sf, npts, f=[2, 4], method='hilbert', cycle=3, sample=None, time=None, \*\*kwargs)**  
Compute the Phase-Locking Value<sup>7</sup>

### Args:

**sf: int** Sampling frequency  
**npts: int** Number of points of the time serie

### Kargs:

**f: tuple/list, optional, [def: [2,4]]** List containing the couple of frequency bands for the phase. Example:  
f=[ [2,4], [5,7], [60,250] ]  
**method: string, optional, [def: 'hilbert']** Method for the phase extraction.  
**cycle: integer, optional, [def: 3]** Number of cycles for filtering the phase.  
**sample: list, optional, [def: None]** Select samples in the time series to compute the plv  
**time: list/array, optional [def: None]** Define a specific time vector  
**amp\_cycle: integer, optional, [def: 6]** Number of cycles for filtering the amplitude.

**get(xelec1, xelec2, n\_perm=200, n\_jobs=-1)**  
Get Phase-Locking Values for a set of distant sites

### Args:

**xelec1, xelec2: array** PLV will be compute between xelec1 and xelec2. Both matrix contains times-series of each trial per electrode. It's not forced that both have the same size but they must have at least the same number of time points (npts) and trials (ntrials). [xelec1] = (n\_elecs, npts, ntrials), [xelec2] = (n\_elecs, npts, ntrials)

### Kargs:

**n\_perm: int, optional, [def: 200]** Number of permutations to estimate the statistical signifiacy of the plv mesure  
**n\_jobs: integer, optional, [def: -1]** Control the number of jobs for parallel computing. Use 1, 2, .. depending of your number of cores. -1 for all the cores.

### Returns:

**plv: array** The plv mesure for each phase and across electrodes of size: [plv] = (n\_phases, n\_elecs, n\_sample)  
**pvalues: array** The p-values with the same shape of plv

## 4.6 PSD based features

Those following features are extracted using a Power Spectrum Density (PSD)

### 4.6.1 Power Spectrum Density

**class feature.PSD(sf, npts, step=None, width=None, time=None)**  
Compute the power spectral density of multiple electrodes.

<sup>7</sup> Lachaux et al, 1999

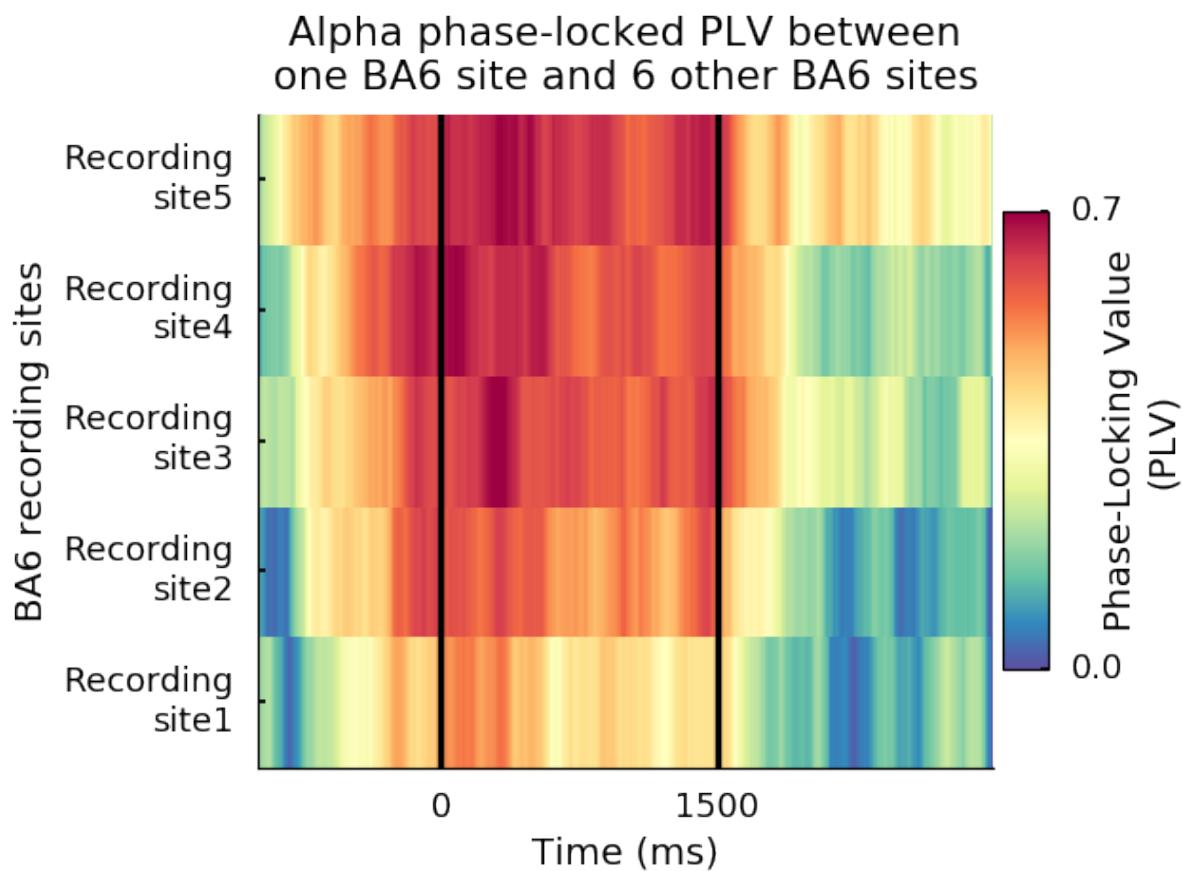


Fig. 4.7: Example of an alpha PLV

**Args:**

**sf: int** Sampling frequency  
**npts: int** Number of points of the time serie  
**width: int, optional [def: None]** width of a single window.  
**step: int, optional [def: None]** Each window will be spaced by the “step” value.  
**time: list/array, optional [def: None]** Define a specific time vector

**get** (*x*, \*\**kwargs*)

Get the PSD

**Args:**

**x: array** Data to get PSD. *x* can be a vector of (npts), a matrix (npts, ntrials) or 3D array (nelectrodes, npts, ntrials).

**Kargs:** *kwargs*: any supplementar argument are directly passed to the welch function of scipy.

**Returns:** *f*: frequency vector of shape (nfce,)

*amp*: PSD array of shape (nelectrodes, nfce, nwin, ntrials)

## 4.6.2 Power PSD

**class feature.powerPSD (sf, npts, f=[60, 200], step=None, width=None, time=None)**  
Compute the power based on psd of multiple electrodes.

**Args:**

**sf: int** Sampling frequency  
**npts: int** Number of points of the time serie  
**width: int, optional [def: None]** width of a single window.  
**step: int, optional [def: None]** Each window will be spaced by the “step” value.  
**time: list/array, optional [def: None]** Define a specific time vector  
**f: tuple/list** List containing the couple of frequency bands to extract spectral informations. Alternatively, *f* can be define with the form *f*=(fstart, fend, fwidth, fstep) where fstart and fend are the starting and endind frequencies, fwidth and fstep are the width and step of each band.

```
>>> # Specify each band:  
>>> f = [ [15, 35], [25, 45], [35, 55], [45, 60], [55, 75] ]  
>>> # Second option:  
>>> f = (15, 75, 20, 10)
```

**get** (*x*, \*\**kwargs*)

Get the PSD power

**Args:**

**x: array** Data to get PSD power. *x* can be a vector of (npts), a matrix (npts, ntrials) or 3D array (nelectrodes, npts, ntrials).

**Kargs:** *kwargs*: any supplementar argument are directly passed to the welch function of scipy.

**Return:** *xpow*: power array of shape (nfce, nelectrodes, nwin, ntrials)

### 4.6.3 Spectral entropy

```
class feature.SpectralEntropy(sf, npts, step=None, width=None, time=None)
    Compute the spectral entropy based on psd of multiple electrodes.
```

**Args:**

**sf: int** Sampling frequency

**npts: int** Number of points of the time serie

**width: int, optional [def: None]** width of a single window.

**step: int, optional [def: None]** Each window will be spaced by the “step” value.

**time: list/array, optional [def: None]** Define a specific time vector

**get** (x, \*\*kwargs)

Get the spectral entropy

**Args:**

**x: array** Data to get spectral entropy. x can be a vector of (npts), a matrix (npts, ntrials) or 3D array (nelectrodes, npts, ntrials).

**Kargs:** kwargs: any supplementar argument are directly passed to the welch function of scipy.

**Return:** xent: spectral entropy of x of shape (nelectrodes, nwin, ntrials)

## 4.7 Tools

### 4.7.1 Physiological bands

```
feat.featools.bandRef(return_as='table')
    Get the traditionnal physiological frequency bands of interest
```

**Args:**

**return\_as: string, optional, [def: ‘table’]** Say how to return bands. Use either ‘table’ to get a pandas Dataframe or ‘list’ to have two list containing bands name and definition.

```
feat.featools.findBandName(band)
    Find the physiological name of a frequency band
```

**Args:**

**band: list** List of frequency bands

**Returns:** List of band names

**Example:**

```
>>> band = [[13, 60], [0, 1], [5, 7]]
>>> findBandName(band)
>>> ['Low-gamma', 'VLFC', 'Theta']
```

```
feat.featools.findBandFcy(band)
    Find the physiological frequency band for a given name
```

**Args:**

**band: list** List of frequency band name

**Returns:** List of frequency bands

### 4.7.2 Cross-frequency vector

```
feat.featoools.cfcVec (pha=(2, 30, 2, 1), amp=(60, 200, 10, 5))
Generate cross-frequency coupling vectors.
```

**Kargs:**

- pha: tuple, optional, [def: (2, 30, 2, 1)]** Frequency parameters for phase. Each argument inside the tuple mean (starting fcy, ending fcy, width, step)
- amp: tuple, optional, [def: (60, 200, 10, 5)]** Frequency parameters for amplitude. Each argument inside the tuple mean (starting fcy, ending fcy, width, step)

**Returns:**

- pVec: array** Centered-frequency vector for the phase
- aVec: array** Centered-frequency vector for the amplitude
- pTuple: list** List of tuple. Each tuple contain the (starting, ending) frequency for phase.
- aTuple: list** List of tuple. Each tuple contain the (starting, ending) frequency for amplitude.

### 4.7.3 Simulation

```
feat.featoools.cfcRndSignals (fPha=2, fAmp=100, sf=1024, ndatasets=10, tmax=1, chi=0,
noise=1, dPha=0, dAmp=0)
Generate randomly phase-amplitude coupled signals.
```

**Kargs:**

- fPha: int/float, optional, [def: 2]** Frequency for phase
- fAmp: int/float, optional, [def: 100]** Frequency for amplitude
- sf: int, optional, [def: 1024]** Sampling frequency
- ndatasets** [int, optional, [def: 10]] Number of datasets
- tmax: int/float (1<=tmax<=3), optional, [def: 1]** Length of the time vector. If tmax=2 and sf=1024, the number of time points npts=1024\*2=2048
- chi: int/float (0<=chi<=1), optional, [def: 0]** Amout of coupling. If chi=0, signals of phase and amplitude are strongly coupled.
- noise: int/float (1<=noise<=3), optional, [def: 1]** Amount of noise
- dPha: int/float (0<=dPha<=100), optional, [def: 0]** Introduce a random incertitude on the phase frequency. If fPha is 2, and dPha is 50, the frequency for the phase signal will be between : [2-0.5\*2, 2+0.5\*2]=[1,3]
- dAmp: int/float (0<=dAmp<=100), optional, [def: 0]** Introduce a random incertitude on the amplitude frequency. If fAmp is 60, and dAmp is 10, the frequency for the amplitude signal will be between : [60-0.1\*60, 60+0.1\*60]=[54,66]

**Return:**

- data: array** The randomly coupled signals. The shape of data will be (ndatasets x npts)
- time: array** The corresponding time vector

```
from brainpipe.classification import *
```

## 4.8 Presentation

Ok, let's say you already have extracted features from your neural activity and now, you want to use machine-learning to verify if your features can discriminate some conditions. For example, you want to discriminate conscious versus unconscious people using alpha power on 64 EEG electrodes. Your data can be organized like this:

```
# Consider that conscious data have 150 trials and 130 for unconscious. So if you
# print the shape of both, you'll have :
print(conscious_data.shape, unconscious_data.shape)
# (150, 67), (130, 67)
# Let's build your data matrix by concatenating along the trial dimension:
x = np.concatenate((conscious_data, unconscious_data), axis=0)
print('New shape of x: ', x.shape)
# New shape of x: (280, 67)
# Now, build your label vector to indicate to machine learning
# which trial belong to which condition. We are going to use
# 0 for conscious / 1 for unconscious. Finally, the label vector
# will have the same length as the number of trials in x :
y = [0]*conscious_data.shape[0] + [1]*unconscious_data.shape[0]
```

Now, we have the concatenated data and the label vector. To start using machine learning, we need two things:

- a classifier
- a cross-validation

In brainpipe, use defClf() to construct your classifier. Use defCv() to construct the cross-validation. Finally, the classify() function will link these two objects in order to classify your conditions.

```
# Define a 50 times 5-folds cross-validation :
cv = defCv(y, cvtype='kfold', rep=50, n_folds=5)
# Define a Random Forest with 200 trees :
clf = defClf(y, clf='rf', n_tree=200, random_state=100)
# Past the two objects inside classify :
clfObj = classify(y, clf=clf, cvtype=cv)
# Evaluate the classifier on data:
da = clfObj.fit(x)
```

## 4.9 Define a classifier

```
class classification.defClf(y, clf='lda', kern='rbf', n_knn=10, n_tree=100, priors=False,
                             **kwargs)
```

Choose a classifier and switch easily between classifiers implemented in scikit-learn.

**Args:**

**y: array** The vector label

**clf: int or string, optional, [def: 0]** Define a classifier. Use either an integer or a string. Choose between:

- 0 / 'lda': Linear Discriminant Analysis (LDA)
- 1 / 'svm': Support Vector Machine (SVM)
- 2 / 'linearsvm' : Linear SVM

- 3 / ‘nusvm’: Nu SVM
- 4 / ‘nb’: Naive Bayesian
- 5 / ‘knn’: k-Nearest Neighbor
- 6 / ‘rf’: Random Forest
- 7 / ‘lr’: Logistic Regression
- 8 / ‘qda’: Quadratic Discriminant Analysis

**kern:** string, optional, [def: ‘rbf’] Kernel of the ‘svm’ classifier

**n\_knn:** int, optional, [def: 10] Number of neighbors for the ‘knn’ classifier

**n\_tree:** int, optional, [def: 100] Number of trees for the ‘rf’ classifier

**Kargs:** optional arguments. To define other parameters, see the description of scikit-learn.

**Return:** A scikit-learn classification objects with two supplementar arguments :

- lgStr : long description of the classifier
- shStr : short description of the classifier

## 4.10 Define a cross-validation

```
class classification.defCv(y, cvtype='skfold', n_folds=10, rndstate=0, rep=10, **kwargs)
Choose a cross_validation (CV) and switch easly between CV implemented in scikit-learn.
```

**Args:**

**y:** array The vector label

**kargs:**

**cvtype:** string, optional, [def: skfold] Define a cross\_validation. Choose between :

- ‘skfold’: Stratified k-Fold
- ‘kfold’: k-fold
- ‘sss’: Stratified Shuffle Split
- ‘ss’: Shuffle Split
- ‘loo’: Leave One Out
- ‘lolo’: Leave One Label Out

**n\_folds:** integer, optional, [def: 10] Number of folds

**rndstate:** integer, optional, [def: 0] Define a random state. Usefull to replicate a result

**rep:** integer, optional, [def: 10] Number of repetitions

**kwargs:** optional arguments. To define other parameters, see the description of scikit-learn.

**Return:** A list of scikit-learn cross-validation objects with two supplementar arguments:

- lgStr: long description of the cross\_validation
- shStr: short description of the cross\_validation

## 4.11 Classify

**class** `classification.classify` (`y, clf='lda', cvtype='skfold', clfArg={}, cvArg={}`)

Define a classification object and apply to classify data. This class can be consider as a centralization of scikit-learn tools, with a few more options.

To classify data, two objects are necessary : - A classifier object (lda, svm, knn...) - A cross-validation object which is used to validate a classification performance. This two objects can either be defined before the classify object with defCv and defClf, or they can be directly defined inside the classify class.

**Args:**

**y: array** The vector label

**Kwargs:**

**clf: int / string / classifier object, optional, [def: 0]** Define a classifier. If clf is an integer or a string, the classifier will be defined inside classify. Otherwise, it is possible to define a classifier before with defClf and past it in clf.

**cvtype: string / cross-validation object, optional, [def: 'skfold']** Define a cross-validation. If cvtype is a string, the cross-validation will be defined inside classify. Otherwise, it is possible to define a cross-validation before with defCv and past it in cvtype.

**clfArg: dictionary, optional, [def: {}]** This dictionary can be used to define supplementar arguments for the classifier. See the documentation of defClf.

**cvArg: dictionary, optional, [def: {}]** This dictionary can be used to define supplementar arguments for the cross-validation. See the documentation of defCv.

**Example:**

```
>>> # 1) Define a classifier and a cross-validation before classify():
>>> # Define a 50 times 5-folds cross-validation :
>>> cv = defCv(y, cvtype='kfold', rep=50, n_folds=5)
>>> # Define a Random Forest with 200 trees :
>>> clf = defClf(y, clf='rf', n_tree=200, random_state=100)
>>> # Past the two objects inside classify :
>>> clfObj = classify(y, clf=clf, cvtype=cv)
```

```
>>> # 2) Define a classifier and a cross-validation inside classify():
>>> clfObj = classify(y, clf = 'rf', cvtype = 'kfold',
>>>                 clfArg = {'n_tree':200, 'random_state':100},
>>>                 cvArg = {'rep':50, 'n_folds':5})
>>> # 1) and 2) are equivalent. Then use clfObj.fit() to classify data.
```

**cm** (`normalize=True`)

Get the confusion matrix of each feature.

**Kargs:**

**normalize: bool, optional, [def: True]** Normalize or not the confusion matrix

**update: bool, optional, [def: True]** If update is True, the data will be re-classified. But, if update is set to False, and if the methods .fit() or .fit\_stat() have been run before, the data won't be re-classified. Instead, the labels previously found will be used to get confusion matrix.

**Return:**

**CM: array** Array of confusion matrix of shape (n\_features x n\_class x n\_class)

---

**fit** (*x, mf=False, center=False, grp=None, method='bino', n\_perm=200, rndstate=0, n\_jobs=-1*)  
Apply the classification and cross-validation objects to the array *x*.

**Args:**

**x: array** Data to classify. Consider that *x.shape* = (N, M), N is the number of trials (which should be the length of *y*). M, the number of columns, is a supplemental dimension for classifying data. If M = 1, the data is consider as a single feature. If M > 1, use the parameter *mf* to say if *x* should be consider as a single feature (*mf=False*) or multi-features (*mf=True*)

**Kargs:**

**mf: bool, optional, [def: False]** If *mf=False*, the returned decoding accuracy (*da*) will have a shape of (1, rep) where *rep* is the number of repetitions. This mean that all the features are used together. If *mf=True*, *da.shape* = (M, rep), where M is the number of columns of *x*.

**center: optional, bool, [def: False]** Normalize fatures with a zero mean by substracting then dividig by the mean. The center parameter should be set to True if the classifier is a svm.

**grp: array, optional, [def: None]** If *mf=True*, the *grp* parameter allow to define group of features. If *x.shape* = (N, 5) and *grp=np.array([0,0,1,2,1])*, this mean that 3 groups of features will be considered : (0,1,2)

**method: string, optional, [def: 'bino']** Four methods are implemented to test the statistical significance of the decoding accuracy :

- 'bino': binomial test
- 'label\_rnd': randomly shuffle the labels
- 'full\_rnd': randomly shuffle the whole array *x*
- 'intra\_rnd': randomly shuffle *x* inside each class and each feature

Methods 2, 3 and 4 are based on permutations. The method 2 and 3 should provide similar results. But 4 should be more conservative.

**n\_perm: integer, optional, [def: 200]** Number of permutations for the methods 2, 3 and 4

**rndstate: integer, optional, [def: 0]** Fix the random state of the machine. Usefull to reproduce results.

**n\_jobs: integer, optional, [def: -1]** Control the number of jobs to cumpute the decoding accuracy. If *n\_jobs* = -1, all the jobs are used.

**Return:**

**da: array** The decoding accuracy of shape *n\_repetitions* x *n\_features*

**pvalue: array** Array of associated pvalue of shape *n\_features*

**daPerm: array** Array of all the decodings obtained for each permutations of shape *n\_perm* x *n\_features*

## 4.12 Leave p-subjects out

---

**class classification.LeavePSubjectOut** (*y, nsuj, pout=1, clf='lda', \*\*clfArg*)  
Leave p-subject out cross-validation

**Args:**

**y: list** List of label vectors for each subject.

**nsuj: int** Number of subjects

**Kargs:**

**pout: int** Number of subjects to leave out for testing. If pout=1, this is a leave one-subject out

**clf: int / string / classifier object, optional, [def: 0]** Define a classifier. If clf is an integer or a string, the classifier will be defined inside classify. Otherwise, it is possible to define a classifier before with defClf and pass it in clf.

**clfArg: supplementar arguments** This dictionnary can be used to define supplementar arguments for the classifier. See the documentation of defClf.

**change\_clf (clf='lda', \*\*clfArg)**

Change the classifier

**fit (x, mf=False, center=False, grp=None, method='bino', n\_perm=200, rndstate=0, n\_jobs=-1)**

Apply the classification and cross-validation objects to the array x.

**Args:**

**x: list** List of dataset for each subject. All the dataset in the list should have the same number of columns but the number of lines could be different for each subject and must correspond to the same number of lines each each label vector of y.

**Kargs:**

**mf: bool, optional, [def: False]** If mf=False, the returned decoding accuracy (da) will have a shape of (1, rep) where rep, is the number of repetitions. This mean that all the features are used together. If mf=True, da.shape = (M, rep), where M is the number of columns of x.

**center: optional, bool, [def: False]** Normalize fatures with a zero mean by substracting then dividng by the mean. The center parameter should be set to True if the classifier is a svm.

**grp: array, optional, [def: None]** If mf=True, the grp parameter allow to define group of features. If x.shape = (N, 5) and grp=np.array([0,0,1,2,1]), this mean that 3 groups of features will be considered : (0,1,2)

**method: string, optional, [def: 'bino']** Four methods are implemented to test the statistical significance of the decoding accuracy :

- 'bino': binomial test
- 'label\_rnd': randomly shuffle the labels

Methods 2 and 3 are based on permutations. They should provide similar results. But 4 should be more conservative.

**n\_perm: integer, optional, [def: 200]** Number of permutations for the methods 2, 3 and 4

**rndstate: integer, optional, [def: 0]** Fix the random state of the machine. Usefull to reproduce results.

**n\_jobs: integer, optional, [def: -1]** Control the number of jobs to cumpute the decoding accuracy. If n\_jobs = -1, all the jobs are used.

**Return:**

**da: array** The decoding accuracy of shape n\_repetitions x n\_features

**pvalue: array** Array of associated pvalue of shape n\_features

**daPerm: array** Array of all the decodings obtained for each permutations of shape n\_perm x n\_features

## 4.13 Generalization

**class** `classification.generalization` (`time, y, x, clf='lda', cvtype=None, clfArg={}, cvArg={}`)  
 Generalize the decoding performance of features. The generalization consist of training and testing at diffrents moments. The use is to see if a feature is consistent and performant in diffrents period of time.

### Args:

**time: array/list** The time vector of dimension npts

**y: array** The vector label of dimension ntrials

**x: array** The data to generalize. If x is a 2D array, the dimension of x should be (ntrials, npts). If x is 3D array, the third dimension is consider as multi-features. This can be usefull to do time generalization in multi-features.

### Kargs:

**clf: int / string / classifier object, optional, [def: 0]** Define a classifier. If clf is an integer or a string, the classifier will be defined inside classify. Otherwise, it is possible to define a classifier before with defClf and past it in clf.

**cvtype: string / cross-validation object, optional, [def: None]** Define a cross-validation. If cvtype is None, the diagonal of the matrix of decoding accuracy will be set at zero. If cvtype is defined, a cross-validation will be performed on the diagonal. If cvtype is a string, the cross-validation will be defined inside classify. Otherwise, it is possible to define a cross-validation before with defCv and past it in cvtype.

**clfArg: dictionary, optional, [def: {}]** This dictionary can be used to define supplementar arguments for the classifier. See the documentation of defClf.

**cvArg: dictionary, optional, [def: {}]** This dictionary can be used to define supplementar arguments for the cross-validation. See the documentation of defCv.

**Return:** An array of dimension (npts, npts) containing the decoding accuracy. The y axis is the training time and the x axis is the testing time (also known as “generalization time”)

## 4.14 Multi-features

**class** `classification.mf` (`y, Id='0', p=0.05, n_perm=200, stat='bino', threshold=None, nbest=10, direction='forward', occurrence='i%', clfIn={'clf': 'lda'}, clfOut={'clf': 'lda'}, cvIn={'rep': 1, 'n_folds': 10, 'cvtype': 'skfold'}, cvOut={'rep': 10, 'n_folds': 10, 'cvtype': 'skfold'}`)

Compute multi-features (mf) with the possibility of using methods in cascade and run the mf on particular groups.

### Args:

**y: array-like** The target variable to try to predict in the case of supervised learning

### Kargs:

**Id: string, optional, [def: '0']** Use this parameter to define a cascade of methods. Here is the list of the current implemented methods:

- ‘0’: No selection. All the features are used
- ‘1’: Select <p significant features using either a binomial law or permutations
- ‘2’: select ‘nbest’ features

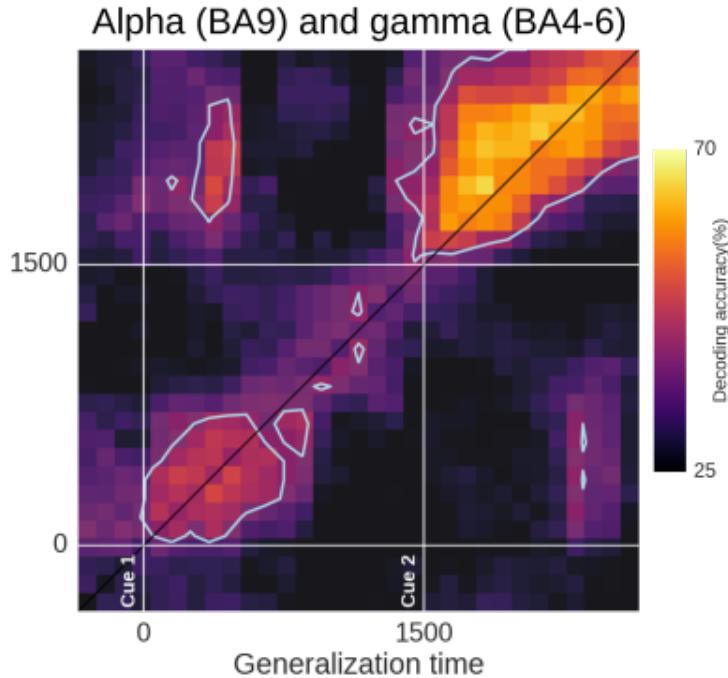


Fig. 4.8: Time-generalization using two features (alpha and gamma power)

- ‘3’: use ‘forward’/‘backward’/‘exhaustive’ to select features

If is for example Id=‘12’, the program will first select significant features, then, on this subset, it will find the nbest features. All the methods can be serialized.

**p: float, optional, [def: 0.05]** The pvalue to select features for the Id=‘1’ method

**n\_perm: integer, optional, [def: 200]** Number of permutations for the Id=‘1’ method

**stat** [string, optional, [def: ‘bino’]] Statistical test for selecting features for the Id=‘1’ method. Choose between:

- ‘bino’: binomial test
- ‘label\_rnd’: randomly shuffle the labels
- ‘full\_rnd’: randomly shuffle the whole array x
- ‘intra\_rnd’: randomly shuffle x inside each class and each feature

Methods 2, 3 and 4 are based on permutations. The method 2 and 3 should provide similar results. But 4 should be more conservative.

**threshold: integer/float, optional, [def: None]** Define a decoding accuracy for thresholding features. equivalent to the p parameter.

**nbest: integer, optional, [def: 10]** For the Id=‘2’, use this parameter to control the number of features to select. If nbest=10, the program will classify each feature and then select the 10 best of them.

**direction: string, optional, [def: ‘forward’]** For the method Id=‘3’, use:

- ‘forward’
- ‘backward’
- ‘exhaustive’

to control the direction of the feature selection.

**occurence: string, optional, [def: ‘i%’]** Use this parameter to modify the way of visualizing the occurrence of each feature apparition. Choose between :

- ‘%’ : in percentage (float)
- ‘i%’ : in integer percentage (int)
- ‘c’ : count (= number of times the feature has been selected)

**clfIn // clfOut** [dictionary, optional] Use those dictionnaries to control the classifier to use.

- clfIn : the classifier use for the training [def: LDA]
- clfOut : the classifier use for the testing [def: LDA]

To have more controlable classifiers, see the defClf() class inside the classification module.

**cvIn // cvOut** [dictionary, optional] Use those dictionnaires to control the cross-validations (cv) to use.

- **cvIn** [the cv to use for the training [def: 1 time stratified] 10-folds]
- **cvOut** [the more extern cv, to separate training and testing and] to avoid over-fitting [def: 10 time stratified 10-folds]

To have more controlable cross-validation, see the defCv() class inside the classification module.

**Return** A multi-features object with a fit() method to apply to model to the data.

**fit** (*x*, *grp*=[], *center*=*False*, *combine*=*False*, *grpas*=‘single’, *grplen*=[], *display*=*True*, *n\_jobs*=-1)

Run the model on the matrix of features *x*

**Args:**

**x: array-like** The features. Dimension [n trials x n features]

**Kargs:**

**grp: list of strings, optional, [def: []]** Group features by using a list of strings. The length of grp must be the same as the number of features. If grp is not empty, the program will run the feature selection inside each group.

**center: optional, bool, [def: False]** Normalize fatures with a zero mean by substracting then dividng by the mean. The center parameter should be set to True if the classifier is a svm.

**combine: boolean, optional, [def: False]** If a group of features is specified using the grp parameter, combine give the access of combining or not groups. For example, if there is three unique groups, combining them will compute the mf model on each combination : [[1],[2],[3],[1,2],[1,3],[2,3],[1,2,3]]

**grpas: string, optional, [def: ‘single’]** Specify how to consider features inside each group. If the parameter grpas (“group as”) is:

- ‘single’: inside each combination of group, the features are considered as independant.
- ‘group’: inside each combination of group, the features are going to be associated. So the mf model will search to add a one by one feature, but it will add groups of features.

**grplen: list, optional, [def: []]** Control the number of combinations by specifying the number of elements to associate. If there is three unique groups, all possible combinations are : [[1],[2],[3],[1,2],[1,3],[2,3],[1,2,3]] but if grplen is specify, for example grplen=[1,3], this will consider combinations of groups only with a length of 1 and 3 and remove combinations of 2 elements: [[1],[2],[3],[1,2,3]]

**display: boolean, optional, [def: True]** Display informations for each step of the mf selection. If n\_jobs is -1, it is advise to set the display to False.

**n\_jobs: integer, optional, [def: -1]** Control the number of jobs to cumpute the decoding accuracy. If n\_jobs=-1, all the jobs are used.

**Returns:**

**da: list** The decoding accuracy (da) for each group with the selected number of repetitions, which by default is set to 10 (see : cvOut // rep)

**prob: list** The appearance probability of each feature. The size of prob is the same as da.

**groupinfo: pandas Dataframe** Dataframe to resume the mf feature selection.

**class classification.MFpipe(y, cv, random\_state=0)**

Define a pipeline of multi-features

**Args:**

**y: ndarray** Vector label

**cv: sklearn.cross\_validation, optional, (def: default)** An external cross-validation to split in training and testing to validate the pipeline without overfitting.

**random\_state: inetege, optional, (def: 0)** Fix the random state of the machine for reproducibility.

**Return** Multi-features pipeline object

**custom\_pipeline(pipeline=None, grid=None)**

Send a custom pipeline

**Kargs:**

**pipeline: sklearn.Pipeline, optional, (def: None)** The pipeline to use

**grid: sklearn.GridCv, optional, (def: None)** A grid for parameters optimisation

**default\_pipeline(name, n\_pca=10, n\_best=10, lda\_shrink=10, svm\_C=10, svm\_gamma=10, fdr\_alpha=[0.05], fpr\_alpha=[0.05])**

Use a default combination of parameters for building a pipeline

**Args:**

**name: string** The string for building a default pipeline (see examples below)

**Kargs:**

**n\_pca: integer, optional, (def: 10)** The number of components to search

**n\_best: integer, optional, (def: 10)** Number of best features to consider using a statistical method

**lda\_shrink: integer, optional, (def: 10)** Fit optimisation parameter for the lda

**svm\_C/svm\_gamma: integer, optional, (def: 10/10)** Parameters to optimize for the svm

**fdr/fpr\_alpha: list, optional, (def: [0.05])** List of float for selecting features using a fdr or fpr

**Examples:**

```
>>> # Basic classifiers :
>>> name = 'lda' # or name = 'svm_linear' for a linear SVM
>>> # Combine a classifier with a feature selection method :
>>> name = 'lda_fdr_fpr_kbest_pca'
>>> # The method above will use an LDA for the features evaluation
>>> # and will combine a FDR, FPR, k-Best and pca feature seelction.
>>> # Now we can combine with classifier optimisation :
```

```
>>> name = 'lda_optimized_pca' # will try to optimize an LDA with a pca
>>> name = 'svm_kernel_C_gamma_kbest' # optimize a SVM by trying
>>> # different kernels (linear/RBF), and optimize C and gamma parameters
>>> # combine with a k-Best features selection.
```

**fit**(*x, name=None, rep=1, n\_iter=5, n\_jobs=1, verbose=0*)

Apply the pipeline.

**Args:**

**x: ndarray** Array of features organize as (n\_trials, n\_features)

**Kargs:**

**name: ndarray, optional, (def: None)** Array of string with the same length as the number of features

**rep: integer, optional, (def: 1)** Number of repetitions for the whole pipeline.

**n\_iter: integer, optional, (def: 5)** Number of iterations in order to find the best set of parameters

**n\_jobs: integer, optional, (def: 1)** Number of jobs for parallel computing. Use -1 for all jobs.

**verbose: integer, optional, (def: 0)** Control displaying state

**Return** da: the final vector of decoding accuracy of shape (n\_repetitions,)

**get\_grid**(*n\_splits=3, n\_jobs=1*)

Return the cross-validated grid search object

**Kargs:**

**n\_splits: int, optional, (def: 3)** The number of folds for the cross-validation

**n\_jobs: int, optional, (def: 1)** Number of jobs to use for parallel processing

**Return:** grid: a sklearn.GridSearchCV object with the defined pipeline

**selected\_features()**

Get the number of times a feature was selected

```
from brainpipe.statistics import *
```

- *Binomial*
- *Permutations*
- *Multiple-comparisons*
- *Circular statistics toolbox*

## 4.15 Binomial

**statistics.bino\_da2p**(*y, da*)

For a given vector label, get p-values of a decoding accuracy using the binomial law.

**Args:**

**y** [array] The vector label

**da: int / float / list /array [0 <= da <= 100]** The decoding accuracy array. Ex : da = [75, 33, 25, 17].

**Return:**

**p: ndarray** The p-value associate to each decoding accuracy

```
statistics.bino_p2da(y, p)
```

For a given vector label, get the decoding accuracy of p-values using the binomial law.

**Args:**

**y: array** The vector label

**p: int / float / list / array [0 <= p < 1]** p-value. Ex : p = [0.05, 0.01, 0.001, 0.00001]

**Return:**

**da: ndarray** The decoding accuracy associate to each p-value

```
statistics.bino_signifeat(feat, th)
```

Get significants features.

**Args:**

**feat: array** Array containing either decoding accuracy (da) either p-values

**th: int / float** The threshold in order to find significants features.

**Return:**

**index: array** The index corresponding to significants features

**signi\_feat: array** The significants features

## 4.16 Permutations

### 4.16.1 Evaluation

```
statistics.perm_2pvalue(data, perm, n_perm, threshold=None, tail=2)
```

Get the associated p-value of a permutation distribution

**Arg:**

**data: array** Array of real data. The shape must be (d1, d2, ..., dn)

**perm: array** Array of permutations. The shape must be (n\_perm, d1, d2, ..., dn)

**n\_perm: int** Number of permutations

**Kargs:**

**threshold: int / float, optional, [def: None]** Every values upper to threshold are going to be set to 1.

**tail: int, optional, [def: 2]** Define if the calculation of p-value must take into account one or two tails of the permutation distribution

**Return:**

**pvalue** [array] Array of associated p-values

```
statistics.perm_metric(metric)
```

Get a metric for permutation normalization.

**Args:** metric: string/function

- None: compare directly values without transformation
- ‘m\_center’: (A-B)/mean(B) transformation
- ‘m\_zscore’: (A-B)/std(B) transformation

- ‘m\_minus’: (A-B) transformation
- function: user defined function [def myfcn(A, B): return array\_like]

`statistics.perm_pvalue2level(perm, p=0.05, maxst=False)`

Get level from which you can consider that it's p-significant;

**Args:**

**perm:** Array of permutations of shape (n\_perm, d1, d2, ..., dn)

**Kargs:**

**p: float, optional, [def: 0.05]** p-value to search in permutation distribution

**maxst: bool, optional, [def: False]** Correct permutations with maximum statistics

**Return:**

**level: float** The level from which you can consider your results as p-significants using permutations

## 4.16.2 Generate

`statistics.perm_rndDatasets(mu=0, sigma=1, dmu=0.1, dsigma=0.1, size=(5, 5), rndstate=0)`

Generate data and permutations uniformly distributed.

**Kargs:**

**mu: int/float** Center of the distribution

**sigma: int/float** Deviation of the distribution

**dmu: int/float** Introduce a shift to the mean of data

**dsigma: int/float** Introduce a shift to the deviation of data

**size: tuple** Size of the generated data and permutations (d1, d2, ..., d3)

**rndstate: int** Fix the random state of the machine

**Returns:**

**data: array** Simulated data of shape (n\_perm, d1, d2, ..., d3)

**perm: array** Simulated permutations of shape (n\_perm, d1, d2, ..., d3)

`statistics.perm_swap(a, b, n_perm=200, axis=-1, rndstate=0)`

Permute values between two arrays and generate a number of permutations.

**Args:**

**a, b: ndarray** Array to swap values. If axis=-1, the shape of a and b could be different. If axis is not -1, the shape of a and b could be different along the specified axis, but must be the same on all other axis.

**n\_perm: int** Number of permutations

**Kargs:**

**axis: int, optional, [def: -1]** Axis for swapping values. If axis is -1, this mean that all values across all dimensions are going to be swap.

**rndstate: int** Fix the random state of the machine

**Return:**

**ash, bsh: array** Swapped arrays

`statistics.perm_rep(x, n_perm)`

Repeat a ndarray x n\_perm times

**Args:**

**x: array** Data to repeat of shape (d1, d2, ..., d3)

**n\_perm: int** Number of permutations

**Returns:**

**xrep: array** Repeated data of shape (n\_perm, d1, d2, ..., d3)

## 4.17 Multiple-comparisons

### 4.17.1 Bonferroni

`statistics.bonferroni(p, axis=-1)`

Bonferroni correction

**Args:**

**p: array** Array of p-values

**Kargs:**

**axis: int, optional, [def: -1]** Axis to apply the Bonferroni correction. If axis is -1, the correction is applied through all dimensions.

**Return:** Corrected pvalues

### 4.17.2 False Discovery Rate (FDR)

`statistics.fdr(p, q)`

False Discovery Rate correction

**Args:**

**p: array** Array of p-values

**q: float** Thresholding p-value

**Return:**

**pcorr: array** Corrected p-values

### 4.17.3 Maximum statistic

`statistics.maxstat(perm, axis=-1)`

Correction by the maximum statistic

**Args:**

**perm: array** The permutations.

**axis: integer, optional, [def: -1]** Use -1 to correct through all dimensions. Otherwise, use d1, d2, ... or dn to correct through a specific dimension.

**Kargs:**

**permR: array** The re-aranged permutations according to the selectionned axis. Then use perm\_2pvalues to get the p-value according to this distribution.

## 4.18 Circular statistics toolbox

Python adaptation of the Matlab toolbox (Berens et al, 2009)

`statistics.circ_corrcc(alpha, x)`  
Correlation coefficient between one circular and one linear random variable.

**Args:**

**alpha: vector** Sample of angles in radians  
**x: vector** Sample of linear random variable

**Returns:**

**rho: float** Correlation coefficient  
**pval: float** p-value

Code taken from the Circular Statistics Toolbox for Matlab By Philipp Berens, 2009 Python adaptation by Etienne Combrisson

`statistics.circ_r(alpha, w=None, d=0, axis=0)`  
Computes mean resultant vector length for circular data.

**Args:**

**alpha: array** Sample of angles in radians

**Kargs:**

**w: array, optional, [def: None]** Number of incidences in case of binned angle data  
**d: radians, optional, [def: 0]** Spacing of bin centers for binned data, if supplied correction factor is used to correct for bias in estimation of r  
**axis: int, optional, [def: 0]** Compute along this dimension

**Return:** r: mean resultant length

Code taken from the Circular Statistics Toolbox for Matlab By Philipp Berens, 2009 Python adaptation by Etienne Combrisson

`statistics.circ_rtest(alpha, w=None, d=0)`

Computes Rayleigh test for non-uniformity of circular data. H0: the population is uniformly distributed around the circle HA: the populatoin is not distributed uniformly around the circle Assumption: the distribution has maximally one mode and the data is sampled from a von Mises distribution!

**Args:**

**alpha: array** Sample of angles in radians

**Kargs:**

**w: array, optional, [def: None]** Number of incidences in case of binned angle data  
**d: radians, optional, [def: 0]** Spacing of bin centers for binned data, if supplied correction factor is used to correct for bias in estimation of r

Code taken from the Circular Statistics Toolbox for Matlab By Philipp Berens, 2009 Python adaptation by Etienne Combrisson

```
from brainpipe.visual import *
```

- *Border plot*
- *p-value plot*
- *tilerplot*

## 4.19 1-D graphics

### 4.19.1 Border plot

```
class visual.BorderPlot(time, x, y=None, kind='sem', color='', alpha=0.2, linewidth=2, legend='',
                        ncol=1, **kwargs)
```

Plot a signal with its associated deviation. The function plots the mean of the signal, and the deviation (std) or standard error on the mean (sem) in transparency.

**Args:**

**time: array/limit** The time vector of the plot (len(time)=N)

**x: numpy array** The signal to plot. One dimension of x must be the length of time N. The other dimension will be considered to define the deviation. For example, x.shape = (N, M)

**Kargs:**

**y: numpy array, optional, [def: None]** Label vector to separate the x signal into different classes. The length of y must be M. If no y is specified, the deviation will be computed for the entire array x. If y is composed of integers Example: y = np.array([1,1,1,1,2,2,2,2]), the function will generate as many curves as the number of unique classes in y. In this case, two curves are going to be considered.

**kind: string, optional, [def: 'sem']** Choose between 'std' for standard deviation and 'sem', standard error on the mean (which is: std(x)/sqrt(N-1))

**color: string or list of strings, optional** Specify the color of each curve. The length of color must be the same as the length of unique classes in y.

**alpha: int/float, optional [def: 0.2]** Control the transparency of the deviation.

**linewidth: int/float, optional, [def: 2]** Control the width of the mean curve.

**legend: string or list of strings, optional, [def: '']** Specify the label of each curve and generate a legend. The length of legend must be the same as the length of unique classes in y.

**ncol: integer, optional, [def: 1]** Number of columns for the legend

**kwargs:** Supplemental arguments to control each subplot: title, xlabel, ylabel (which can be lists for each subplot) xlim, ylim, xticks, yticks, xticklabels, yticklabels, style.

**Return:** The axes of the plot.

### 4.19.2 p-value plot

```
visual.addPval(ax, pval, y=0, x=None, p=0.05, minsucc=1, color='b', shape='-', lw=2, **kwargs)
```

Add significant p-value to an existing plot

**Args:**

**ax: matplotlib axes** The axes to add lines. Use for example plt.gca()

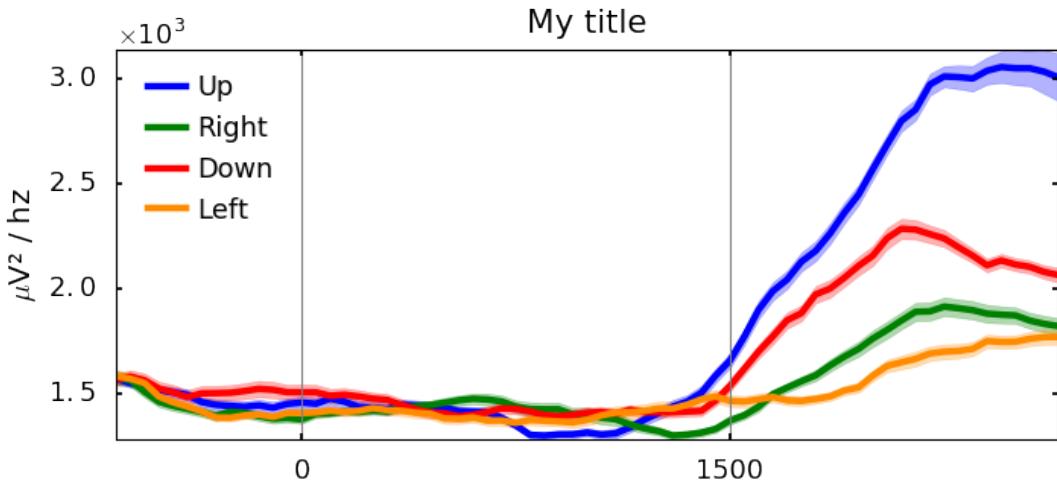


Fig. 4.9: Border plot example

**pval:** vector Vector of pvalues

**Kargs:**

**y:** int/float The y location of your p-values

**x:** vector x vector of the plot. Must have the same size as pval

**p:** float p-value threshold to plot

**minsucc:** int Minimum number of successive significant p-values

**color:** string Color of the p-value line

**shape:** string Shape of the p-value line

**lw:** int Linewidth of the p-value line

**kwargs:** Any supplemental arguments are passed to the plt.plot() function

**Return:** ax: updated matplotlib axes

### 4.19.3 Continuous color

```
class visual.continuouscol(ax, y, x=None, color=None, cmap='inferno', pltargs={}, **kwargs)
    Plot signal with continuous color
```

**Args:**

**ax:** matplotlib axes The axes to add lines. Use for example plt.gca()

**y:** vector Vector to plot

**Kargs:**

**x:** vector, optional, [def: None] Values on the x-axis. x should have the same length as y. By default, x-values are 0, 1, ..., len(y)

**color:** vector, optional, [def: None] Values to colorize the line. color should have the same length as y.

**cmap:** string, optional, [def: 'inferno'] The name of the colormap to use

**pltargs:** dict, optional, [def: {}] Arguments to pass to the LineCollection() function of matplotlib

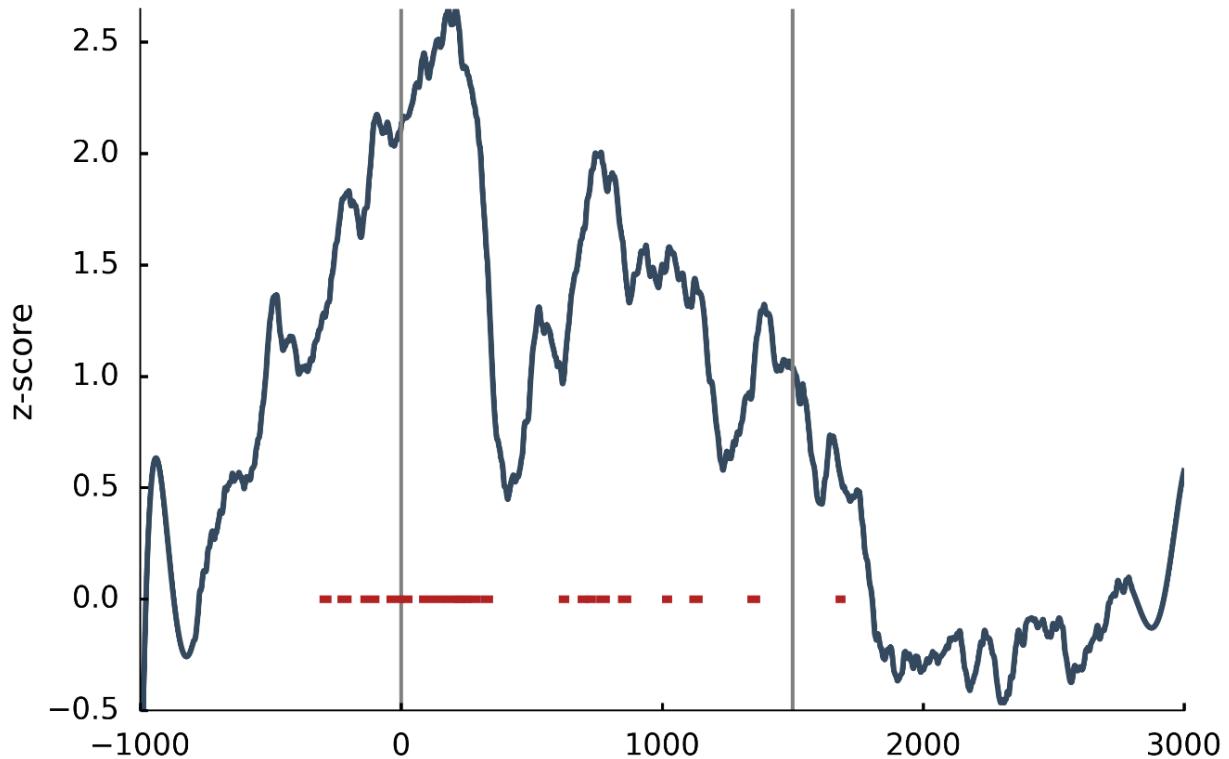


Fig. 4.10: Add p-values to an existing plot

**kwarg:** Supplementar arguments to control each subplot: title, xlabel, ylabel (which can be list for each subplot) xlim, ylim, xticks, yticks, xticklabels, yticklabels, style.

## 4.20 1-D or 2-D graphics

### 4.20.1 Add lines

```
class visual.addLines(ax, vLines=[], vColor=None, vShape=None, vWidth=None, hLines=[],
                      hColor=None, hWidth=None, hShape=None)
```

Add vertical and horizontal lines to an existing plot.

**Args:**

**ax: matplotlib axes** The axes to add lines. USE for example plt.gca()

**Kargs:**

**vLines: list, [def: []]** Define vertical lines. vLines should be a list of int/float

**vColor: list of strings, [def: ['gray']]** Control the color of the vertical lines. The length of the vColor list must be the same as the length of vLines

**vShape: list of strings, [def: ['-']]** Control the shape of the vertical lines. The length of the vShape list must be the same as the length of vLines

**hLines: list, [def: []]** Define horizontal lines. hLines should be a list of int/float

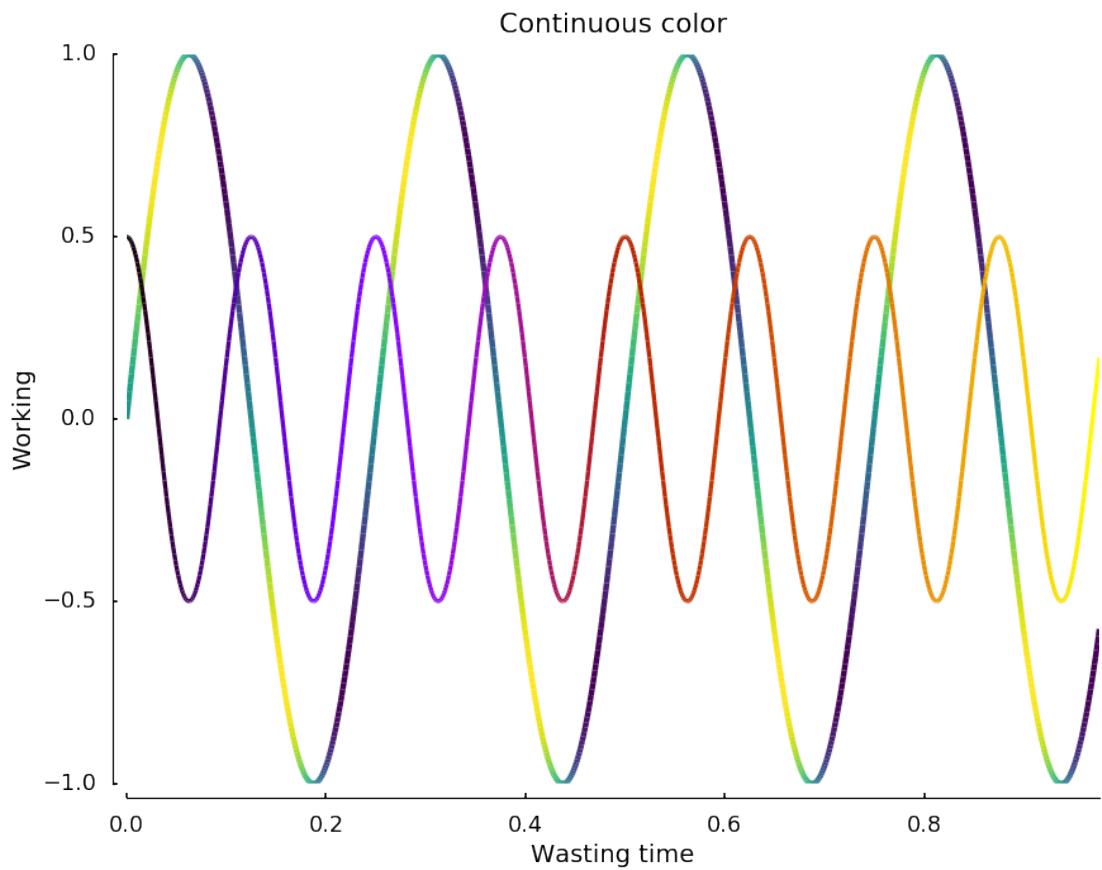


Fig. 4.11: Continuous color-line

**hColor: list of strings, [def: ['black']]** Control the color of the horizontal lines. The length of the hColor list must be the same as the length of hLines

**hShape: list of strings, [def: ['-']]** Control the shape of the horizontal lines. The length of the hShape list must be the same as the length of hLines

**Return:** The current axes

**Example:**

```
>>> # Create an empty plot:
>>> plt.plot([])
>>> plt.ylim([-1, 1]), plt.xlim([-10, 10])
>>> addLines(plt.gca(), vLines=[0, -5, 5, -7, 7], vColor=['k', 'r', 'g', 'y', 'b'],
>>>           vWidth=[5, 4, 3, 2, 1], vShape=['-', '--', '--', '--', '--'],
>>>           hLines=[0, -0.5, 0.7], hColor=['k', 'r', 'g'], hWidth=[5, 4, 3],
>>>           hShape=['-', '--', '--'])
```

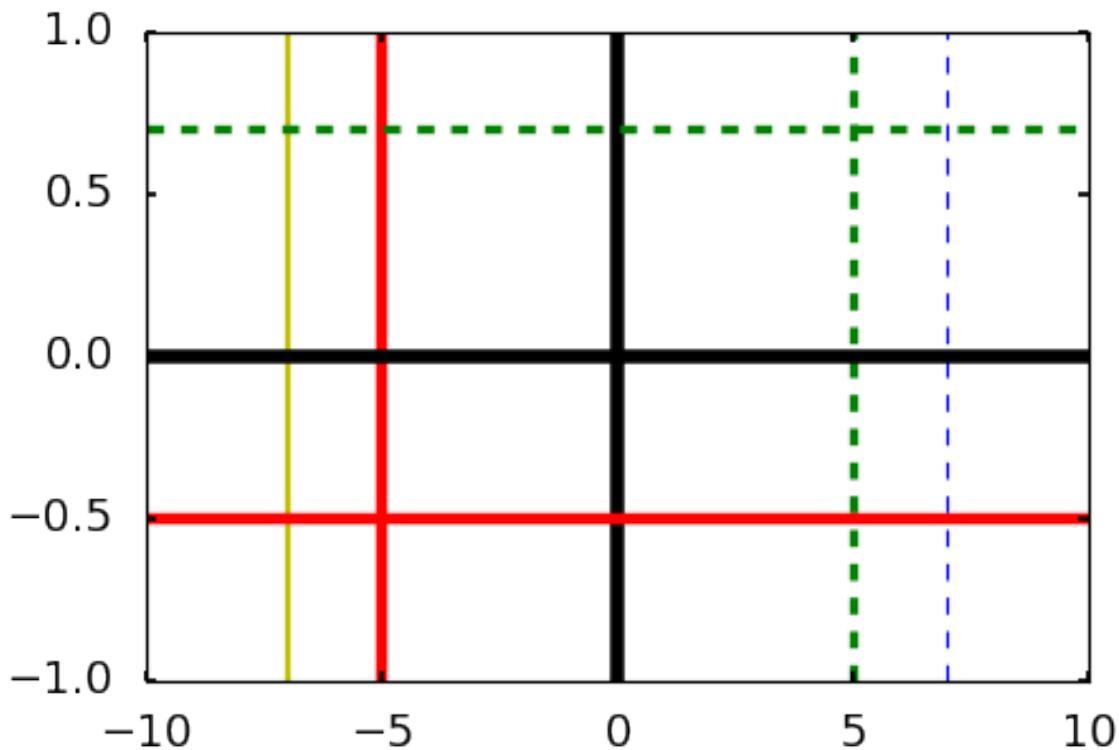


Fig. 4.12: Quickly add some lines to your plot

## 4.20.2 tilerplot

```
class visual.tilerplot
```

Automatic tiler plot for 1, 2 and 3D data.

```
plot1D (fig, y, x=None, maxplot=10, figtitle='', sharex=False, sharey=False, subdim=None, transpose=False, color='b', subspace=None, **kwargs)
```

Simple one dimentional plot

**Args:**

**y: array** Data to plot. y can either have one, two or three dimensions. If y is a vector, it will be plot in a simple window. If y is a matrix, all values inside are going to be superimpose. If y is a 3D matrix, the first dimension control the number of subplots.

**x: array, optional, [def: None]** x vector for plotting data.

**Kargs:**

**figtitle: string, optional, [def: ‘’]** Add a name to your figure

**subdim: tuple, optional, [def: None]** Force subplots to be subdim=(n\_columns, n\_rows)

**maxplot: int, optional, [def: 10]** Control the maximum number of subplot to prevent very large plot. By default, maxplot is 10 which mean that only 10 subplot can be defined.

**transpose: bool, optional, [def: False]** Invert subplot (row <-> column)

**color: string, optional, [def: ‘b’]** Color of the plot

**subspace: dict, optional, [def: None]** Control the distance in subplots. Use ‘left’, ‘bottom’, ‘right’, ‘top’, ‘wspace’, ‘hspace’. Example: {‘top’:0.85, ‘wspace’:0.8}

**kwargs:** Supplementar arguments to control each suplot: title, xlabel, ylabel (which can be list for each subplot) xlim, ylim, xticks, yticks, xticklabels, yticklabels, style, dpax, rmax.

**plot2D**(*fig*, *y*, *xvec=None*, *yvec=None*, *cmap=’inferno’*, *colorbar=True*, *cbticks=’minmax’*, *ycb=-10*, *cblabel=’’*, *under=None*, *over=None*, *vmin=None*, *vmax=None*, *sharex=False*, *sharey=False*, *textin=False*, *textcolor=’w’*, *texttype=’%.4f’*, *subdim=None*, *mask=None*, *interpolation=’none’*, *resample=(0, 0)*, *figtitle=’’*, *transpose=False*, *maxplot=10*, *subspace=None*, *contour=None*, *pltargs={}*, *pltype=’pcolor’*, *ncontour=10*, *polar=False*, *\*\*kwargs*)

Plot y as an image

**Args:**

**fig: figure** A matplotlib figure where plotting

**y: array** Data to plot. y can either have one, two or three dimensions. If y is a vector, it will be plot in a simple window. If y is a matrix, all values inside are going to be superimpose. If y is a 3D matrix, the first dimension control the number of subplots.

**Kargs:**

**xvec, yvec: array, optional, [def: None]** Vectors for y and x axis of each picture

**cmap: string, optional, [def: ‘inferno’]** Choice of the colormap

**colorbar: bool/string, optional, [def: True]** Add or not a colorbar to your plot. Alternatively, use ‘center-max’ or ‘center-dev’ to have a centered colorbar

**cbticks: list/string, optional, [def: ‘minmax’]** Control colorbar ticks. Use ‘auto’ for [min,(min+max)/2,max], ‘minmax’ for [min, max] or your own list.

**ycb: int, optional, [def: -10]** Distance between the colorbar and the label.

**cblabel: string, optional, [def: ‘’]** Label for the colorbar

**under, over: string, optional, [def: ‘’]** Color for everything under and over the colorbar limit.

**vmin, vmax: int/float, optional, [def: None]** Control minimum and maximum of the image

**sharex, sharey: bool, optional, [def: False]** Define if subplots should share x and y

**textin: bool, optional, [def: False]** Display values inside the heatmap

**textcolor: string, optional, [def: ‘w’]** Color of values inside the heatmap

**texttype: string, optional, [def: ‘%.4f’]** Way of display text inside the heatmap

**subdim: tuple, optional, [def: None]** Force subplots to be subdim=(n\_columns, n\_rows)

**interpolation: string, optional, [def: ‘none’]** Plot interpolation

**resample: tuple, optional, [def: (0, 0)]** Interpolate the map for a specific dimension. If (0.5, 0.1), this mean that the programme will insert one new point on x-axis, and 10 new points on y-axis. Pimp you map and make it sooo smooth.

**figtitle: string, optional, [def: ‘’]** Add a name to your figure

**maxplot: int, optional, [def: 10]** Control the maximum number of subplot to prevent very large plot. By default, maxplot is 10 which mean that only 10 subplot can be defined.

**transpose: bool, optional, [def: False]** Invert subplot (row <-> column)

**subspace: dict, optional, [def: None]** Control the distance in subplots. Use ‘left’, ‘bottom’, ‘right’, ‘top’, ‘wspace’, ‘hspace’. Example: {‘top’:0.85, ‘wspace’:0.8}

**contour: dict, optional, [def: None]** Add a contour to your 2D-plot. In order to use this parameter, define contour={‘data’:yourdata, ‘label’:[yourlabel], kwargs} where yourdata must have the same shape as y, level must float/int from smallest to largest. Use kwargs to pass other arguments to the contour function

**kwargs:** Supplementar arguments to control each subplot: title, xlabel, ylabel (which can be list for each subplot) xlim, ylim, xticks, yticks, xticklabels, yticklabels, style dpax, rmax.

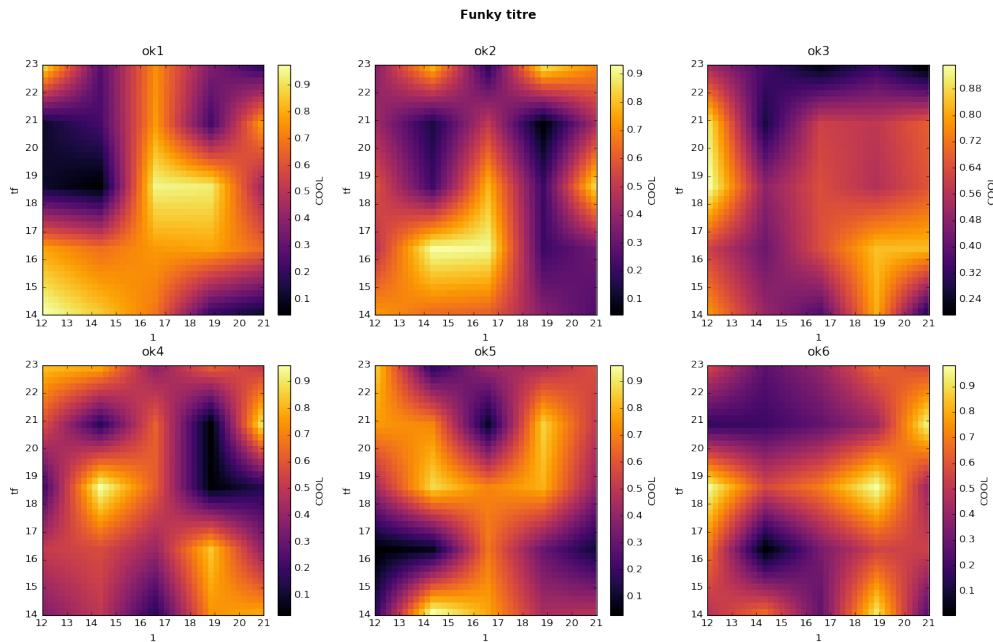


Fig. 4.13: Automatic 1D and 2D plot

## 4.21 Tools

`visual.rmaxis(ax, rmax)`

Remove ticks and axis of a existing plot

Args:

**ax: matplotlib axes** Axes to remove axis

**rmax: list of strings** List of axis name to be removed. For example, use ['left', 'right', 'top', 'bottom']

`visual.despine(ax, dpax, outward=10)`

Despine axis of a existing plot

Args:

**ax: matplotlib axes** Axes to despine axis

**dpax: list of strings** List of axis name to be despined. For example, use ['left', 'right', 'top', 'bottom']

Kargs:

**outward: int/float, optional, [def: 10]** Distance of despined axis from the original position.

## 4.22 Tools

### 4.22.1 Bpstudy

`class system.study(name=None)`

Create and manage a study with a files database.

Args:

**name: string, optional [def [None]]** Name of the study. If this study already exists, this will load the path with the associated database.

Example:

```
>>> # Define variables:  
>>> path = '/home/Documents/database'  
>>> studyName = 'MyStudy'
```

```
>>> # Create a study object :  
>>> studObj = study(name=studyName)  
>>> studObj.add(path)    # Create the study  
>>> studObj.studies()   # Print the list of studies
```

```
>>> # Manage files in your study :  
>>> fileList = studObj.search('filter1', 'filter2', folder='features')  
>>> # Let say that fileList contain two files : ['file1.mat', 'file2.pickle']  
>>> # Load the second file :  
>>> data = studObj.load('features', 'file2.pickle')
```

**add(path)**

Add a new study

Args:

**path: string** path to your study.

**The following folders are going to be created :**

- name: the root folder of the study. Same name as the study
- /database: datasets of the study
- /feature: features extracted from the diffrents datasets
- /classified: classified features
- /multifeature: multifeatures files
- /figure: figures of the study
- /physiology: physiological informations
- /backup: backup files
- /setting: study settings
- /other: any other kind of files

**delete()**

Delete the current study

**load(folder, name)**

Load a file. The file can be a .pickle or .mat

**Args:**

**folder: string** Specify where the file is located

**file: string** the complete name of the file

**Return:** A dictionary containing all the variables.

**search(\*args, \*, folder=' ', lower=True)**

Get a list of files

**Args:**

**args: string, optional** Add some filters to get a restricted list of files, according to the defined filters

**folder: string, optional [def: '']** Define a folder to search. By default, no folder is specified so the search will focused on the root folder.

**lower: bool, optional [def: True]** Define if the search method have to take care of the case. Use False if case is important for searching.

**Return:** A list containing the files found in the folder.

**static studies()**

Get the list of all defined studies

**static update()**

Update the list of studies

## 4.22.2 Pandas complements

```
class system.pdTools
    Tools for pandas DataFrame
```

**Syntax:**

- To search if arg1 is in column1:

```
>>> keep = ('column1', arg1)
```

- AND condition for arg1 and arg2 in column1:

```
>>> keep = ('column1', [arg1, 2])
```

- AND condition for arg1 in column1 and arg2 in column2:

```
>>> keep = [('column1', arg1), ('column2', arg2)]
```

- OR condition:

```
>>> keep = ('column1', arg1), ('column2', arg2), ...
```

**keep**(*df*, \**keep*, \*, *keep\_idx=True*)

Filter a pandas dataframe and keep only interresting rows.

**Args:**

**df: pandas dataframe** The dataframe to filter

**keep: tuple/list** Control the informations to keep. See the syntax definition

**keep\_idx: bool, optional [def [True]]** Add a column to df to check what are the rows that has been kept.

**Return:** A pandas Dataframe with only the informations to keep.

**remove**(*df*, \**rm*, \*, *rm\_idx=True*)

Filter a pandas dataframe and remove only interresting rows.

**Args:**

**df: pandas dataframe** The dataframe to be filter

**rm: tuple/list** Control the informations to remove. See the syntax definition

**rm\_idx: bool, optional [def [True]]** Add a column to df to check what are the rows that has been removed.

**Return:** A pandas Dataframe without the removed informations.

**search**(*df*, \**keep*)

Search in a pandas dataframe.

**Args:**

**df: pandas dataframe** The dataframe to filter

**keep: tuple/list** Control the informations to search. See the syntax definition

**Return:** List of row index for informations found.

### 4.22.3 Arrays

**tools.ndsplit**(*x*, *sp*, *axis=0*)

Split array (work for odd dimensions)

**Args:**

**x: array** Data to split

**sp: int** Number of chunk

**Kargs:**

**axis: int, optional, [def: 0]** Axis for splitting array

**Return:** List of splitted arrays

`tools.ndjoin(x, axis=0)`

Join arrays in a list

**Args:**

**x: list** List of data.

**Kargs:**

**axis: optional, [def: 0]** Axis to join arrays

**Return:** Array

#### 4.22.4 File management

##### Save file

`system.savefig(name, *arg, **kwargs)`

Save a file without carrying of extension.

arg: for .npy extension kwargs: for .pickle or .mat extensions

##### Load file

`system.loadfile(name)`

Load a file without carrying of extension. The function return a dictionary data.

#### 4.22.5 Others

`tools.p2str(p)`

Convert a pvalue to a string. Usefull for saving !

`tools.uorderlst(lst)`

Return a unique set of a list, and preserve order of appearance

### 4.23 References

This is a non exhaustive list of function and description of what is actually implemented in brainpipe. This list correspond to the most usefull functions.

#### 4.23.1 Pre-processing

Bundle of functions to pre-process data.

```
from brainpipe.preprocessing import *
```

Function	Description
bipolarization	Bipolarise stereotactic eeg electrodes
xyz2phy	Get physiological informations about structures using MNI or Talairach coordinates

### 4.23.2 Features

Bundle of functions to extract features from neural signals.

```
from brainpipe.features import *
```

Function	Description
sigfilt	Filtered signal only
amplitude	Amplitude of the signal
power	Power of the signal
phase	Phase of the signal
PLF	Phase-Locking Factor
TF	Time-frequency maps
pac	Phase-Amplitude Coupling (large variety of methods)
PhaseLockedPower	Time-frequency maps phase locked to a specific phase
erpac	Event Related Phase-Amplitude Coupling (time-resolved pac)
pfphase	Preferred-phase
PLV	Phase-Locking Value
PSD	Power Spectrum Density
powerPSD	Power exacted from PSD
SpectralEntropy	Spectral entropy (entropy extracted from PSD)

brainpipe also provide a bundle of tools for features

Function	Description
bandRef	Get usual oscillation bands informations
findBandName	Get physiological name of a frequency band
findBandFcy	Get frequency band from a physiological name
cfcVec	Generate cross-frequency vectors
cfcRndSignals	Generate signals artificially coupled (great to test pac methods)

### 4.23.3 Classification

Bundle of functions to classify extracted features.

```
from brainpipe.classification import *
```

Function	Description
defClf	Define a classifier
defCv	Define a cross-validation
classify	Classify features (either each one of them or grouping)
generalization	Generalization of decoding performance (generally, across time)
mf	Multi-features procedure. Select the best combination of features
MFpipe	Multi-features pipeline

### 4.23.4 Statistics

Bundle of functions to apply statistics.

```
from brainpipe.statistics import *
```

Function	Description
bino_da2p	Get associated p-value of a decoding accuracy using a binomial law
bino_p2da	Get associated decoding accuracy of a p-value using a binomial law
bino_signifeat	Get significant features using a binomial law
perm_2pvalue	Get p-value from a permutation dataset
perm_metric	Get a metric (usefull for mastat)
perm_pvalue2level	Get p-associated level in a permutation distribution
perm_rndDatasets	Generate random dataset of permutations
perm_swap	Randomly swap ndarray (matricial implementation)
perm_rep	Repeat a ndarray of permutations (matricial implementation)
bonferroni	Multiple comparison: Bonferroni
fdr	Multiple comparison: False Discovery Rate
maxstat	Multiple comparison: Maximum statistic
circ_corrc	Correlation coefficient between one circular and one linear random variable
circ_r	Computes mean resultant vector length for circular data
circ_rtest	Computes Rayleigh test for non-uniformity of circular data

## 4.23.5 Visualization

Bundle of functions to visualize results and make some <3 pretty plots <3.

```
from brainpipe.visual import *
```

Function	Description
BorderPlot	Plot data and deviation/sem in transparency
addPval	Add p-values to an existing plot
continuouscol	Plot lines with continuous color
addLines	Quickly add vertical and horizontal lines
tilerplot	Generate automatic 1D or 2D subplots with a lot of control
addPval	Add significants p-value to an existing plot
rmaxis	Remove ticks and axis of a existing plot
despine	Despine axis of a existing plot

## 4.23.6 Tools

This part provide a set complement

```
from brainpipe.tools import *
```

Function	Description
study	Manage your current study without carrying of path
savefile	Quickly save files using most common extensions
loadfile	Quickly load files using most common extensions
pdTools	Some complement functions for pandas Dataframe (search, keep, remove)
ndsplit	Split ndarray (works on odd and even dimensions)
ndjoin	Join ndarray (works on odd and even dimensions)
p2str	Transform a p-value to string (usefull to save files with corresponding p-value)
uorderlst	Get unique ordered elements from a list

Indices and tables

- genindex

- modindex
- search

# BIBLIOGRAPHIE

- Acharya, S., Fifer, M. S., Benz, H. L., Crone, N. E., and Thakor, N. V. (2010). Electrocorticographic amplitude predicts finger positions during slow grasping motions of the hand. *Journal of neural engineering*, 7(4) :046002.
- Allison, B. Z., McFarland, D. J., Schalk, G., Zheng, S. D., Jackson, M. M., and Wolpaw, J. R. (2008). Towards an independent brain-computer interface using steady state visual evoked potentials. *Clinical neurophysiology*, 119(2) :399–408.
- Amzica, F. and Steriade, M. (1998). Electrophysiological correlates of sleep delta waves. *Electroencephalography and Clinical Neurophysiology*, 107(2) :69–83.
- Aru, J., Aru, J., Priesemann, V., Wibral, M., Lana, L., Pipa, G., Singer, W., and Vicente, R. (2015). Untangling cross-frequency coupling in neuroscience. *Current Opinion in Neurobiology*, 31 :51–61.
- Bahramisharif, A., van Gerven, M. A. J., Aarnoutse, E. J., Mercier, M. R., Schwartz, T. H., Foxe, J. J., Ramsey, N. F., and Jensen, O. (2013). Propagating Neocortical Gamma Bursts Are Coordinated by Traveling Alpha Waves. *Journal of Neuroscience*, 33(48) :18849–18854.
- Ball, T., Schreiber, A., Feige, B., Wagner, M., Lücking, C. H., and Kristeva-Feige, R. (1999). The role of higher-order motor areas in voluntary movement as revealed by high-resolution EEG and fMRI. *Neuroimage*, 10(6) :682–694.
- Ball, T., Schulze-Bonhage, A., Aertsen, A., and Mehring, C. (2009). Differential representation of arm movement direction in relation to cortical anatomy and function. *Journal of Neural Engineering*, 6(1) :016006.
- Bastin, J., Deman, P., David, O., Gueguen, M., Benis, D., Minotti, L., Hoffman, D., Combrisson, E., Kujala, J., Perrone-Bertolotti, M., Kahane, P., Lachaux, J.-P., and Jerbi, K. (2016). Direct Recordings from Human Anterior Insula Reveal its Leading Role within the Error-Monitoring Network. *Cerebral Cortex*, page bhv352.
- Bekaert, M. H., Botte-Lecocq, C., Cabestaing, F., Rakotomamonjy, A., and others (2009). Les interfaces Cerveau-Machine pour la palliation du handicap moteur sévère. *Sciences et Technologies pour le Handicap*, 3(1) :95–121.
- Berens, P. and others (2009). CircStat a MATLAB toolbox for circular statistics. *J Stat Softw*, 31(10) :1–21.
- Berger, H. (1929). Ueber das elektroenkephalogramm des menschen. *Archiv für Psychiatrie und Nervenkrankheiten*, 87(1) :527–570.
- Bertrand, O., Bohorquez, J., and Pernier, J. (1994). Time-frequency digital filtering based on an invertible wavelet transform : An application to evoked potentials. *IEEE Transactions on Biomedical Engineering*, 41(1) :77–88.

- Besserve, M. (2007). *Analyse de la dynamique neuronale pour les Interfaces Cerveau-Machines : un retour aux sources*. PhD thesis, Université Paris Sud - Paris XI.
- Besserve, M., Jerbi, K., Laurent, F., Baillet, S., Martinerie, J., Garnero, L., and others (2007). Classification methods for ongoing EEG and MEG signals. *Biol Res*, 40(4) :415–437.
- Beverina, F., Palmas, G., Silvoni, S., Piccione, F., and Giove, S. (2003). User adaptive BCIs : SSVEP and P300 based interfaces. *PsychNology Journal*, 1(4) :331–354.
- Birbaumer, N. (1997). Slow cortical potentials : Their origin, meaning, and clinical use. *Brain and behavior past, present, and future*, pages 25–39.
- Birbaumer, N. (1999). Slow cortical potentials : Plasticity, operant control, and behavioral effects. *The Neuroscientist*, 5(2) :74–78.
- Birbaumer, N. (2006). Breaking the silence : Brain computer interfaces (BCI) for communication and motor control. *Psychophysiology*, 43(6) :517–532.
- Birbaumer, N., Elbert, T., Canavan, A. G., and Rockstroh, B. (1990). Slow potentials of the cerebral cortex and behavior. *Physiological reviews*, 70(1) :1–41.
- Birbaumer, N., Ghanayim, N., Hinterberger, T., Iversen, I., Kotchoubey, B., Kübler, A., Perelmouter, J., Taub, E., and Flor, H. (1999). A spelling device for the paralysed. *Nature*, 398(6725) :297–298.
- Birbaumer, N., Hinterberger, T., Kubler, A., and Neumann, N. (2003). The thought-translation device (ttd) : Neurobehavioral mechanisms and clinical outcome. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 11(2) :120–123.
- Birbaumer, N., Kubler, A., Ghanayim, N., Hinterberger, T., Perelmouter, J., Kaiser, J., Iversen, I., Kotchoubey, B., Neumann, N., and Flor, H. (2000). The thought translation device (TTD) for completely paralyzed patients. *IEEE Transactions on Rehabilitation Engineering*, 8(2) :190–193.
- Blankertz, B., Krauledat, M., Dornhege, G., Williamson, J., Murray-Smith, R., and Müller, K.-R. (2007). A note on brain actuated spelling with the Berlin brain-computer interface. In *International Conference on Universal Access in Human-Computer Interaction*, pages 759–768. Springer.
- Blankertz, B., Muller, K., Krusienski, D., Schalk, G., Wolpaw, J., Schlogl, A., Pfurtscheller, G., Millan, J., Schroder, M., and Birbaumer, N. (2006). The BCI Competition III : Validating Alternative Approaches to Actual BCI Problems. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 14(2) :153–159.
- Blankertz, B., Muller, K. R., Curio, G., Vaughan, T. M., Schalk, G., Wolpaw, J. R., Schlogl, A., Neuper, C., Pfurtscheller, G., Hinterberger, T., and others (2004). The BCI competition 2003 : Progress and perspectives in detection and discrimination of EEG single trials. *Biomedical Engineering, IEEE Transactions on*, 51(6) :1044–1051.
- Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152. ACM.
- Bradley, A. P. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7) :1145–1159.

- Brouwer, A.-M. and Van Erp, J. B. (2010). A tactile P300 brain-computer interface. *Frontiers in neuroscience*, 4 :19.
- Brunia, C. and Van Boxtel, G. (2000). Motor preparation.
- Brunner, P., Ritaccio, A. L., Emrich, J. F., Bischof, H., and Schalk, G. (2011). Rapid communication with a “P300” matrix speller using electrocorticographic signals (ECOG). *Frontiers in neuroscience*, 5 :5.
- Bruns, A. and Eckhorn, R. (2004). Task-related coupling from high-to low-frequency signals among visual cortical areas in human subdural recordings. *International Journal of Psychophysiology*, 51(2) :97–116.
- Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2) :121–167.
- Campagnola, L., Klein, A., Rossant, C., and Rougier, N. (2013). VISPY, A Modern and Interactive Scientific Visualisation. In *Euroscipy 2013*.
- Canolty, R. T., Edwards, E., Dalal, S. S., Soltani, M., Nagarajan, S. S., Kirsch, H. E., Berger, M. S., Barbaro, N. M., and Knight, R. T. (2006). High Gamma Power Is Phase-Locked to Theta Oscillations in Human Neocortex. *Science*, 313(5793) :1626–1628.
- Canolty, R. T. and Knight, R. T. (2010). The functional role of cross-frequency coupling.
- Cassim, F., Monaca, C., Szurhaj, W., Bourriez, J.-L., Defebvre, L., Derambure, P., and Guieu, J.-D. (2001). Does post-movement beta synchronization reflect an idling motor cortex? *Neuroreport*, 12(17) :3859–3863.
- Cecotti, H. (2010). A self-paced and calibration-less SSVEP-based brain–computer interface speller. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 18(2) :127–133.
- Chai, H. and Domeniconi, C. (2004). An evaluation of gene selection methods for multi-class microarray data classification. In *Proceedings of the Second European Workshop on Data Mining and Text Mining in Bioinformatics*, pages 3–10.
- Chang, C.-C. and Lin, C.-J. (2011). LIBSVM a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3) :27.
- Chapin, J. K., Moxon, K. A., Markowitz, R. S., and Nicolelis, M. A. (1999). Real-time control of a robot arm using simultaneously recorded neurons in the motor cortex. *Nature neuroscience*, 2(7) :664–670.
- Chaudhary, U., Birbaumer, N., and Curado, M. (2015). Brain-Machine Interface (BMI) in paralysis. *Annals of Physical and Rehabilitation Medicine*, 58(1) :9–13.
- Cohen, M. X. (2008). Assessing transient cross-frequency coupling in EEG data. *Journal of Neuroscience Methods*, 168(2) :494–499.
- Cohen, M. X., Elger, C. E., and Fell, J. (2008). Oscillatory activity and phase–amplitude coupling in the human medial frontal cortex during decision making. *Journal of cognitive neuroscience*, 21(2) :390–402.

- Collinger, J. L., Wodlinger, B., Downey, J. E., Wang, W., Tyler-Kabara, E. C., Weber, D. J., McMorland, A. J., Velliste, M., Boninger, M. L., and Schwartz, A. B. (2013). High performance neuroprosthetic control by an individual with tetraplegia. *The Lancet*, 381(9866) :557–564.
- Combrisson, E. and Jerbi, K. (2015). Exceeding chance level by chance : The caveat of theoretical chance levels in brain signal classification and statistical assessment of decoding accuracy. *Journal of Neuroscience Methods*, 250 :126–136.
- Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3) :273–297.
- Coyle, D., Garcia, J., Satti, A. R., and McGinnity, T. M. (2011). EEG-based continuous control of a game using a 3 channel motor imagery BCI : BCI game. In *Computational Intelligence, Cognitive Algorithms, Mind, and Brain (CCMB), 2011 IEEE Symposium On*, pages 1–7. IEEE.
- Coyle, S., Ward, T., Markham, C., and McDarby, G. (2004). On the suitability of near-infrared (NIR) systems for next-generation brain–computer interfaces. *Physiological Measurement*, 25(4) :815–822.
- Crone, N. (1998). Functional mapping of human sensorimotor cortex with electrocorticographic spectral analysis. I. Alpha and beta event- related desynchronization. *Brain*, 121(12) :2271–2299.
- Crone, N. E., Miglioretti, D. L., Gordon, B., and Lesser, R. P. (1998). Functional mapping of human sensorimotor cortex with electrocorticographic spectral analysis. II. Event-related synchronization in the gamma band. *Brain*, 121(12) :2301–2315.
- Curran, E. (2003). Learning to control brain activity : A review of the production and control of EEG components for driving brain–computer interface (BCI) systems. *Brain and Cognition*, 51(3) :326–336.
- Das, S. (2001). Filters, wrappers and a boosting-based hybrid for feature selection. In *ICML*, volume 1, pages 74–81. Citeseer.
- de Hemptinne, C., Ryapolova-Webb, E. S., Air, E. L., Garcia, P. A., Miller, K. J., Ojemann, J. G., Ostrem, J. L., Galifianakis, N. B., and Starr, P. A. (2013). Exaggerated phase–amplitude coupling in the primary motor cortex in Parkinson disease. *Proceedings of the National Academy of Sciences*.
- de Vries, S., Tepper, M., Otten, B., and Mulder, T. (2011). Recovery of Motor Imagery Ability in Stroke Patients. *Rehabilitation Research and Practice*, 2011 :1–9.
- Deiber, M.-P., Honda, M., Ibañez, V., Sadato, N., and Hallett, M. (1999). Mesial motor areas in self-initiated versus externally triggered movements examined with fMRI : Effect of movement type and rate. *Journal of neurophysiology*, 81(6) :3065–3077.
- Demandt, E., Mehring, C., Vogt, K., Schulze-Bonhage, A., Aertsen, A., and Ball, T. (2012). Reaching Movement Onset- and End-Related Characteristics of EEG Spectral Power Modulations. *Frontiers in Neuroscience*, 6.
- Di Rienzo, F., Collet, C., Hoyek, N., and Guillot, A. (2014). Impact of Neurologic Deficits on Motor Imagery : A Systematic Review of Clinical Evaluations. *Neuropsychology Review*, 24(2) :116–147.

- Di Rienzo, F., Debarnot, U., Daligault, S., Saruco, E., Delpuech, C., Doyon, J., Collet, C., and Guillot, A. (2016). Online and Offline Performance Gains Following Motor Imagery Practice : A Comprehensive Review of Behavioral and Neuroimaging Studies. *Frontiers in Human Neuroscience*, 10.
- Diez, P. F., Torres Müller, S. M., Mut, V. A., Laciár, E., Avila, E., Bastos-Filho, T. F., and Sarcinelli-Filho, M. (2013). Commanding a robotic wheelchair with a high-frequency steady-state visual evoked potential based brain-computer interface. *Medical Engineering & Physics*, 35(8) :1155–1164.
- Ding, C. and Peng, H. (2005). Minimum Redundancy Feature Selection from Microarray Gene Expression Data.
- Donchin, E., Spencer, K., and Wijesinghe, R. (2000). The mental prosthesis : Assessing the speed of a P300-based brain-computer interface. *IEEE Transactions on Rehabilitation Engineering*, 8(2) :174–179.
- Donoghue, J. P. (2002). Connecting cortex to machines : Recent advances in brain interfaces. *Nature Neuroscience*, 5(Supp) :1085–1088.
- Doud, A. J., Lucas, J. P., Pisansky, M. T., and He, B. (2011). Continuous three-dimensional control of a virtual helicopter using a motor imagery based brain-computer interface. *PloS one*, 6(10) :e26322.
- Driskell, J. E., Copper, C., and Moran, A. (1994). Does mental practice enhance performance? *Journal of applied psychology*, 79(4) :481.
- Duda, R. O., Hart, P. E., and Stork, D. G. (2001). Pattern classification. 2nd. Edition. New York.
- Dvorak, D. and Fenton, A. A. (2014). Toward a proper estimation of phase-amplitude coupling in neural oscillations. *Journal of Neuroscience Methods*, 225 :42–56.
- Efron, B. and Tibshirani, R. J. (1994). *An Introduction to the Bootstrap*. CRC press.
- Engel, A. K., Moll, C. K. E., Fried, I., and Ojemann, G. A. (2005). Invasive recordings from the human brain : Clinical insights and beyond. *Nature Reviews Neuroscience*, 6(1) :35–47.
- Farwell, L. and Donchin, E. (1988). Talking off the top of your head : Toward a mental prosthesis utilizing event-related brain potentials. *Electroencephalography and Clinical Neurophysiology*, 70(6) :510–523.
- Fazli, S., Mehnert, J., Steinbrink, J., Curio, G., Villringer, A., Müller, K.-R., and Blankertz, B. (2012). Enhanced performance by a hybrid NIRS-EEG brain computer interface. *Neuroimage*, 59(1) :519–529.
- Felton, E. A., Wilson, J. A., Williams, J. C., and Garell, P. C. (2007). Electrocorticographically controlled brain-computer interfaces using motor and sensory imagery in patients with temporary subdural electrode implants : Report of four cases. *Journal of neurosurgery*, 106(3) :495–500.
- Fetz, E. E. (1969). Operant conditioning of cortical unit activity. *Science (New York, N.Y.)*, 163(3870) :955–958.

- Fetz, E. E. and others (1999). Real-time control of a robotic arm by neuronal ensembles. *nature neuroscience*, 2 :583–584.
- Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2) :179–188.
- Fix, E. and Hodges Jr, J. L. (1951). Discriminatory analysis-nonparametric discrimination : Consistency properties. Technical report, DTIC Document.
- Fries, P. (2005). A mechanism for cognitive dynamics : Neuronal communication through neuronal coherence. *Trends in Cognitive Sciences*, 9(10) :474–480.
- Friston, K. J. (1997). Another neural code ? *Neuroimage*, 5(3) :213–220.
- Fukunaga, K. (1990). Introduction to statistical pattern classification.
- Furdea, A., Halder, S., Krusienski, D., Bross, D., Nijboer, F., Birbaumer, N., and Kübler, A. (2009). An auditory oddball (P300) spelling system for brain-computer interfaces. *Psychophysiology*, 46(3) :617–625.
- Galán, F., Nuttin, M., Lew, E., Ferrez, P., Vanacker, G., Philips, J., and Millán, J. d. R. (2008a). A brain-actuated wheelchair : Asynchronous and non-invasive Brain–computer interfaces for continuous control of robots. *Clinical Neurophysiology*, 119(9) :2159–2169.
- Galán, F., Nuttin, M., Lew, E., Ferrez, P. W., Vanacker, G., Philips, J., and Millán, J. d. R. (2008b). A brain-actuated wheelchair : Asynchronous and non-invasive brain–computer interfaces for continuous control of robots. *Clinical Neurophysiology*, 119(9) :2159–2169.
- Garnero, L., Baillet, S., and Renault, B. (1998). Magnétoencéphalographie / électroencéphalographie et imagerie cérébrale fonctionnelle. *Annales de l'Institut Pasteur / Actualités*, 9(3) :215–226.
- Georgopoulos, A., Schwartz, A., and Kettner, R. (1986). Neuronal population coding of movement direction. *Science*, 233(4771) :1416–1419.
- Georgopoulos, A. P. and Carpenter, A. F. (2015). Coding of movements in the motor cortex. *Current Opinion in Neurobiology*, 33 :34–39.
- Georgopoulos, A. P., Kalaska, J. F., Caminiti, R., and Massey, J. T. (1982). On the relations between the direction of two-dimensional arm movements and cell discharge in primate motor cortex. *The Journal of Neuroscience*, 2(11) :1527–1537.
- Graimann, B., Allison, B., and Pfurtscheller, G. (2009). Brain–Computer Interfaces : A Gentle Introduction. In Graimann, B., Pfurtscheller, G., and Allison, B., editors, *Brain-Computer Interfaces*, pages 1–27. Springer Berlin Heidelberg, Berlin, Heidelberg.
- Gregoriou, G. G., Gotts, S. J., Zhou, H., and Desimone, R. (2009). High-frequency, long-range coupling between prefrontal and visual cortex during attention. *Science (New York, N.Y.)*, 324(5931) :1207–1210.
- Guillot, A., Collet, C., Nguyen, V. A., Malouin, F., Richards, C., and Doyon, J. (2008). Functional neuroanatomical networks associated with expertise in motor imagery. *NeuroImage*, 41(4) :1471–1483.

- Guillot, A., Collet, C., Nguyen, V. A., Malouin, F., Richards, C., and Doyon, J. (2009). Brain activity during visual versus kinesthetic imagery : An fMRI study. *Human Brain Mapping*, 30(7) :2157–2172.
- Gunduz, A., Brunner, P., Sharma, M., Leuthardt, E. C., Ritaccio, A. L., Pesaran, B., and Schalk, G. (2016). Differential roles of high gamma and local motor potentials for movement preparation and execution. *Brain-Computer Interfaces*, 3(2) :88–102.
- Gunduz, A., Sanchez, J. C., Carney, P. R., and Principe, J. C. (2009). Mapping broadband electrocorticographic recordings to two-dimensional hand trajectories in humans : Motor control features. *Neural Networks*, 22(9) :1257–1270.
- Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3 :1157–1182.
- Hamadicharef, B., Xu, M., and Aditya, S. (2010). Brain-Computer Interface (BCI) based musical composition. In *Cyberworlds (CW), 2010 International Conference On*, pages 282–286. IEEE.
- Hammer, J., Fischer, J., Ruescher, J., Schulze-Bonhage, A., Aertsen, A., and Ball, T. (2013). The role of ECoG magnitude and phase in decoding position, velocity, and acceleration during continuous motor behavior. *Frontiers in Neuroscience*, 7.
- Hammer, J., Pistohl, T., Fischer, J., Kršek, P., Tomášek, M., Marusič, P., Schulze-Bonhage, A., Aertsen, A., and Ball, T. (2016). Predominance of Movement Speed Over Direction in Neuronal Population Signals of Motor Cortex : Intracranial EEG Data and A Simple Explanatory Model. *Cerebral Cortex*, 26(6) :2863–2881.
- Hammon, P., Makeig, S., Poizner, H., Todorov, E., and De Sa, V. (2008). Predicting Reaching Targets from Human EEG. *IEEE Signal Processing Magazine*, 25(1) :69–77.
- Hanakawa, T., Dimyan, M. A., and Hallett, M. (2008). Motor Planning, Imagery, and Execution in the Distributed Motor Network : A Time-Course Study with Functional MRI. *Cerebral Cortex*, 18(12) :2775–2788.
- Hochberg, L. R., Bacher, D., Jarosiewicz, B., Masse, N. Y., Simeral, J. D., Vogel, J., Haddadin, S., Liu, J., Cash, S. S., van der Smagt, P., and Donoghue, J. P. (2012). Reach and grasp by people with tetraplegia using a neurally controlled robotic arm. *Nature*, 485(7398) :372–375.
- Hochberg, L. R., Serruya, M. D., Friehs, G. M., Mukand, J. A., Saleh, M., Caplan, A. H., Branner, A., Chen, D., Penn, R. D., and Donoghue, J. P. (2006). Neuronal ensemble control of prosthetic devices by a human with tetraplegia. *Nature*, 442(7099) :164–171.
- Hoffmann, U., Vesin, J.-M., Ebrahimi, T., and Diserens, K. (2008). An efficient P300-based brain-computer interface for disabled subjects. *Journal of Neuroscience methods*, 167(1) :115–125.
- Huang, J. and Ling, C. X. (2005). Using AUC and accuracy in evaluating learning algorithms. *Knowledge and Data Engineering, IEEE Transactions on*, 17(3) :299–310.
- Hyafil, A., Giraud, A.-L., Fontolan, L., and Gutkin, B. (2015). Neural Cross-Frequency Coupling : Connecting Architectures, Mechanisms, and Functions. *Trends in Neurosciences*, 38(11) :725–740.

- Jackson, P. L., Lafleur, M. F., Malouin, F., Richards, C., and Doyon, J. (2001). Potential role of mental practice using motor imagery in neurologic rehabilitation. *Archives of Physical Medicine and Rehabilitation*, 82(8) :1133–1141.
- Jackson, P. L., Meltzoff, A. N., and Decety, J. (2006). Neural circuits involved in imitation and perspective-taking. *NeuroImage*, 31(1) :429–439.
- Jankelowitz and Colebatch (2002). Movement-related potentials associated with self-paced, cued and imagined arm movements. *Experimental Brain Research*, 147(1) :98–107.
- Jenkins, I. H., Jahanshahi, M., Jueptner, M., Passingham, R. E., and Brooks, D. J. (2000). Self-initiated versus externally triggered movements. *Brain*, 123(6) :1216–1228.
- Jerbi, K., Freyermuth, S., Minotti, L., Kahane, P., Berthoz, A., and Lachaux, J.-P. (2009a). Chapter 12 Watching Brain TV and Playing Brain Ball. In *International Review of Neurobiology*, volume 86, pages 159–168. Elsevier.
- Jerbi, K., Ossandón, T., Hamamé, C. M., Senova, S., Dalal, S. S., Jung, J., Minotti, L., Bertrand, O., Berthoz, A., Kahane, P., and Lachaux, J.-P. (2009b). Task-related gamma-band dynamics from an intracerebral perspective : Review and implications for surface EEG and MEG. *Human Brain Mapping*, 30(6) :1758–1771.
- Jervis, B. W., Nichols, M. J., Johnson, T. E., Allen, E., and Hudson, N. R. (1983). A fundamental investigation of the composition of auditory evoked potentials. *IEEE transactions on bio-medical engineering*, 30(1) :43–50.
- Jiang, D., Edwards, M. G., Mullins, P., and Callow, N. (2015). The neural substrates for the different modalities of movement imagery. *Brain and Cognition*, 97 :22–31.
- Jobsis, F. F. (1977). Noninvasive, infrared monitoring of cerebral and myocardial oxygen sufficiency and circulatory parameters. *Science*, 198(4323) :1264–1267.
- Kahane, P., Landré, E., Minotti, L., Francione, S., and Ryvlin, P. (2006). The Bancaud and Talairach view on the epileptogenic zone : A working hypothesis. *Epileptic disorders : international epilepsy journal with videotape*, 8 :S16–26.
- Kaiser, V., Bauernfeind, G., Kreilinger, A., Kaufmann, T., Kübler, A., Neuper, C., and Müller-Putz, G. R. (2014). Cortical effects of user training in a motor imagery based brain–computer interface measured by fNIRS and EEG. *Neuroimage*, 85 :432–444.
- Kayagil, T. A., Bai, O., Henriquez, C. S., Lin, P., Furlani, S. J., Vorbach, S., and Hallett, M. (2009). A binary method for simple and accurate two-dimensional cursor control from EEG with minimal subject training. *Journal of neuroengineering and rehabilitation*, 6(1) :1.
- Kim, S.-P., Simeral, J. D., Hochberg, L. R., Donoghue, J. P., Friehs, G. M., and Black, M. J. (2011). Point-and-Click Cursor Control With an Intracortical Neural Interface System by Humans With Tetraplegia. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 19(2) :193–203.
- King, J.-R. and Dehaene, S. (2014). Characterizing the dynamics of mental representations : The temporal generalization method. *Trends in Cognitive Sciences*, 18(4) :203–210.

- Kleber, B. and Birbaumer, N. (2005). Direct brain communication : Neuroelectric and metabolic approaches at Tübingen. *Cognitive Processing*, 6(1) :65–74.
- Klimesch, W. (1999). EEG alpha and theta oscillations reflect cognitive and memory performance : A review and analysis. *Brain research reviews*, 29(2) :169–195.
- Klimesch, W., Sauseng, P., and Hanslmayr, S. (2007). EEG alpha oscillations : The inhibition-timing hypothesis. *Brain Research Reviews*, 53(1) :63–88.
- Kohavi, R. and others (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI*, volume 14, pages 1137–1145.
- Kornhuber, H. H. and Deecke, L. (1965). Hirnpotentialänderungen bei Willkürbewegungen und passiven Bewegungen des Menschen : Bereitschaftspotential und reafferente Potentiale. *Pflüger's Archiv für die gesamte Physiologie des Menschen und der Tiere*, 284(1) :1–17.
- Kosslyn, S. M., Seger, C., Pani, J. R., and Hillger, L. A. (1990). When is imagery used in everyday life ? A diary study. *Journal of Mental Imagery*.
- Kotchoubey, B., Strehl, U., Holzapfel, S., Schneider, D., Blankenhorn, V., and Birbaumer, N. (1998). Control of cortical excitability in epilepsy. *Advances in neurology*, 81 :281–290.
- Krusienski, D. J. and Shih, J. J. (2011a). Control of a brain-computer interface using stereotactic depth electrodes in and adjacent to the hippocampus. *Journal of Neural Engineering*, 8(2) :025006.
- Krusienski, D. J. and Shih, J. J. (2011b). Control of a visual keyboard using an electrocorticographic brain-computer interface. *Neurorehabilitation and neural repair*, 25(4) :323–331.
- Kübler, A., Kotchoubey, B., Hinterberger, T., Ghanayim, N., Perelmouter, J., Schauer, M., Fritsch, C., Taub, E., and Birbaumer, N. (1999). The thought translation device : A neurophysiological approach to communication in total motor paralysis. *Experimental brain research*, 124(2) :223–232.
- Kübler, A., Kotchoubey, B., Kaiser, J., Wolpaw, J. R., and Birbaumer, N. (2001). Brain computer communication : Unlocking the locked in. *Psychological bulletin*, 127(3) :358.
- Lachaux, J., Rudrauf, D., and Kahane, P. (2003). Intracranial EEG and human brain mapping. *Journal of Physiology-Paris*, 97(4-6) :613–628.
- Lachaux, J.-P., Jerbi, K., Bertrand, O., Minotti, L., Hoffmann, D., Schoendorff, B., and Kahane, P. (2007). BrainTV a novel approach for online mapping of human brain functions. *Biological research*, 40(4) :401–413.
- Lachaux, J.-P., Rodriguez, E., Le Van Quyen, M., Lutz, A., Martinerie, J., and Varela, F. J. (2000). Studying single trials of phase synchronous activity in the brain. *International Journal of Bifurcation and Chaos*, 10(10) :2429–2439.
- Lachaux, J.-P., Rodriguez, E., Martinerie, J., Varela, F. J., and others (1999). Measuring phase synchrony in brain signals. *Human brain mapping*, 8(4) :194–208.

- Lajnef, T., Chaibi, S., Ruby, P., Aguera, P.-E., Eichenlaub, J.-B., Samet, M., Kachouri, A., and Jerbi, K. (2015). Learning machines and sleeping brains : Automatic sleep stage classification using decision-tree multi-class support vector machines. *Journal of Neuroscience Methods*, 250 :94–105.
- Lakatos, P. (2005). An Oscillatory Hierarchy Controlling Neuronal Excitability and Stimulus Processing in the Auditory Cortex. *Journal of Neurophysiology*, 94(3) :1904–1911.
- Lalor, E., Kelly, S. P., Finucane, C., Burke, R., Reilly, R. B., and McDarby, G. (2004). Brain computer interface based on the steady-state VEP for immersive gaming control. *Biomed. Tech*, 49(1) :63–64.
- Lalor, E. C., Kelly, S. P., Finucane, C., Burke, R., Smith, R., Reilly, R. B., and McDarby, G. (2005). Steady-State VEP-Based Brain-Computer Interface Control in an Immersive 3D Gaming Environment. *EURASIP Journal on Advances in Signal Processing*, 2005(19) :3156–3164.
- Latinne, P., Debeir, O., and Decaestecker, C. (2001). Limiting the number of trees in random forests. In *International Workshop on Multiple Classifier Systems*, pages 178–187. Springer.
- Lebedev, M. A. and Nicolelis, M. A. (2006). Brain-machine interfaces : Past, present and future. *Trends in Neurosciences*, 29(9) :536–546.
- Lee, J. and Jeong, J. (2013). Correlation of risk-taking propensity with cross-frequency phase-amplitude coupling in the resting EEG. *Clinical Neurophysiology*.
- Leeb, R., Friedman, D., Müller-Putz, G. R., Scherer, R., Slater, M., and Pfurtscheller, G. (2007). Self-Paced (Asynchronous) BCI Control of a Wheelchair in Virtual Environments : A Case Study with a Tetraplegic. *Computational Intelligence and Neuroscience*, 2007 :1–8.
- Lega, B., Burke, J., Jacobs, J., and Kahana, M. J. (2016). Slow-Theta-to-Gamma Phase-Amplitude Coupling in Human Hippocampus Supports the Formation of New Episodic Memories. *Cerebral Cortex*, 26(1) :268–278.
- Leuthardt, E. C., Schalk, G., Wolpaw, J. R., Ojemann, J. G., and Moran, D. W. (2004). A brain-computer interface using electrocorticographic signals in humans. *Journal of Neural Engineering*, 1(2) :63–71.
- Li, Z., O'Doherty, J. E., Hanson, T. L., Lebedev, M. A., Henriquez, C. S., and Nicolelis, M. A. L. (2009). Unscented Kalman Filter for Brain-Machine Interfaces. *PLoS ONE*, 4(7) :e6243.
- Liao, L.-D., Chen, C.-Y., Wang, I.-J., Chen, S.-F., Li, S.-Y., Chen, B.-W., Chang, J.-Y., and Lin, C.-T. (2012). Gaming control using a wearable and wireless EEG-based brain-computer interface device with novel dry foam-based sensors. *Journal of NeuroEngineering and Rehabilitation*, 9 :5.
- Lin, J.-S., Chen, K.-C., and Yang, W.-C. (2010). EEG and eye-blinking signals through a Brain-Computer Interface based control for electric wheelchairs with wireless scheme. In *New Trends in Information Science and Service Science (NISS), 2010 4th International Conference On*, pages 731–734. IEEE.

- Ling, C. X., Huang, J., and Zhang, H. (2003). AUC a statistically consistent and more discriminating measure than accuracy. In *IJCAI*, volume 3, pages 519–524.
- Liu, J., Ranka, S., and Kahveci, T. (2008). Classification and feature selection algorithms for multi-class CGH data. *Bioinformatics*, 24(13) :i86–i95.
- Liu, Y., Jiang, X., Cao, T., Wan, F., Mak, P. U., Mak, P.-I., and Vai, M. I. (2012). Implementation of SSVEP based BCI with Emotiv EPOC. In *2012 IEEE International Conference on Virtual Environments Human-Computer Interfaces and Measurement Systems (VECIMS) Proceedings*, pages 34–37. IEEE.
- Lorey, B., Bischoff, M., Pilgramm, S., Stark, R., Munzert, J., and Zentgraf, K. (2009). The embodied nature of motor imagery : The influence of posture and perspective. *Experimental Brain Research*, 194(2) :233–243.
- Lotte, F., Congedo, M., Lécuyer, A., Lamarche, F., Arnaldi, B., and others (2007). A review of classification algorithms for EEG-based brain–computer interfaces. *Journal of neural engineering*, 4.
- Malouin, F., Jackson, P. L., and Richards, C. L. (2013). Towards the integration of mental practice in rehabilitation programs. A critical review. *Frontiers in Human Neuroscience*, 7.
- Markand, O. N. (1990). Alpha rhythms. *Journal of Clinical Neurophysiology*, 7(2) :163–190.
- McFarland, D. J., Krusienski, D. J., Sarnacki, W. A., and Wolpaw, J. R. (2008). Emulation of computer mouse control with a noninvasive brain-computer interface. *Journal of neural engineering*, 5(2) :101.
- McFarland, D. J., Sarnacki, W. A., and Wolpaw, J. R. (2010). Electroencephalographic (EEG) control of three-dimensional movement. *Journal of Neural Engineering*, 7(3) :036007.
- McFarland, D. J. and Wolpaw, J. R. (2008). Brain-computer interface operation of robotic and prosthetic devices. *Computer*, 41(10) :52–56.
- Mehring, C., Nawrot, M. P., de Oliveira, S. C., Vaadia, E., Schulze-Bonhage, A., Aertsen, A., and Ball, T. (2004). Comparing information about arm movement direction in single channels of local and epicortical field potentials from monkey and human motor cortex. *Journal of Physiology-Paris*, 98(4-6) :498–506.
- Mellinger, J., Schalk, G., Braun, C., Preissl, H., Rosenstiel, W., Birbaumer, N., and Kübler, A. (2007). An MEG-based brain-computer interface (BCI). *Neuroimage*, 36(3) :581.
- Mihara, M., Hattori, N., Hatakenaka, M., Yagura, H., Kawano, T., Hino, T., and Miyai, I. (2013). Near-infrared spectroscopy-mediated neurofeedback enhances efficacy of motor imagery-based training in poststroke victims a pilot study. *Stroke*, 44(4) :1091–1098.
- Milekovic, T., Fischer, J., Pistohl, T., Ruescher, J., Schulze-Bonhage, A., Aertsen, A., Rickert, J., Ball, T., and Mehring, C. (2012). An online brain–machine interface using decoding of movement direction from the human electrocorticogram. *Journal of Neural Engineering*, 9(4) :046003.

- Miranda, E. R. (2006). Brain-computer music interface for composition and performance. *International Journal on Disability and Human Development*, 5(2) :119–126.
- Miranda, E. R., Sharman, K., Kilborn, K., and Duncan, A. (2003). On Harnessing the Electroencephalogram for the Musical Braincap. *Computer Music Journal*, 27(2) :80–102.
- Mugler, E. M., Ruf, C. A., Halder, S., Bensch, M., and Kubler, A. (2010). Design and implementation of a P300-based brain-computer interface for controlling an internet browser. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 18(6) :599–609.
- Muller-Putz, G. and Pfurtscheller, G. (2008). Control of an Electrical Prosthesis With an SSVEP-Based BCI. *IEEE Transactions on Biomedical Engineering*, 55(1) :361–364.
- Muller Putz, G., Scherer, R., Neuper, C., and Pfurtscheller, G. (2006). Steady state somatosensory evoked potentials suitable brain signals for brain computer interfaces? *IEEE transactions on neural systems and rehabilitation engineering*, 14(1) :30–37.
- Müller-Putz, G. R., Scherer, R., Pfurtscheller, G., and Rupp, R. (2005). EEG-based neuroprosthesis control : A step towards clinical practice. *Neuroscience letters*, 382(1) :169–174.
- Münßinger, J. I., Halder, S., Kleih, S. C., Furdea, A., Raco, V., Hösle, A., and Kübler, A. (2010). Brain Painting : First Evaluation of a New Brain–Computer Interface Application with ALS-Patients and Healthy Volunteers. *Frontiers in Neuroscience*, 4.
- Nagaoka, T., Sakatani, K., Awano, T., Yokose, N., Hoshino, T., Murata, Y., Katayama, Y., Ishikawa, A., and Eda, H. (2010). Development of a new rehabilitation system based on a brain-computer interface using near-infrared spectroscopy. In *Oxygen Transport to Tissue XXXI*, pages 497–503. Springer.
- Nakhnikian, A., Ito, S., Dwiel, L., Grasse, L., Rebec, G., Lauridsen, L., and Beggs, J. (2016). A novel cross-frequency coupling detection method using the generalized Morse wavelets. *Journal of Neuroscience Methods*, 269 :61–73.
- Naseer, N. and Hong, K.-S. (2015). fNIRS based brain-computer interfaces : A review. *Frontiers in Human Neuroscience*, 9.
- Neuper, C., Müller-Putz, G. R., Scherer, R., and Pfurtscheller, G. (2006). Motor imagery and EEG-based control of spelling devices and neuroprostheses. *Progress in brain research*, 159 :393–409.
- Neuper, C., Scherer, R., Reiner, M., and Pfurtscheller, G. (2005). Imagery of motor actions : Differential effects of kinesthetic and visual–motor mode of imagery in single-trial EEG. *Cognitive Brain Research*, 25(3) :668–677.
- Niedermeyer, E. (2004). The Normal EEG of the Waking Adult. *Electroencephalography : Basic principles, clinical applications, and related fields*, page 167.
- Niedermeyer, E. and da Silva, F. L. (2005). *Electroencephalography : Basic Principles, Clinical Applications, and Related Fields*. Lippincott Williams & Wilkins.

- Nijholt, A. (2008). BCI for games : A 'state of the art'survey. In *International Conference on Entertainment Computing*, pages 225–228. Springer.
- Ojala, M. and Garriga, G. C. (2010). Permutation tests for studying classifier performance. *The Journal of Machine Learning Research*, 11 :1833–1863.
- Ossandon, T., Jerbi, K., Vidal, J. R., Bayle, D. J., Henaff, M.-A., Jung, J., Minotti, L., Bertrand, O., Kahane, P., and Lachaux, J.-P. (2011). Transient Suppression of Broadband Gamma Power in the Default-Mode Network Is Correlated with Task Complexity and Subject Performance. *Journal of Neuroscience*, 31(41) :14521–14530.
- Oude Bos, D. and Reuderink, B. (2008). Brainbasher a bci game.
- Ozkurt, T. E. (2012). Statistically Reliable and Fast Direct Estimation of Phase-Amplitude Cross-Frequency Coupling. *Biomedical Engineering, IEEE Transactions on*, 59(7) :1943–1950.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn : Machine Learning in Python. *Journal of Machine Learning Research*, 12 :2825–2830.
- Penny, W., Duzel, E., Miller, K., and Ojemann, J. (2008). Testing for nested oscillation. *Journal of Neuroscience Methods*, 174(1) :50–61.
- Perelmouter, J., Kotchoubey, B., Kubler, A., Taub, E., and Birbaumer, N. (1999). Language support program for thought-translation-devices. *Automedica*, 18(1) :67–84.
- Pfurtscheller, G. and Berghold, A. (1989). Patterns of cortical activation during planning of voluntary movement. *Electroencephalography and clinical neurophysiology*, 72(3) :250–258.
- Pfurtscheller, G., Guger, C., Müller, G., Krausz, G., and Neuper, C. (2000). Brain oscillations control hand orthosis in a tetraplegic. *Neuroscience letters*, 292(3) :211–214.
- Pfurtscheller, G. and Lopes da Silva, F. H. (1999). Event-related EEG/MEG synchronization and desynchronization : Basic principles. *Clinical neurophysiology*, 110(11) :1842–1857.
- Pfurtscheller, G., Muller-Putz, G., Scherer, R., and Neuper, C. (2008). Rehabilitation with Brain-Computer Interface Systems. *Computer*, 41(10) :58–65.
- Philips, J., del R Millan, J., Vanacker, G., Lew, E., Galán, F., Ferrez, P. W., Van Brussel, H., and Nuttin, M. (2007). Adaptive shared control of a brain-actuated simulated wheelchair. In *Rehabilitation Robotics, 2007. ICORR 2007. IEEE 10th International Conference On*, pages 408–414.
- Pires, G., Castelo-Branco, M., and Nunes, U. (2008). Visual P300-based BCI to steer a wheelchair : A Bayesian approach. In *2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 658–661. IEEE.
- Pistohl, T., Ball, T., Schulze-Bonhage, A., Aertsen, A., and Mehring, C. (2008). Prediction of arm movement trajectories from ECoG-recordings in humans. *Journal of Neuroscience Methods*, 167(1) :105–114.

- Pistohl, T., Schulze-Bonhage, A., Aertsen, A., Mehring, C., and Ball, T. (2012). Decoding natural grasp types from human ECoG. *NeuroImage*, 59(1) :248–260.
- Rebsamen, B. (2009). *A Brain Controlled Wheelchair to Navigate in Familiar Environments*. PhD thesis, national university of singapore.
- Rickert, J. (2005). Encoding of Movement Direction in Different Frequency Ranges of Motor Cortical Local Field Potentials. *Journal of Neuroscience*, 25(39) :8815–8824.
- Rockstroh, B. (1989). *Slow Cortical Potentials and Behaviour*. Urban & Schwarzenberg.
- Rockstroh, B., Elbert, T., Birbaumer, N., Wolf, P., Dückting-Röth, A., Reker, M., Daum, I., Lutzenberger, W., and Dichgans, J. (1993). Cortical self-regulation in patients with epilepsies. *Epilepsy research*, 14(1) :63–72.
- Ruby, P. and Decety, J. (2001). Effect of subjective perspective taking during simulation of action : A PET investigation of agency. *Nature neuroscience*, 4(5) :546–550.
- Sajda, P., Gerson, A., Muller, K.-R., Blankertz, B., and Parra, L. (2003). A data analysis competition to evaluate machine learning algorithms for use in brain-computer interfaces. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 11(2) :184–185.
- Salmelin, R. and Hari, R. (1994). Spatiotemporal characteristics of sensorimotor neuromagnetic rhythms related to thumb movement. *Neuroscience*, 60(2) :537–550.
- Sato, S., Balish, M., and Muratore, R. (1991). Principles of Magnetoencephalography :. *Journal of Clinical Neurophysiology*, 8(2) :144–156.
- Schacter, D. L. (1977). EEG theta waves and psychological phenomena : A review and analysis. *Biological psychology*, 5(1) :47–82.
- Schalk, G., Kubanek, J., Miller, K., Anderson, N., Leuthardt, E., Ojemann, J., Limbrick, D., Moran, D., Gerhardt, L., and Wolpaw, J. (2007). Decoding two-dimensional movement trajectories using electrocorticographic signals in humans. *Journal of neural engineering*, 4(3) :264.
- Schalk, G. and Leuthardt, E. C. (2011). Brain-Computer Interfaces Using Electrocorticographic Signals. *IEEE Reviews in Biomedical Engineering*, 4 :140–154.
- Schalk, G., McFarland, D., Hinterberger, T., Birbaumer, N., and Wolpaw, J. (2004). BCI2000 A General-Purpose Brain-Computer Interface (BCI) System. *IEEE Transactions on Biomedical Engineering*, 51(6) :1034–1043.
- Scherer, R., Zanos, S. P., Miller, K. J., Rao, R. P., and Ojemann, J. G. (2009). Classification of contralateral and ipsilateral finger movements for electrocorticographic brain-computer interfaces. *Neurosurgical Focus*, 27(1) :E12.
- Schreuder, M., Blankertz, B., and Tangermann, M. (2010). A new auditory multi-class brain-computer interface paradigm : Spatial hearing as an informative cue. *PloS one*, 5(4) :e9813.
- Schuster, C., Hilfiker, R., Amft, O., Scheidhauer, A., Andrews, B., Butler, J., Kischka, U., and Ettlin, T. (2011). Best practice for motor imagery : A systematic literature review on motor imagery training elements in five different disciplines. *BMC Medicine*, 9(1).

- Seiler, B. D., Monsma, E. V., and Newman-Norlund, R. D. (2015). Biological Evidence of Imagery Abilities : Intraindividual Differences. *Journal of Sport and Exercise Psychology*, 37(4) :421–435.
- Sellers, E. W. and Donchin, E. (2006). A P300-based brain-computer interface : Initial tests by ALS patients. *Clinical neurophysiology*, 117(3) :538–548.
- Sellers, E. W., Kubler, A., and Donchin, E. (2006). Brain-computer interface research at the University of South Florida Cognitive Psychophysiology Laboratory : The P300 speller. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 14(2) :221–224.
- Sharma, N., Pomeroy, V. M., and Baron, J.-C. (2006). Motor Imagery : A Backdoor to the Motor System After Stroke ? *Stroke*, 37(7) :1941–1952.
- Shibasaki, H. and Hallett, M. (2006). What is the Bereitschaftspotential ? *Clinical Neurophysiology*, 117(11) :2341–2356.
- Shih, J. J. and Krusienski, D. J. (2012). Signals from intraventricular depth electrodes can control a brain–computer interface. *Journal of Neuroscience Methods*, 203(2) :311–314.
- Shih, J. J., Krusienski, D. J., and Wolpaw, J. R. (2012). Brain-Computer Interfaces in Medicine. *Mayo Clinic Proceedings*, 87(3) :268–279.
- Shirvalkar, P. R., Rapp, P. R., and Shapiro, M. L. (2010). Bidirectional changes to hippocampal theta–gamma comodulation predict memory for recent spatial episodes. *Proceedings of the National Academy of Sciences*, 107(15) :7054–7059.
- Siegel, M., Warden, M. R., and Miller, E. K. (2009). Phase-dependent neuronal coding of objects in short-term memory. *Proceedings of the National Academy of Sciences*, 106(50) :21341–21346.
- Silber, M. H., Ancoli-Israel, S., Bonnet, M. H., Chokroverty, S., Grigg-Damberger, M. M., Hirshkowitz, M., Kapen, S., Keenan, S. A., Kryger, M. H., Penzel, T., and others (2007). The visual scoring of sleep in adults. *J Clin Sleep Med*, 3(2) :121–131.
- Sitaram, R., Caria, A., Veit, R., Gaber, T., Rota, G., Kuebler, A., and Birbaumer, N. (2007a). fMRI Brain-Computer Interface : A Tool for Neuroscientific Research and Treatment. *Computational Intelligence and Neuroscience*, 2007 :1–10.
- Sitaram, R., Zhang, H., Guan, C., Thulasidas, M., Hoshi, Y., Ishikawa, A., Shimizu, K., and Birbaumer, N. (2007b). Temporal classification of multichannel near-infrared spectroscopy signals of motor imagery for developing a brain–computer interface. *NeuroImage*, 34(4) :1416–1427.
- Solodkin, A. (2004). Fine Modulation in Network Activation during Motor Execution and Motor Imagery. *Cerebral Cortex*, 14(11) :1246–1255.
- Soto, J. L. and Jerbi, K. (2012). Investigation of cross-frequency phase-amplitude coupling in visuomotor networks using magnetoencephalography. In *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, pages 1550–1553. IEEE.

- Sunny, T., Aparna, T., Neethu, P., Venkateswaran, J., Vishnupriya, V., and Vyas, P. (2016). Robotic Arm with Brain – Computer Interfacing. *Procedia Technology*, 24 :1089–1096.
- Sutter, E. E. (1992). Special Issue on Computers for Handicapped PeopleThe brain response interface : Communication through visually-induced electrical brain responses. *Journal of Microcomputer Applications*, 15(1) :31–45.
- Talairach, J. and Tournoux, P. (1993). *Referentially Oriented Cerebral MRI Anatomy : An Atlas of Stereotaxic Anatomical Correlations for Gray and White Matter*. Thieme.
- Tallon-Baudry, C. and Bertrand, O. (1999). Oscillatory gamma activity in humans and its role in object representation. *Trends in cognitive sciences*, 3(4) :151–162.
- Tallon-Baudry, C., Bertrand, O., Delpuech, C., and Pernier, J. (1997). Oscillatory  $\gamma$ -band (30–70 Hz) activity induced by a visual search task in humans. *The Journal of neuroscience*, 17(2) :722–734.
- Tanaka, K., Matsunaga, K., and Wang, H. O. (2005). Electroencephalogram-based control of an electric wheelchair. *IEEE transactions on robotics*, 21(4) :762–766.
- Tangermann, M., Müller, K.-R., Aertsen, A., Birbaumer, N., Braun, C., Brunner, C., Leeb, R., Mehring, C., Miller, K. J., Müller-Putz, G. R., Nolte, G., Pfurtscheller, G., Preissl, H., Schalk, G., Schlögl, A., Vidaurre, C., Waldert, S., and Blankertz, B. (2012). Review of the BCI Competition IV. *Frontiers in Neuroscience*, 6.
- Taylor, D. M. (2002). Direct Cortical Control of 3D Neuroprosthetic Devices. *Science*, 296(5574) :1829–1832.
- Tort, A. B. L., Komorowski, R., Eichenbaum, H., and Kopell, N. (2010). Measuring Phase-Amplitude Coupling Between Neuronal Oscillations of Different Frequencies. *Journal of Neurophysiology*, 104(2) :1195–1210.
- Trejo, L. J., Rosipal, R., and Matthews, B. (2006). Brain-computer interfaces for 1-D and 2-D cursor control : Designs using volitional control of the EEG spectrum or steady-state visual evoked potentials. *IEEE transactions on neural systems and rehabilitation engineering*, 14(2) :225–229.
- Vadera, S., Marathe, A. R., Gonzalez-Martinez, J., and Taylor, D. M. (2013). Stereoelectroencephalography for continuous two-dimensional cursor control in a brain-machine interface. *Neurosurgical focus*, 34(6) :E3.
- van Elswijk, G., Maij, F., Schoffelen, J.-M., Overeem, S., Stegeman, D. F., and Fries, P. (2010). Corticospinal Beta-Band Synchronization Entails Rhythmic Gain Modulation. *The Journal of Neuroscience*, 30(12) :4481–4488.
- Vanacker, G., Millán, J. d. R., Lew, E., Ferrez, P. W., Moles, F. G., Philips, J., Van Brussel, H., and Nuttin, M. (2007). Context-Based Filtering for Assisted Brain-Actuated Wheelchair Driving. *Computational Intelligence and Neuroscience*, 2007 :1–12.
- Vaughan, T., Mcfarland, D., Schalk, G., Sarnacki, W., Krusinski, D., Sellers, E., and Wolpaw, J. (2006). The Wadsworth BCI Research and Development Program : At Home With BCI. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 14(2) :229–233.

- Velliste, M., Perel, S., Spalding, M. C., Whitford, A. S., and Schwartz, A. B. (2008). Cortical control of a prosthetic arm for self-feeding. *Nature*, 453(7198) :1098–1101.
- Vidal, J. J. (1973). Toward Direct Brain-Computer Communication. *Annual Review of Biophysics and Bioengineering*, 2(1) :157–180.
- Vidal, J. J. (1977). Real-time detection of brain events in EEG. *Proceedings of the IEEE*, 65(5) :633–641.
- Vidaurre, C., Krämer, N., Blankertz, B., and Schlögl, A. (2009). Time domain parameters as a feature for EEG-based brain-computer interfaces. *Neural Networks*, 22(9) :1313–1319.
- Vladimir, V. N. and Vapnik, V. (1995). The nature of statistical learning theory.
- Voytek, B. (2010). Shifts in gamma phase-amplitude coupling frequency from theta to alpha over posterior cortex during visual tasks. *Frontiers in Human Neuroscience*, 4.
- Voytek, B., D’Esposito, M., Crone, N., and Knight, R. T. (2013). A method for event-related phase/amplitude coupling. *NeuroImage*, 64 :416–424.
- Waldert, S., Braun, C., Preissl, H., Birbaumer, N., Aertsen, A., and Mehring, C. (2007). Decoding performance for hand movements : EEG vs. MEG. In *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE*, pages 5346–5348. IEEE.
- Waldert, S., Pistohl, T., Braun, C., Ball, T., Aertsen, A., and Mehring, C. (2009). A review on directional information in neural signals for brain-machine interfaces. *Journal of Physiology-Paris*, 103(3-5) :244–254.
- Waldert, S., Preissl, H., Demandt, E., Braun, C., Birbaumer, N., Aertsen, A., and Mehring, C. (2008). Hand Movement Direction Decoded from MEG and EEG. *Journal of Neuroscience*, 28(4) :1000–1008.
- Watrous, A. J., Deuker, L., Fell, J., and Axmacher, N. (2015). Phase-amplitude coupling supports phase coding in human ECoG. *eLife*, 4 :e07886.
- Weinberger, K. Q., Blitzer, J., and Saul, L. K. (2005). Distance metric learning for large margin nearest neighbor classification. In *Advances in Neural Information Processing Systems*, pages 1473–1480.
- Weiskopf, N., Mathiak, K., Bock, S., Scharnowski, F., Veit, R., Grodd, W., Goebel, R., and Birbaumer, N. (2004). Principles of a Brain-Computer Interface (BCI) Based on Real-Time Functional Magnetic Resonance Imaging (fMRI). *IEEE Transactions on Biomedical Engineering*, 51(6) :966–970.
- Wieland, M. and Pittore, M. (2014). Performance Evaluation of Machine Learning Algorithms for Urban Pattern Recognition from Multi-spectral Satellite Images. *Remote Sensing*, 6(4) :2912–2939.
- Wilson, J. A., Felton, E. A., Garell, P. C., Schalk, G., and Williams, J. C. (2006). ECoG factors underlying multimodal control of a brain-computer interface. *IEEE transactions on neural systems and rehabilitation engineering*, 14(2) :246–250.

- Wolpaw, J., McFarland, D., Vaughan, T., and Schalk, G. (2003). The wadsworth center brain-computer interface (bci) research and development program. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 11(2) :204–207.
- Wolpaw, J. R. (2007). Brain-computer interfaces as new brain output pathways : BCIs as new brain outputs. *The Journal of Physiology*, 579(3) :613–619.
- Wolpaw, J. R., Birbaumer, N., McFarland, D. J., Pfurtscheller, G., Vaughan, T. M., and others (2002). Brain computer interfaces for communication and control. *Clinical neurophysiology*, 113(6) :767–791.
- Wolpaw, J. R. and McFarland, D. J. (2004a). Control of a two-dimensional movement signal by a noninvasive brain-computer interface in humans. *Proceedings of the National Academy of Sciences of the United States of America*, 101(51) :17849–17854.
- Wolpaw, J. R. and McFarland, D. J. (2004b). Control of a two-dimensional movement signal by a noninvasive brain-computer interface in humans. *Proceedings of the National Academy of Sciences of the United States of America*, 101(51) :17849–17854.
- Wolpaw, J. R., McFarland, D. J., Neat, G. W., and Forneris, C. A. (1991). An EEG-based brain-computer interface for cursor control. *Electroencephalography and clinical neurophysiology*, 78(3) :252–259.
- Worrell, G., Jerbi, K., Kobayashi, K., Lina, J., Zelmann, R., and Le Van Quyen, M. (2012). Recording and analysis techniques for high-frequency oscillations. *Progress in neurobiology*, 98(3) :265–278.
- Wu, W., Black, M., Mumford, D., Gao, Y., Bienenstock, E., and Donoghue, J. (2004). Modeling and Decoding Motor Cortical Activity Using a Switching Kalman Filter. *IEEE Transactions on Biomedical Engineering*, 51(6) :933–942.
- Wu, W., Black, M. J., Gao, Y., Bienenstock, E., Serruya, M., and Donoghue, J. P. (2002). Inferring hand motion from multi-cell recordings in motor cortex using a Kalman filter. In *SAB'02-Workshop on Motor Control in Humans and Robots : On the Interplay of Real Brains and Artificial Devices*, pages 66–73.
- Wu, W., Black, M. J., Gao, Y., Bienenstock, E., Serruya, M., Shaikhouni, A., and Donoghue, J. P. (2003). Neural decoding of cursor motion using a Kalman filter. *Advances in neural information processing systems*, pages 133–140.
- Wu, W., Gao, Y., Bienenstock, E., Donoghue, J. P., and Black, M. J. (2006). Bayesian population decoding of motor cortical activity using a Kalman filter. *Neural computation*, 18(1) :80–118.
- Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Yu, P. S., Zhou, Z.-H., Steinbach, M., Hand, D. J., and Steinberg, D. (2008). Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1) :1–37.
- Yanagisawa, T., Hirata, M., Saitoh, Y., Kato, A., Shibuya, D., Kamitani, Y., and Yoshimine, T. (2009). Neural decoding using gyral and intrasulcal electrocorticograms. *NeuroImage*, 45(4) :1099–1106.

- Yanagisawa, T., Hirata, M., Saitoh, Y., Kishima, H., Matsushita, K., Goto, T., Fukuma, R., Yokoi, H., Kamitani, Y., and Yoshimine, T. (2012a). Electrocorticographic control of a prosthetic arm in paralyzed patients. *Annals of Neurology*, 71(3) :353–361.
- Yanagisawa, T., Yamashita, O., Hirata, M., Kishima, H., Saitoh, Y., Goto, T., Yoshimine, T., and Kamitani, Y. (2012b). Regulation of Motor Representation by Phase-Amplitude Coupling in the Sensorimotor Cortex. *The Journal of Neuroscience*, 32(44) :15467–15475.
- Yu, L. and Liu, H. (2004). Redundancy based feature selection for microarray data. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 737–742. ACM.
- Zickler, C., Halder, S., Kleih, S. C., Herbert, C., and Kübler, A. (2013). Brain painting : Usability testing according to the user-centered design in end users with severe motor paralysis. *Artificial intelligence in medicine*, 59(2) :99–110.
- Zimmermann, R., Marchal-Crespo, L., Edelmann, J., Lambercy, O., Fluet, M.-C., Rieiner, R., Wolf, M., and Gassert, R. (2013). Detection of motor execution using a hybrid fNIRS-biosignal BCI : A feasibility study. *Journal of neuroengineering and rehabilitation*, 10(1) :1.